Calcul haute performance pour la détection de rayon Gamma

Pierre Aubert

université PARIS-SACLAY

NNT : 2018SACLV058

Thèse de doctorat

Informatique Hautes Performances pour la détection de rayons gammas

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université de Versailles-Saint-Quentin-en-Yvelines

Ecole doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Annecy, le 4 octobre 2018, par

PIERRE AUBERT

Composition du Jury :

Giovanni Lamanna Directeur de recherche, LAPP Annecy	Président
Jean-Pierre Ernenwein Professeur, CPPM Marseille	Rapporteur
Catherine Lambert Directrice de recherche, CERFACS Toulouse	Rapporteur
Jean-Marc Delosme Professeur, Université d'Evry-Val d'Essonne	Examinateur
Karl Kosack Chercheur, CEA Saclay	Examinateur
Nahid Emad Professeure, MDLS Saclay	Directrice de thèse
Gilles Maurin Maître de conférence, LAPP Annecy	Co-directeur de thèse

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Informatique Hautes Performances pour la détection de rayons gammas

Mots clés : astronomie Gamma de très haute énergie, parallélisme multi-niveaux, analyse de donnés (calibration/reconstruction/réjection), ré-utilisabilité, séquentielle/parallèle, programmation hétérogène à large échelle

Résumé : La nouvelle génération d'expériences de physique produira une quantité de données sans précédent. Cette augmentation du flux de données cause des bouleversements techniques à tous les niveaux, comme le stockage des données, leur analyse, leur dissémination et leur préservation. Le projet CTA sera le plus grand observatoire d'astronomie gamma au sol à partir de 2021. Il produira plusieurs centaines de Péta-octets de données jusqu'en 2030 qui devront être analysées, stockée, compressées, et réanalysées tous les ans. Ce travail montre comment optimiser de telles analyses de physique avec les techniques de l'informatique hautes performances par le biais d'un générateur de format de données efficace, d'optimisation bas niveau de l'utilisation du pipeline CPU et de la vectorisation des algorithmes existants, un algorithme de compression rapide d'entiers et finalement une nouvelle analyse de données basée sur une méthode de comparaison d'image optimisée.

Title : High Performance Computing for gamma ray detection

Keywords : High Energy gamma astronomy, multiple parallelism level, data analysis (calibration/reconstruction/rejection), reusability, sequential/parallel, large scale heterogeneous programming

Abstract : The new generation research experiments will introduce huge data surge to a continuously increasing data production by current experiments. This increasing data rate causes upheavals at many levels, such as data storage, analysis, diffusion and conservation. The CTA project will become the utmost observatory of gamma astronomy on the ground from 2021. It will generate hundreds Peta-Bytes of data by 2030 and will have to be stored, compressed and analyzed each year. This work address the problems of data analysis optimization using high performance computing techniques via an efficient data format generator, very low level programming to optimize the CPU pipeline and vectorization of existing algorithms, introduces a fast compression algorithm for integers and finally exposes a new analysis algorithm based on efficient pictures comparison.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Table des matières

Chapitre 1

Astronomie gamma contemporaine

Sujet du chapitre 1 –

Ce chapitre décrit le contexte physique de cette thèse : l'étude des sources de rayons cosmiques. L'historique de cette découverte et les recherches associées y sont présentés suivit par les techniques de détection et d'analyse actuelles. Les infrastructures liées au calcul ainsi que les limites de la méthode y sont également présenté.

Sommaire

Les rayons cosmiques constituent un rayonnement qui baignent la Terre. Ce rayonnement est composé principalement de particules chargées (protons, noyaux lourd, électrons, positrons). Depuis plus de 100 ans des scientifiques se succèdent dans cette étude.

1.1 Historique

La première manifestation de rayons cosmiques fut observée par Charles Augustin de Coulomb (1736-1806). Durant ses travaux en électrostatique il observa qu'une sphère chargée et isolée se déchargeait lentement. Malheureusement, ses connaissances ne lui permirent pas de résoudre cette énigme.

En 1894, Pierre Curie réussit à mesurer l'intensité d'un rayonnement ionisant à l'aide d'un électroscope (définition 1.1.1). Un rayonnement ionisant est constitué de particules ionisantes, chargées. Ces particules transportent des charges électriques qui interagissent avec la matière, ce qui a pour effet de décharger un électroscope.

Définition 1.1.1 – **Electroscope**

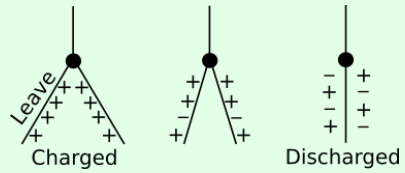


FIGURE 1.1 – Décharge d'un électroscope.

Un électroscope est un instrument composé de deux fines feuilles de métal (or, cuivre ou aluminium). Les feuilles sont connectées ensemble via un axe métallique utilisé pour charger et décharger l'électroscope. Charger l'électroscope consiste à ajouter des charges positives ou négatives en le touchant avec une matière électrostatique. Dès lors, les charges identiques se repoussent et les feuilles s'écartent. L'électroscope est chargé. Ensuite, les feuilles se rapprochent lentement jusqu'à ce que l'électroscope soit totalement déchargé (figure 1.1). Généralement un électroscope est protégé dans un globe en verre (figure 1.3).

FIGURE 1.3 – Électroscope à feuilles d'or, Université "La Sapienza", Rome.

En 1901, Charles Thomson Rees Wilson (prix nobel 1927, figure 1.2), Elster et Geitel étudiaient l'ionisation de gaz. Ils furent confrontés à un problème d'ionisation résiduelle : même les électroscopes bien protégés des radiations se déchargeaient lentement. Á cette époque, les physiciens connaissaient la création de paires d'ions dans les gaz. Mais la raison de cette ionisation demeurait un mystère. Wilson pensa à une radiation venant de l'espace.

FIGURE 1.2 – Charles Thomson Rees Wilson, prix nobel 1927.

Dès lors, la communauté des physiciens fut divisée en deux groupes. Le premier groupe était en accord avec Wilson et le second prétendait que la radioactivité terrestre (récemment découverte par Henri Bécquerel, 1896) était en cause. Les deux groupes tentèrent de prouver leurs théories respectives. Ce débat répondit à la question et permit de mieux comprendre la radioactivité et les rayons cosmiques.

Rutherford fut le premier à mettre en évidence le fait que l'ionisation n'était pas causée par l'instrument. En effet, l'ionisation diminuait lorsque l'on augmentait l'épaisseur de plomb entourant l'électroscope.

Cependant le sol était toujours près du dispositif. Des expériences en mer furent menées pour s'assurer que la radioactivité des roches n'était pas en cause. Mais de petites ionisations étaient toujours observées.

La prochaine étape fut naturellement d'élever le dispositif du sol. En 1910, Théodore Wulf fut le premier à installer un instrument au sommet de la tour Eiffel (environ 300 m de hauteur). Il observa que le temps de décharge était plus important à mesure que la hauteur augmentait. Ce résultat tendait à confirmer l'hypothèse que les radiations provenant de la

Terre étaient en cause. Mais l'effet d'ionisation était plus important que la seule ionisation due aux sources radioactives [1]. Les physiciens furent convaincus de la nécessité de répéter l'expérience à une altitude bien plus importante que les précédentes car quelque chose atténuait la diminution attendue.

Le 7 août 1912, Victor Franz Hess (figure 1.4) mesura l'ionisation au cours de plusieurs vols en ballon. Il atteignit l'altitude de 5 350 m (figure 1.5). Il voyagea sur plus de 150 km et montra que l'ionisation augmentait considérablement à partir d'un kilomètre d'altitude. Il conclue que cette radiation provenait probablement de l'espace [2]. Deux ans plus tard, Kolhörrster vola à plusieurs reprise jusqu'à 9 200 m et confirma ce résultat [3].

FIGURE 1.4 – Victor Franz Hess (1883-1964), prix nobel 1936.

Cependant, ces résultats ne furent acceptés qu'après discussions et l'existence de rayons extraterrestres fut établit en 1926. Depuis cette date de nombreux physiciens furent attirés par ce nouveau champs de recherche, la physique des particules était née. Victor Hess reçu le prix nobel en 1936 pour la découverte des rayons cosmiques.

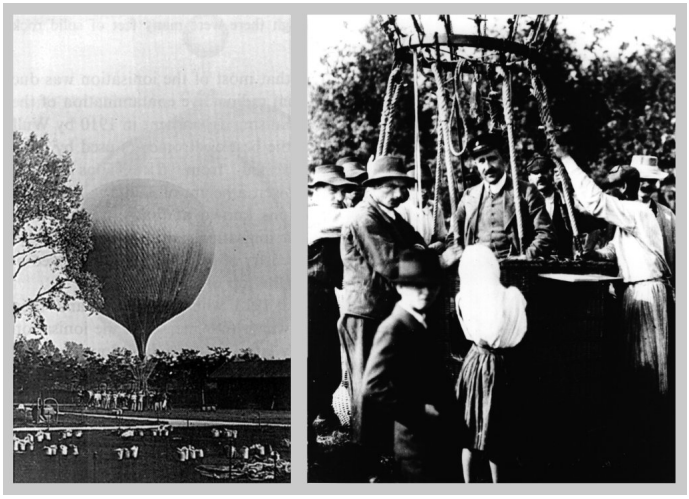


FIGURE 1.5 – Victor Franz Hess (1883-1964), prix nobel 1936, prêt pour un vol en ballon.

Robert Millikan était convaincu que les "radiations de Hess" étaient des photons de très hautes énergie, appelés rayons gamma, avant d'entrer dans l'atmosphère. Il les appela les rayons cosmiques [4] et essaya d'imposer son idée à la communauté. Arthur Compton (prix nobel 1927) quant à lui, était convaincu de la nature corpusculaire des rayons cosmiques.

Le débat animé entre les deux scientifiques fut rapporté dans le New-York Times du 31 décembre 1932 (figure 1.6). Mais l'hypothèse de Compton semblait bien admise depuis 1927.


The New York Times

MILLIKAN RETORTS HOTLY TO COMPTON IN COSMIC RAY CLASH

Debate of Rival Theorists Brings Drama to Session of Nation's Scientists.

THEIR DATA AT VARIANCE

New Findings of His Ex-Pupil Lead to Thrust by Millikan at 'Less Cautious' Work.

FIGURE 1.6 – La première page du New-York Times du 31 décembre 1932 first page. Le débat entre Compton et Millikan sur les rayons cosmiques.

En effet, la même année, Clay découvrit l'effet de latitude [5], pendant un voyage entre Amsterdam et Bataria. Les radiations diminuaient lorsque l'on se rapprochait de l'équateur. Arthur Compton vérifia cette hypothèse lors de plusieurs expéditions à travers le monde et fut rejoint par plus de 60 scientifiques [6]. Quelques années plus tard Bothe et Kolhörrster montrèrent que ces radiations étaient chargées à l'aide des compteurs Geiger-Müller en coïncidence. Le champ magnétique terrestre déviait ces particules [7].

En 1927, Dimitry Skobelzyn (1892-1990) montra que ces particules étaient chargées. Il prouva que les trajectoires de ces particules étaient déviées par un champ magnétique, à l'aide d'une chambre de Wilson [8] (définition 1.1.2). Il découvrit également que ces rayons arrivaient en groupe et les appela gerbes de particules.

Définition 1.1.2 – **Chambre de Wilson**

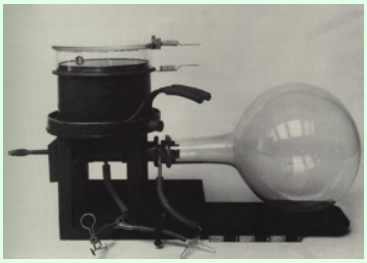


FIGURE 1.7 – Premier type de la chambre de Wilson.

La chambre de Wilson (ou chambre à brouillard, figure 1.7) fut inventée en 1912 par Charles Thomson Rees Wilson (figure 1.2). Á l'origine, elle fut mise au point pour simuler la formation des nuages.

La chambre de Wilson est remplie de gaz. Elle fonctionne avec une violente dépression. Lorsqu'une particule ionisante entre dans la chambre, des goûtes apparaissent à cause des ions créés par l'ionisation du gaz. Une photographie est prise et les goûtes dessinent les trajectoires des particules. Le taux de déclenchement d'une telle chambre est faible : une photographies toute les deux minutes (pour laisser le temps aux trajectoires des particules de disparaître et au gaz de se stabiliser).

Cette méthode n'est plus utilisée actuellement mais fut améliorée plusieurs fois : les chambres à bulles, chambre à étincelles, chambre à fils.

En 1932, Rossi étudia la formation des gerbes [9]. Il observa qu'elles passaient à travers

un écran en plomb. En augmentant l'épaisseur de cet écran, il observa que leur intensité était amplifiée jusqu'à un maximum supérieur à l'intensité obtenue sans écran, puis diminuait. Il en conclut qu'un phénomène amplifiait l'intensité de la gerbe avant de l'absorber. Ce résultat contre intuitif fut confirmé par Pfotzer qui observa les même effets à différentes altitudes dans l'atmosphère. Il fut finalement accepté grâce aux mesures précises de Pfotzer.

Parallèlement, les observations de rayons cosmiques ont permis des avancées dans les autres domaines de la physique. En 1929, Paul Dirac (prix nobel 1933) construisit les fondements de la mécanique quantique. Sa théorie prédisait l'existence de particules à énergie négative, les antiparticules [10] (ce qui lui semblait un non sens). En 1932, C.D. Anderson pris une photographie de la trajectoire d'un électron positif, il l'appela plus tard positron [11]. Il obtint le prix nobel en 1936 pour la découverte des antiparticules.

Dès lors, ce fut la fin des pionniers. Les instruments devinrent de plus en plus complexes et précis. De plus en plus de physiciens vinrent étudier cette nouvelle radiation. Rapidement, deux composantes furent trouvées dans cette radiation. Une faible, composée de photons et d'électrons et une forte, pouvant traverser un mètre de plomb. Mais aucune particule n'était connue pour avoir cette propriété.

En 1937, Neddermeyer et Anderson découvrirent la particule de la composante forte. En premier lieu, ils l'appelèrent le mésotron mais il devint le muon quelques temps plus tard [12].

En 1938, Pierre Auger, R. Maze et T. Grivet-Mezer utilisèrent des compteurs Geiger-Müller afin de mesurer le nombre de particules détectées en coïncidence, ou taux de coïncidences, au Pic du midi. Les compteurs étaient séparés de 1 à 70 mètres. Les taux diminuaient étrangement en accord avec la distance. De ce fait, les coïncidences seules ne pouvaient pas expliquer la mesure, un phénomène supplémentaire était donc attendu. Auger et ses collaborateurs connaissaient les résultats de Rossi et une nouvelle théorie proposée par Carlson et Hoppenheimer entre 1933 et 1937 : la théorie des gerbes. Ils proposèrent que les particules détectées au sol provenaient d'une gerbe, produite par une seule particule initiale interagissant avec l'atmosphère. Pierre Auger estima que l'énergie d'une telle particule excédait un million de GeV (voir définition 1.1.3). Ces gerbes furent appelées les gerbes atmosphériques d'Auger [13]. Dix ans plus tard, la formation de telles gerbes fut mis en évidence grâce aux recherches en physique des particules.

Définition 1.1.3 – **Électron-Volt, (eV)**

L'électron-volt (eV) est une unité classique pour l'énergie d'une particule. Il décrit le gain d'énergie d'un électron soumis à une tension d'un volt. Mais cela peut être déroutant pour un non physicien. La plus basse énergie des rayons cosmiques qui arrivent sur Terre est 10^8 eV, elle correspond à l'énergie cinétique d'une mouche drosophile en vol. L'énergie d'un rayon cosmique d'ultra haute énergie, 10^{21} eV, correspond à celle d'un train lancé à 200 km/h. Dans la vie de tous les jours, ces énergies peuvent ne pas paraître importantes, mais cela décrit l'énergie d'une **seule** particule.

La recherche moderne des rayons cosmiques a pris son essor après la deuxième guerre

mondiale, lorsque les scientifiques étudièrent les particules messagères provenant de l'espace afin de déterminer les sources de rayons cosmiques.

Tout d'abord, ils cherchèrent à détecter directement les rayons cosmiques. Il envoyèrent des chambres à émulsion avec des ballons sondes, comme pour l'expérience JACEE [14] dans les années 80. Ces chambres devaient être légères, inférieures à 200 kg. Si le détecteur est lancé à l'aide d'un ballon, il revient lorsque le ballon explose, après qu'il ait atteint la stratosphère. Malheureusement quelques expériences furent endommagées ou perdues. Certaines furent mise dans des cargos ou dans les soudes du Concorde jusqu'en 1978, pour l'expérience franco-japonaise ECHOS [15].

La détection directe de rayon cosmiques continua avec l'expérience AMS [16], installée sur l'ISS (International Space Station), et lancée à bord de la navette Discovery en 1998. L'expérience suivante, AMS 2, débuta le 19 mai 2011, et continue à collecter des données sur les rayons cosmiques.

Quelques expériences cherchèrent aussi les rayons cosmiques de manière indirecte, en détectant les grosses gerbes atmosphériques (ou gerbes d'Auger) avec des dispositifs au sol. Entre 1959 et 1963, Volcano Ranch fut la première expérience à détecter un événement très énergétique (plus de 10^{20} eV). L'expérience suivante, Fly's eye, détecta l'événement le plus énergétique de tous les temps, estimé à 3.2×10^{21} eV [1] [17] le 15 octobre 1991.

En 1997, un second télescope fut ajouté à Fly's eye, et devint HIRES. Pendant ce temps le Akeno Giant Air Shower Array, AGASA [18] prenait des données depuis 1990 via des scintillateurs et des détecteurs de muons. Malheureusement, ces deux expériences eurent des résultats différents, probablement causés par la différence des techniques de détection.

De nos jours, l'observatoire Pierre Auger prend des données en utilisant les deux techniques de détections précédentes pour étudier les rayons cosmiques d'ultra hautes énergie (figure 1.8). Ceci dans le but de permettre aux scientifiques de comprendre l'origine du rayonnement cosmique, sa composition et sa nature.

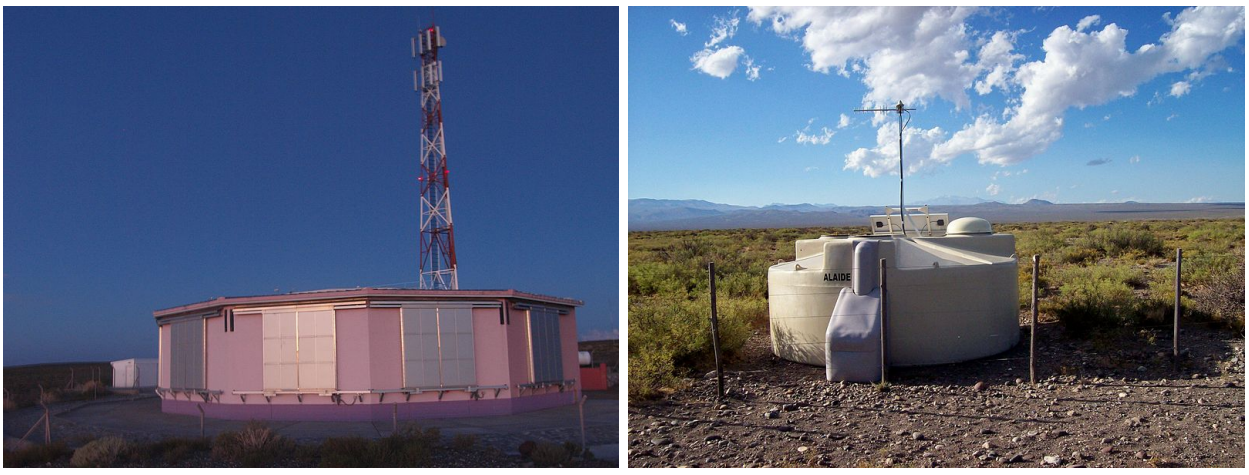


FIGURE 1.8 – À gauche, un scintillateur de l'expérience Auger. À droite, une cuve de détection remplie d'eau.

1. Mais cette estimation est rejetée par une partie de la communauté scientifique.

Il permis de cataloguer de nombreuses sources de rayons cosmiques. La figure 1.9 montre la carte du ciel des sources de rayons cosmiques d'ultra haute énergie ($E > 10^{17}$ eV) déterminées par l'observatoire Pierre Auger [19].

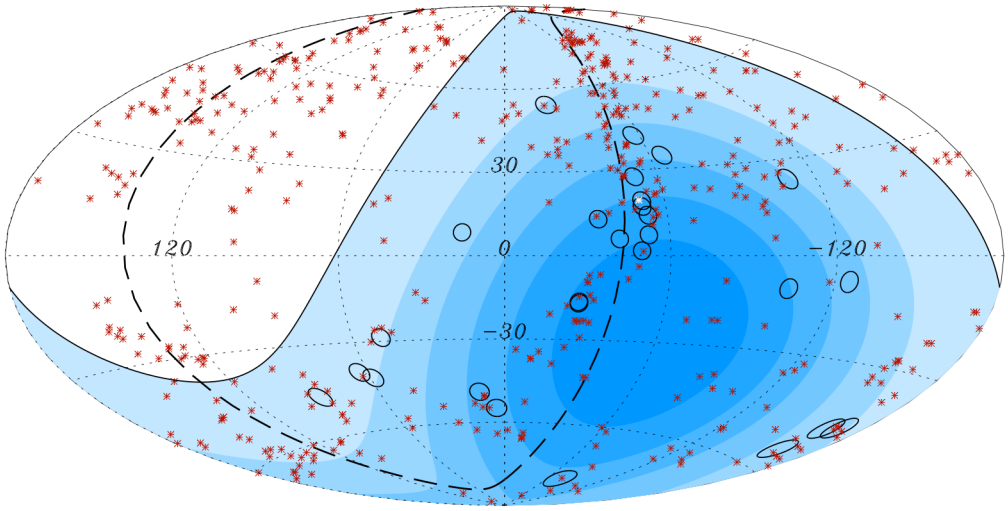


FIGURE 1.9 – Carte des positions des sources de rayons cosmiques d'ultra haute énergie ($E > 10^{17}$ eV) déterminées par l'observatoire Pierre Auger [19].

Ces informations proviennent des manuscrits de thèse de Gilles maurin, Nukri Komin et des notes de cours de Damir Buskulic, mon professeur d'astro-particules.

1.2 Phénomènes physiques

1.2.1 Composition et énergie

Les expériences les plus récentes montrent que le rayonnement cosmique qui arrive sur Terre est composé principalement de protons (89%), de noyaux lourds (10%), d'électrons et de positrons (1%). Leur énergie s'étale sur 13 ordres de grandeur, de l'énergie d'une drosophile en vol à un train lancé à 200 km/h. En termes physique, de 10^8 eV à 10^{21} eV. Leur flux varie sur 32 ordres de grandeur, d'une particule par cm^2 et par seconde pour les basses énergies à une particule par km^2 et par siècle pour les plus hautes. En terme physique, de 10^4 $(m^2\,sr\,s\,GeV)^{-1}$ à 10^{-28} $(m^2\,sr\,s\,GeV)^{-1}$.

Le lien entre un flux et l'énergie correspondante est appelée un spectre. Le spectre des rayons cosmiques est illustré par la figure 1.10. Il suit une loi de puissance sur plusieurs ordres de grandeurs. Ces particules ont été accélérées dans des accélérateurs spatiaux.

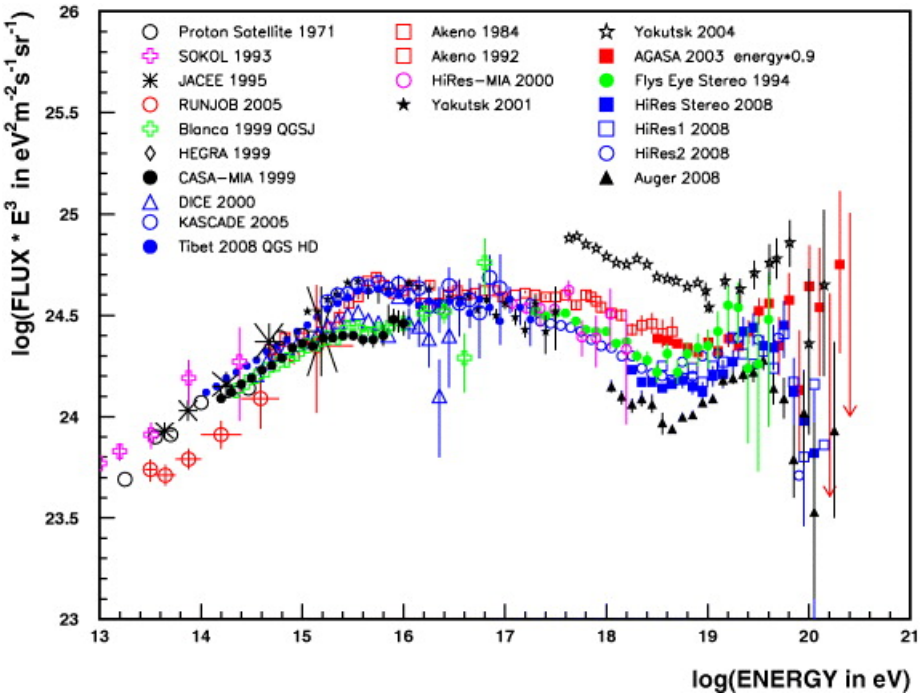


FIGURE 1.10 – Spectre des rayons cosmiques [20].

1.2.2 Les sources

Par définition, l'origine des rayons cosmiques ne répond pas à la question de la création de particules, en terme de nouvelles particules créées. Les futurs rayons cosmiques sont des particules comme les protons et les électrons dans l'espace interstellaire. L'origine du rayonnement cosmique cherche à déterminer de quelles façons et par quels mécanismes ces particules peuvent être accélérées.

Aujourd'hui, les sources de rayons cosmiques sont apparentées à leurs énergies :

• Les basses énergies sont affectées par les modulations du Soleil (des variations dues à l'activité du Soleil ont été observées). Le Soleil envoie sur Terre des électrons de basse énergie. Ces électrons interagissent avec le champ magnétique terrestre et produisent des aurores boréales et australes aux pôles.

• Les rayon cosmiques de hautes énergies sont accélérés par des sources galactiques via des mécanismes de Fermi (définitions 1.2.1). Ces sources sont :

— **Les vestiges de supernova**, aussi appelés éjectas. L'explosion d'une étoile super-massive est une supernova. La matière éjectée lors de l'explosion peut accélérer des particules via des mécanismes de Fermi.

— **Les nébuleuses de pulsars**. Une supernova peut produire un résidu, une étoile à neutron ou un trou noir. Un pulsar est une étoile à neutron en rotation très rapide qui émet deux jets, diamétralement opposés. Ils tournent avec elle et accélèrent des particules. La régularité des pulars milliseconde est parfois plus précise que celle des horloges atomiques grâce à leur densité.

— **Le PeVatron**, est une source présente dans la galaxie qui produit des rayons cosmiques de très hautes énergies (PeV, peta-electron-volt, définition 1.1.3), les mécanismes d'accélération en jeu sont inconnus mais pourraient être dûs au trou noir super massif au centre de la Voie Lactée [21].

• Les rayons cosmiques de très hautes énergies (10^{18} eV) ne sont pas déviés par le champ magnétique galactique. Leurs sources, encore inconnues, semblent extra-galactiques. Les candidats potentiels sont :

— **Les sursauts gamma**, (GRB, Gamma Ray Bursts), produisent des bouffées de rayons gamma pendant un temps court (de quelques secondes à quelques minutes). Ils sont causés par l'effondrement gravitationnel d'une étoile géante ou la fusion de deux étoiles à neutron.

— **Les noyaux actifs de galaxies**, (AGN, Active Galactic Nuclei). Les noyaux actifs de galaxie possèdent un bulbe plus brillant que le reste de la galaxie. Le bulbe contient un trou noir super-massif qui accrète de la matière. L'accrétion de matière autour du trou noir produit des jets relativistes. Lorsqu'un jet pointe en direction de la Terre, cet AGN est alors appelé blazar.

Les rayons cosmiques sont chargés. Il sont donc déviés par les champs magnétiques. Leurs trajectoires sont chaotiques dans l'espace à cause de ces derniers. Donc, reconstruire la direction d'un rayon cosmique ne donne aucune information sur la position de sa source. Heureusement, les mécanismes d'accélération des rayons cosmiques produisent, en même temps et au même endroit, des ondes radio, des rayons X, et dans certains cas des rayons gamma, des neutrinos ou des ondes gravitationnelles (définition 1.2.2). Ces messagers ne sont pas déviés par les champs magnétiques.

Un rayon gamma (γ) est un photon de haute énergie, à partir de quelques centaines de keV (kilo-electron-volt). Ils sont produits naturellement sur Terre par des éléments radioactifs et dans l'espace par des sources qui seront détaillées dans ce qui suit.

Le but de l'astronomie gamma est d'étudier les objets galactiques et extra-galactiques comme les pulsars, les vestiges de supernova, les nébuleuses, les noyaux actifs de galaxies

Définition 1.2.1 – **Mécanismes d'accélération de Fermi**

Les mécanismes d'accélération de Fermi sont très populaires et ont été proposés par Enrico Fermi en 1949 [22]. En astrophysique, un choc défini une surface ou ces propriétés physiques semblable à la pression, la température, la densité, un champ magnétique ou électrique subissent d'importantes variations. Cela se produit lorsque la vitesse d'un objet est plus importante que la vitesse du son dans le milieu considéré. Dans un milieu dense, les atomes de front (ceux de l'objet) entrent en collision avec ceux du milieu. La température du milieu augmente et le choc perd une partie de son énergie cinétique sous forme de chaleur. Cependant, un milieu astrophysique est fortement dilué. Dans ce cas, les collisions ne sont pas réelles, mais si le milieu supersonique est un plasma, il créé un champs magnétique qui interagit avec les autres particules du milieu. Ainsi, les chocs astrophysiques sont en réalité des chocs sans collision.

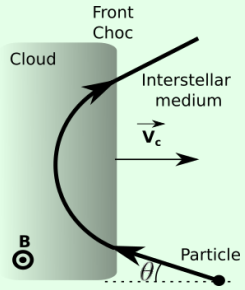


FIGURE 1.11 – Particule chargée réfléchie par un nuage magnétique.

Mécanisme de Fermi du premier ordre :

Le mécanisme d'accélération de Fermi du premier ordre joue avec le concept de miroirs magnétiques. Considérons un champ magnétique en mouvement à vitesse constante. Quand une particule chargée entre dans le nuage, le champ magnétique, **B**, défléchi la particule hors du nuage (figure 1.11). Puisque c'est un choc sans collision, la particule ne perd pas d'énergie et est accélérée grâce à la vitesse du nuage. Pour accélérer, une particule doit traverser l'onde de choc à nouveau. Trois cas sont possible. Premièrement, les particules qui pénètrent dans le nuage avec un angle (angle θ dans la figure 1.11) ou une vitesse trop faible ne pourront pas sortir du nuage à nouveau. Deuxièmement, elles peuvent quitter le nuage (en croisant de nouveau l'onde de choc) et ne pas revenir, dû à leurs vitesses ou leurs trajectoires. Finalement, les particules peuvent traverser l'onde de choc plusieurs fois et être grandement accélérées. Plus une particule traverse l'onde de choc et plus elle est accélérée.

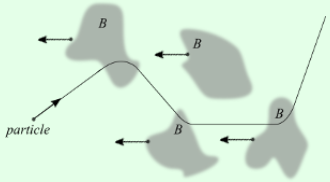


FIGURE 1.12 – Particule chargée qui est défléchie par des nuages magnétiques

Mécanisme de Fermi du second ordre :

L'accélération de Fermi du second ordre n'implique pas une réflexion totale de la particule, nécessaire pour le premier ordre. la figure 1.12 illustre le mécanisme d'accélération avec des nuages venant à l'encontre d'une particule.
Ce phénomène peut être comparé à une course de bateaux (trimaran or catamaran) qui accélèrent avec un vent transversal. La propriété remarquable de cette situation est que la vitesse d'un bateau peut, dans certains cas, être supérieure à celle du vent.
Dans notre cas, le bateau est la particule et le vent représente les nuages. La particule passe à travers de multiple nuages et est défléchie à chaque fois, sans collision donc sans perdre d'énergie.
La particule gagne de l'énergie lorsqu'elle se déplace à l'encontre du nuage et en perd dans le cas contraire.

Définition 1.2.2 – **Mécanismes de perte d'énergie des rayons cosmiques**
Quelques interactions de rayons cosmiques produisent des messagers comme des photons ou des neutrinos.

- Émission d'électrons :
  - L'interaction avec un champ magnétique produit des ondes radios ou des rayons X, appelé effet Synchrotron.
  - L'interaction avec un photon produit des rayons gamma du GeV au TeV. C'est l'effet Inverse-Compton (IC).
  - L'interaction avec un atome produit des rayons gamma du GeV au TeV, appelé bremsstrahlung (ou rayonnement de freinage).
- L'émission de protons se produit lorsqu'ils interagissent avec la matière. Cette interaction produit des pions. Les pions chargés se désintègrent en neutrinos et en pions neutres qui se désintègrent en rayons gamma du GeV au TeV. Cette interaction est appelée diffusion inélastique.

(AGN), les blazars et les galaxies à flambée d'étoiles.

1.3 Techniques actuelles en astronomie gamma

En astronomie classique, une observation consiste à collecter des photons provenant des sources étudiées (étoiles, nébuleuses, etc). Des miroirs concentrent cette lumière sur une caméra qui produit les images à analyser.

Les rayons gamma sont très énergétiques. Ils ne sont pas déviés par des miroirs et passent au travers de la matière (ceci est appelé irradiation pour le corps humain). L'étude de ces rayons gamma n'est possible que si ils traversent suffisamment de matière, de manière à perdre assez d'énergie. Un tel dispositif est appelé un calorimètre. Quand le flux de rayons gamma est suffisamment important, le calorimètre a une taille humaine. Ce genre de calorimètre est utilisé dans des expériences spatiales comme *Fermi*-LAT [23], avec une sensibilité aux rayons gamma de 20 MeV à 300 GeV. Lorsqu'un rayon gamma interagit avec le calorimètre, il créé une paire électron-positron, appelée paire e^+e^-. Les trajectoires des particules produites sont déterminées de manière à reconstruire les caractéristiques du rayon gamma initial.

Quand le flux de rayons gamma n'est plus suffisant, aux alentours de 10^{12} eV (TeV), un calorimètre plus grand est utilisé. Dans ce cas, l'atmosphère fait office de calorimètre.

1.3.1 Intéraction d'un gamma avec l'atmosphère

Lorsqu'un rayon gamma entre dans l'atmosphère, il créé une gerbe de particules de hautes énergies composée d'électrons et de photons de plus basses énergies. Ces particules secondaires sont absorbées par l'atmosphère. Leurs vitesses sont supérieures à celle de la lumière dans l'air. Cela produit un rayonnement typique proche du violet, appelé lumière Cherenkov (lumière que l'on voit dans les piscines des réacteurs nucléaires). Si le rayon gamma a suffisamment d'énergie, la lumière Cherenkov atteint le sol.

L'interaction d'un rayon cosmique avec l'atmosphère produit des gerbes similaires mais quelque peu différentes puisque les mécanismes de création de particules ne sont pas exactement tous les mêmes. La figure 1.13 montre les différentes formes que peuvent avoir les gerbes électromagnétiques dans l'atmosphère. Un rayon gamma produit une gerbe de particules plus collimatée que celle produite par un proton. Cette particularité permet de différentier un rayon gamma (considéré comme du signal) d'un rayon cosmique chargé (considéré comme du bruit). En moyenne, une gerbe de particules sur mille est produite par un rayon gamma. Les méthodes utilisées pour rejeter le bruit doivent donc être très efficace.

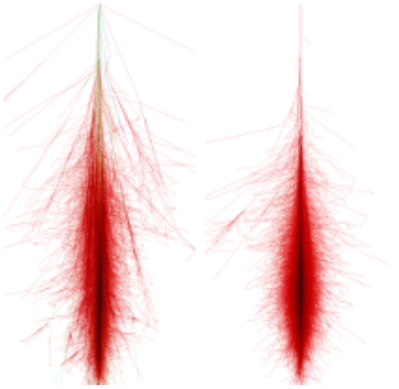


FIGURE 1.13 – À gauche : une gerbe de particules produite par un proton. À droite : une gerbe de particule produite par un photon (γ).

1.3.2 Détection du flash Cherenkov

De grands télescopes sont placés au sol pour détecter et enregistrer le flash de lumière Cherenkov. Leurs miroirs concentrent cette lumière sur une caméra très sensible. Cette caméra est composée de photomultiplicateurs (PM). Ils sont très sensibles et ont un taux de déclenchement très élevé. Ils peuvent détecter un seul photon et retrouver un signal stabilisé en quelques nanosecondes. Ils sont cylindriques et l'espace libre entre eux est réduit en les plaçant selon un motif hexagonal.

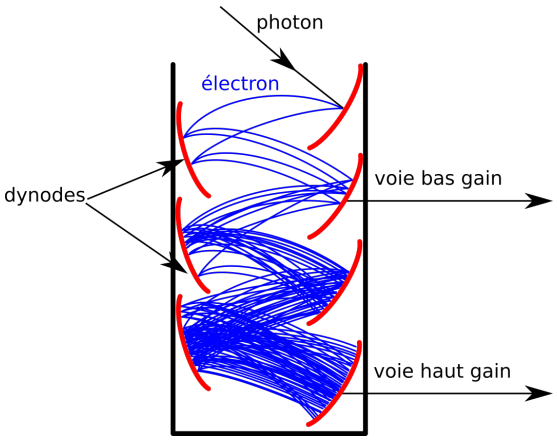


FIGURE 1.14 – Illustration du fonctionnement d'un photomultiplicateur.

Lorsqu'un photon entre dans un PM, il créé une cascade d'électrons en frappant la première plaque métallique appelée dynode (voir figure 1.14). Une haute tension amplifie la cascade d'électron vers une deuxième dynode et ainsi de suite jusqu'à la dernière. Enfin, un signal électrique est produit en collectant les électrons de la dernière dynode. Ce signal analogique, alors appelé signal haut gain, est digitalisé en 12, 16 ou 24 bits. Un autre signal, créé après seulement quelques dynode est aussi gardé, c'est le signal bas gain. Les signaux haut et bas gains sont digitalisés grâce à l'ADC (Analogic Digital Converter).

1.3.3 Détection stéréoscopique

Les données collectées par les télescopes sont les images des gerbes Cherenkov vues sous différents angles. Ces gerbes peuvent être approximées par des ellipses dans les images. Les images sont relativement bruitées du au fait que les PM sont très sensibles. L'analyse de données la plus simple consiste à nettoyer le bruit des images afin de faciliter l'extraction des paramètres géométriques des gerbes. La figure 1.15 montre la méthode de reconstruction stéréoscopique.

• **La reconstruction géométrique** détermine l'angle d'arrivée de la particule initiale grâce aux directions extraites des images de la gerbe.
• **La reconstruction en énergie** utilise ces paramètres géométriques (notamment le point d'impact de la gerbe au sol) pour calculer l'énergie de la particule initiale.
• **La discrimination** estime la probabilité que la particule initiale soit un rayon gamma ou non.

L'optimisation de cette analyse sera détaillée dans le chapitre 5.

Une autre méthode d'analyse utilise une comparaison d'images. Les images des télescopes à analyser sont comparées à des images préalablement simulées, où les caractéristiques

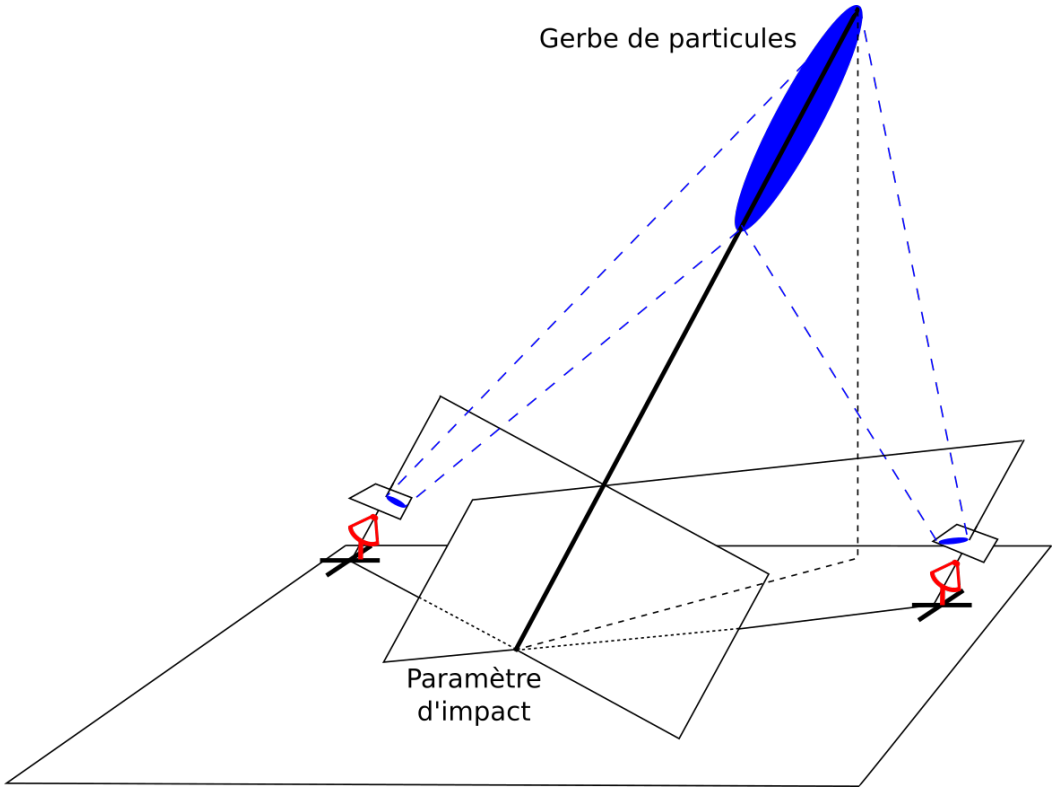


FIGURE 1.15 – Reconstruction stéréoscopique d'une gerbe de particules au sol.

du rayon gamma incident son fixées. La particule incidente aura les caractéristiques de la meilleur configuration (jeu d'images en stéréoscopie) et un calcul de vraisemblance indiquera sa ressemblance avec un rayon gamma. Le chapitre 6 traitera de l'optimisation de cette analyse.

1.4 Expériences actuelles

Quelques expériences de détection de rayons gamma au sol ont été construites depuis 2000 pour améliorer notre connaissance des rayons cosmiques :

- MAGIC : Major Atmospheric Gamma Imaging Cherenkov à La Palma (Espagne), observe depuis 2004 entre 30 GeV et 100 GeV avec deux télescopes.
- CANGAROO : Collaboration of Australia and Nippon for a GAmma Ray Observatory in the Outback, à Woomera, Australie. Le premier télescope fut construit en 1992. Le deuxième en 1999. Quatre télescopes furent opérationnels en mars 2004.
- VERITAS : Very Energetic Radiation Imaging Telescope Array System au sud de l'Arizona, USA, observe depuis 2004 avec un télescope, et avec quatre télescopes depuis 2007 entre 50 GeV et 50 TeV.
- H.E.S.S., High Energetic Stereoscopic System, la plus grande expérience de détection de rayons gamma au sol à ce jour avec 5 télescopes, décrite dans la section 1.4.1.

L'expérience H.E.S.S. sera d'avantage détaillée dans ce qui suit puisque ce travail de thèse, effectué dans le groupe H.E.S.S.-CTA au LAPP (Laboratoire d'Annecy de Physique des Particules), consiste à adapter la reconstruction de H.E.S.S. aux contraintes de calcul de CTA.

1.4.1 L'expérience H.E.S.S. (High Energetic Stereoscopic System)

L'expérience High Energetic Stereoscopic System, (H.E.S.S.), débuta à l'été 2002. Le nom H.E.S.S. fut donné pour rendre hommage à Victor Hess (voir figure 1.4), qui reçu le prix nobel de physique en 1936 pour la découverte du rayonnement cosmique.

La collaboration H.E.S.S. est composée de plus de 260 scientifiques répartis dans environ 40 institutions scientifiques de 13 pays : la Namibie, l'Afrique de sud, l'Allemagne, la France, l'Angleterre, l'Irlande, l'Autriche, les Pays-Bas, la Pologne, la Suède, l'Arménie, le Japon et l'Australie. À ce jour, cette expérience a découvert et étudiée plus de 80 sources astrophysiques de rayons gamma.

1.4.1.1 Un système de télescopes

Les télescopes de H.E.S.S. se situent en Namibie, sur le plateau de Komas, à une altitude de 1800m et à environ 100 km de la capitale Windhoek [24]. Il est composé de cinq télescopes Cherenkov. Les quatre petits, qui possèdent chacun un miroir de 12 mètres de diamètre, furent mis en service en 2003. Ils sont placés aux quatre coins d'un carré de 120 mètres de côté. Depuis 2012 un cinquième télescope équipé d'un miroir de 28 mètres de diamètre a été construit au centre de ce carré (voir figure 1.16)

FIGURE 1.16 – Le site de l'expérience H.E.S.S. en Namibie, sur le plateau de Komas à 100 km de la capitale Windhoek.

Le tableau 1.1 résume les caractéristiques des télescopes de H.E.S.S. Les quatre petits télescopes pèsent 60 tonnes chacun alors que le grand pèse 580 tonnes. Pointer les télescopes dans une direction est donc une tâche ardue. Les petits télescopes ont une vitesse de rotation de 100 deg/min et le grand de 200 deg/min.

Les miroirs des petits télescopes sont composés de 382 miroirs sphériques plus petits, appelés facettes, de 60 cm de diamètre ce qui représente une aire totale de 108 m^2 par télescope. Le miroir du grand télescope est composé de 875 facettes hexagonales de 90 cm de diamètre représentant une aire de 614 m^2.

Les miroirs des petits télescopes ont une distance focale de 15 mètres, le miroir du grand télescope à une distance focale de 36 m.

Caractéristique	Petits télescopes	Grand télescope
Masse	60 t	580 t
Vitesse de rotation	100 deg/min	200 deg/min
Nombre de facettes	382	875
Diamètre des facettes	60 cm	90 cm
Aire du miroir	108 m^2	614 m^2
Distance focale	15	36 m

TABLE 1.1 – Tableau des caractéristiques des télescopes de l'expérience H.E.S.S.

1.4.1.2 Les caméras

Les caméras de H.E.S.S. sont illustrées dans la figure 1.17. Elles sont fixées proches du plan focal de leur miroir respectif, soit à 15 ou 36 mètres de leur miroirs selon la taille du télescope. Une caméra composée de photomultiplicateurs est très massive car ces dispositifs

sont lourds, du fait de l'électronique, de l'alimentation, du système de refroidissement, etc. Les caméras des petits télescopes pèsent une tonne chacune et la caméra du grand télescope pèse environ 3 tonnes.

Comme nous l'avons dit dans la section 1.3, les photomultiplicateurs sont des dispositifs qui permettent de prendre beaucoup de données dans un court laps de temps. Les caméras des petits télescopes enregistrent 300 images de 960 pixels par seconde et celle du grand télescope prend 3 600 images de 2 048 pixels par seconde.

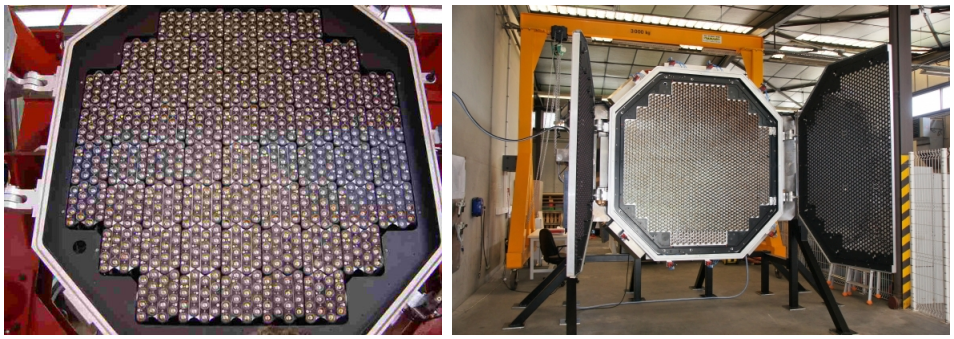


FIGURE 1.17 – À gauche : une caméra de H.E.S.S. I. À droite : la caméra H.E.S.S. II.

1.4.1.3 Flux, format et stockage de données

Le flux de données de H.E.S.S. est constant et d'environ 3 MO/s. Une observation dure 28 minutes et produit un fichier de 5.5 GO.

Le format de données de H.E.S.S. est basé sur la librairie Root [25] (utilisée pour les analyses de physique des particules). Les classes qui décrivent les télescopes héritent d'une classe plus générale qui décrit un télescope avec une liste de pointeurs de pixels. Une classe a été développée pour chaque type de données, les données brutes (images), les piédestaux, les gains et les pixels morts (ou broken pixels). Chacune de ces classes utilise son propre type de pixel qui hérite d'un pixel plus général. Toutes ces classes ont leurs propres configurations qui contiennent la position des pixels, leur numéro d'identification ainsi que les données enregistrées.

Le volume de données acquis par H.E.S.S. est important (plusieurs péta-octets par an depuis 2003). Ces données sont stockées sur bandes magnétiques au centre de calcul CCALI/CNRS/IN2P3 [26] à Lyon et leurs copies sont stockées en Allemagne au Max-Plank-Institute für Kernphysik (MPIK).

Dans cette expérience, les données sont stockées sur bandes et rapatriées par avion aux centres de calcul. En conséquences, elles sont systématiquement analysées deux mois après la prise de données.

1.4.1.4 Analyse et résultats

L'analyse de données de H.E.S.S. traite en plus d'une dizaine d'heures pour la prise de données d'une seule observation (28 minutes représentant 5.5 GO). Son but est d'utiliser les

différentes images d'une même gerbe pour déterminer la direction de la particule incidente, son énergie et son type. Seuls les rayons gamma donnent de l'information sur la position des sources de rayons cosmiques (voir section 1.3). Déterminer le type de la particule initiale est donc indispensable pour ne garder que les rayons gamma.

Les résultats de ces analyses donnent une liste de photons comportant une direction, une énergie et une probabilité d'être un rayon gamma (que l'on appelle aussi qualité). Cette liste de photons est alors utilisée pour produire des cartes du ciel (voir figures 1.18 et 1.19), des spectres en énergie, des courbes de lumières et autres graphiques.

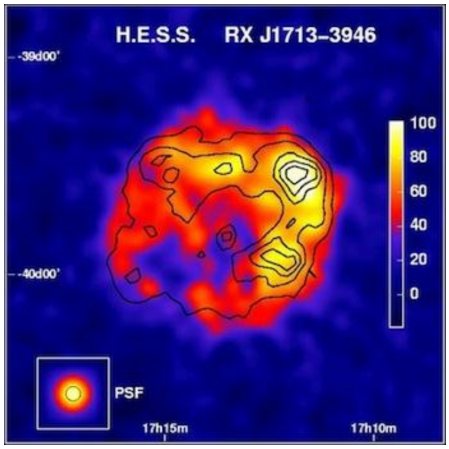


FIGURE 1.18 – La vestige de supernova RXJ1713-3946 vue par H.E.S.S. [27].

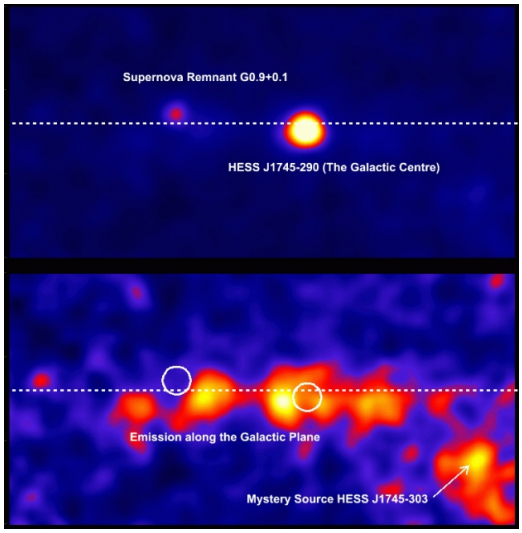


FIGURE 1.19 – Le centre galactique vu par H.E.S.S. [28].

1.5 Infrastructures de calcul

Pour les physiciens, les ressources de calcul sont multiples. La ressource de calcul la plus courante est la grille du CERN. Cette grille était destinée à l'analyse des données du LHC (Large Hadron Colider) et fut nommée WLCG (Worldwide LHC Computing Grid). Le LHC est le plus grand collisionneur de particule au monde, il fut utilisé dans la découverte du boson de Higgs en 2012. De nos jours, la grille est composée de millier d'ordinateurs reliés entre eux.

La grille (WLCG) est composée de quatre niveaux ou tiers :

— Le niveau 0 est le centre de calcul du CERN, il représente moins de 20% de la capacité de calcul totale de la grille.
— Le niveau 1 est composé de 13 grands centres de calcul (suffisamment grands pour stoker les données du LHC).
— Le niveau 2 rassemble des universités et d'autres instituts scientifiques qui peuvent stocker suffisamment de données et fournir une puissance de calcul adéquate pour des tâches d'analyses spécifiques.
— Le niveau 3 est fourni par les ordinateurs individuels des scientifiques.

Beaucoup d'expériences comme ATLAS, ALICE, LHCb (les trois expériences principales du LHC), H.E.S.S. ou CTA utilisent la grille pour procéder à de l'analyse de données ou de la simulation.

En France, l'*Institut National de Physique Nucléaire et de Physique des Particules* (IN2P3) utilise également un centre de calcul national : le CCALI, à Lyon. À ce jour, le CCALI contient 312 000 coeurs et 48 serveurs connectés avec un réseau local de 1 Gbits/s et une connexion sur l'extérieur à $20 - 30$ Gbits/s ce qui en fait un centre de calcul de niveau 2 pour la grille.

À Annecy, le *Laboratoire d'Annecy de Physique des Particules* (LAPP), a depuis 2007 son propre centre de calcul : le Mésocentre de calcul et de Stockage (MUST). Il fournit des ressources de calcul aux physiciens du LAPP et du LAPTH (*Laboratoire d'Annecy de Physique THéorique*). C'est un centre de calcul de niveau 2 du WLCG. Actuellement, 1.4 PO de stockage sont disponible pour ATLAS et 120 TO pour CTA.

Une sous-partie de MUST permet de fournir des architectures permettant le calcul hautes performances (HPC, voir section 2.4) avec 16 serveurs connectés *via* un réseau l'Infiniband QDR (40 Gbits/s). Deux noeuds sont équipés de GPU NVIDIA K80 et de nouveaux GPU P100 seront bientôt ajoutés.

Cette organisation multi-niveaux de ressources calcul offre une grande flexibilité. Tous ces centres de calcul sont connectés entre eux et échangent des données sur un réseau rapide. Donc, un calcul lancé sur MUST peut utiliser des données stockées sur la Grille avec un temps de transfert raisonnable. De plus, un calcul effectué sur un centre de calcul plus petit, comme MUST, sera exécuté plus rapidement contrairement au CCALI ou sur la grille du CERN.

Notre groupe utilise principalement MUST pour effectuer des tests les optimisations proposées de l'analyse de données. Le fait d'utiliser un centre de calcul plus petit facilite grandement l'exploration.

Les centres de calcul plus grands, tel que le CCALI, sont plutôt utilisés pour faire des productions ou de très grandes quantités de données sont nécessaires.

Cependant, la complexité des calculs (ou jobs) est limitée :

— Seulement, 2 GO de mémoire sont disponibles par thread.
— Un job ne peut avoir au maximum que 8 threads (il est possible de lancer des jobs sur 16 threads mais le temps d'attente peut être infini).
— Les jobs ne peuvent pas communiquer entre eux.
— Les ordinateurs utilisés ont généralement plus de quinze ans. Ils n'ont donc pas des extensions CPU courantes. Lorsque de nouveaux ordinateurs sont ajoutés dans un centre de calcul, ils sont installés avec une vieille version du système d'exploitation scientific LINUX, sur laquelle des compilateurs ne permettent pas la vectorisation. Toutes ces machines sont renouvelées régulièrement, ce qui rend cette infrastructure particulièrement hétérogènes.

Malgré tout, les jobs peuvent être lancés par centaines sur ces centres de calcul. Mais puisque qu'ils ne peuvent pas communiquer, le calcul doit être trivialement parallélisable. Ces architectures sont bien différentes de celles utilisées en informatique hautes performances (voir section 2.4).

1.6 Limites de la méthode

Comme nous l'avons vu dans les sections précédentes, l'expérience H.E.S.S. a découvert et étudiée plus de 80 sources de rayons cosmiques de tout type. À ce jour, ces sources sont étudiées une par une. Malgré l'amélioration des techniques de détection et des analyses précédentes, le catalogue actuel de sources ne permet pas de faire des études de population car il ne contient pas assez de sources d'un même type pour que l'étude soit suffisamment pertinente. Une telle étude ne serait possible qu'en améliorant la sensibilité de l'expérience. Cette amélioration permettrait également d'étudier plus finement les sources déjà connues.

Cela nécessite, une augmentation drastique des capteurs (nombre de télescopes, tailles des miroirs, PM), ainsi qu'un accroissement de leur sensibilité. Désormais, les nouvelles technologies le permettent.

L'astrophysique multi longueurs d'ondes et multi messagers n'est possible qu'avec la coopération des expériences entre elles et la mise en place d'un système d'alertes. Si leurs données sont publiques elles deviendront de ce fait des observatoires. Cela fut le cas le 14 août 2017, lorsque l'expérience LIGO-VIRGO lança une alerte, concernant la détection d'une coalescence de deux trous noirs, vers toutes les expériences d'astronomie gamma [29]. L'astronomie multi-messagers est née.

Réagir à une alerte implique d'être capable d'observer n'importe quelle portion du ciel en quelques secondes tout en analysant les données en temps réel. Cette contrainte implique de réduire le temps d'analyse de quelques heures (habituellement) à quelques secondes !

L'expérience Cherenkov Telescope Array (CTA) sera le plus grand observatoire d'astronomie gamma au sol à relever ces défis.

L'essentiel du chapitre 1 –

Les sources de rayons cosmiques ne peuvent être étudiées que via les rayons gamma qui ne représentent que 0.1% des rayons cosmiques arrivant sur Terre. L'astronomie gamma au sol étudie des sources galactiques et extra-galactiques. Elle diffère de l'astronomie conventionnelle car les photons étudiés proviennent de gerbes électromagnétiques créées par une particule initiale (rayon cosmique ou rayon gamma) qui interagit avec l'atmosphère. De ce fait, la discrimination est une tâche très importante qui permet d'augmenter le rapport signal sur bruit des données à analyser.

Plusieurs expériences sont en cours depuis 2000, comme H.E.S.S., MAGIC ou VERITAS. Elles ont permis des avancées dans ce domaine mais beaucoup reste à accomplir. Améliorer la sensibilité des instruments permettrait d'effectuer des études de population et d'agrandir le catalogue de sources à étudier.
La prochaine échelle de ces expériences est définie par le projet Cherenkov Telescope Array, CTA, qui repoussera les limites de l'astronomie gamma au sol.

Nous vivons désormais dans un monde de coopération entre les différentes expériences d'astrophysique. L'astrophysique multi-messagers a fait son premier pas le 14 août 2017 lorsque 70 expériences ont suivis l'évolution du même objet astrophysique en même temps. Toutes les échelles changent. Il ne s'agit plus d'analyser des données quelques mois après l'observation, mais bien de les traiter dans les quelques secondes qui suivent une mesure. Les capteurs sont plus nombreux et plus précis. Ils produisent un flux inédit de données qu'il faudra tout de même analyser en temps réel.
Les objectifs des analyses en temps réel ont eux aussi évolués. Il ne s'agit plus uniquement de contrôler que la prise de données se déroule correctement mais de pouvoir faire de la science avec.

Toutes ces évolutions appellent des optimisation, tant physique que informatique. Nous suivrons dans ce manuscrit comment les techniques d'optimisation du calcul hautes performances peuvent répondre à ces nouveaux défis.

Chapitre 2

Le projet CTA (Cherenkov Telescope Array)

Sujet du chapitre 2 –
Ce chapitre présente le projet CTA, nouvelle échelle de l'astronomie gamma au sol et sujet principal de ce manuscrit. Après la présentation des objectifs, de la structure et des défis de cet observatoire, le calcul haute performance y est introduit ainsi que les possibilités qu'il offre pour relever les défis liés au traitement des données.

Sommaire

Depuis plus de 40 ans, des expériences ont étudié les rayons cosmiques. Chaque génération a apporté de nouvelle méthodes de détection ou d'analyse. Les expériences actuelles d'astronomie gamma au sol ont montré qu'utiliser plusieurs télescopes simultanément, en stéréoscopie, améliorait la résolution de l'instrument. Jusqu'à aujourd'hui, H.E.S.S. est la plus grande expérience d'astronomie gamma au sol avec cinq télescopes (voir section 1.4.1).

Le projet CTA est un pas vers une nouvelle échelle. Les télescopes seront bien plus nombreux qu'auparavant, les caméras seront plus sensibles et auront plus de pixels. Tout ces changements lui permettent d'atteindre de nouveaux objectifs.

2.1 Les objectifs

L'expérience CTA sera consacrée à l'étude des rayons cosmiques par le biais des rayons gamma. Trois thèmes majeurs de recherche seront approfondis.

Le premier axe de recherche sera orienté sur les rayons cosmiques. Les positions de leurs sources et leurs mécanismes d'accélération à différentes énergies, leur rôle dans la formation des étoiles et l'évolution des galaxies.

Le second axe étudiera les environnements extrêmes, tels que les étoiles à neutron et les trous noirs, afin de comprendre les processus physiques en jeu. Ces environnements sont propices à l'étude de phénomènes relativistes comme les jets de matière, les vents ou les explosions. Les vides spatiaux extrêmes seront également observés pour quantifier l'intensité des radiations et des champs magnétiques dans ces environnements.

Le dernier axe tentera d'explorer les frontières de la physique connue. La nature de la matière noire ou l'effet de la gravitation quantique sur les photons pourraient être élucidés.

CTA met en commun les connaissances des expériences précédentes, notamment H.E.S.S. et MAGIC. Son objectif premier est de gagner un ordre de grandeur en sensibilité. Des gerbes de particules dix fois moins intenses seront détectées. L'intervalle d'énergie en détection sera également augmenté. La qualité de la reconstruction de l'analyse de données devra être améliorée pour être cohérente avec la sensibilité accrue des télescopes. Cela permettra d'effectuer des études de populations. Ces études sont très limitées pour le moment car le catalogue des sources est relativement restreint.

Une analyse de données en temps réel sera développée pour vérifier le bon déroulement des observations et également pour lancer des alertes aux observatoires du monde entier. Elle devra donc être suffisamment précise et extrêmement rapide.

Les télescopes de l'observatoire CTA devront également être capable de répondre à une alerte en repointant dans un délais relativement court.

2.2 L'observatoire

L'expérience précédente H.E.S.S. regroupe 260 physiciens contrairement à la collaboration CTA qui rassemble 1 420 personnes, avec notamment des ingénieurs, des électroniciens, des physiciens, des informaticiens, etc.

La conception du projet CTA commença en 2005. Le premier prototype de télescope est actuellement en cours de construction (voir figure 2.1) et sera achevé au troisième trimestre 2018. Le projet passera alors en phase de pré-production de 2019 à 2021, puis sera mis en production.

FIGURE 2.1 – Prototype de grand télescope de CTA en construction à La Palma (Espagne).

2.2.1 Les sites

Le système solaire est légèrement décalé vers le haut par rapport au plan de rotation de notre galaxie (La Voie Lactée). Cela expose d'avantage l'hémisphère sud de la Terre au rayonnements provenant de la galaxie. Cela est idéal pour observer les sources galactiques. À l'inverse, un site d'observation basé dans l'hémisphère nord sera moins soumis au rayonnements en provenance de notre galaxie et sera donc plutôt consacré à l'étude des sources extra-galactiques.

Il est tout de même possible d'étudier des sources extra-galactiques avec un site dans l'hémisphère sud. Mais elles ne seront pas les mêmes et les sources présentes dans notre galaxie rendent cette tâche plus difficile car elles émettent des signaux relativement plus fort dus à leur proximité.

L'expérience CTA sera implantée dans les deux hémisphères de la planète pour permettre de diversifier les observations.

CTA sera composé de deux sites :

— Le site sud, au Chili, dans le désert du Paranal, proche du VLT (Very Large Telescope), sera composée de 100 télescopes. Il sera utilisé pour étudier les sources galactiques.
— Le site nord, en Espagne, à La Palma, une des iles Canaries, sera constitué de 19 télescopes. Ce site étudiera principalement les sources extra-galactiques.

2.2.2 Les telescopes

Les sites seront composés de six types de télescopes de quatre tailles différentes. Les grands télescopes seront placés au centre des sites, les moyens les entoureront, puis les petits couvriront une vaste surface. Cette hétérogénéité permettra d'élargir le spectre en énergie détectable par CTA tout en couvrant une plus grande surface au sol. Les plus grands seront légèrement plus petits que le grand télescope de H.E.S.S. Les moyens seront comparable aux petits télescopes de H.E.S.S. Les petits ne mesureront que quelques mètres mais seront en beaucoup plus grand nombre de sorte à avoir une surface de collection plus grande.

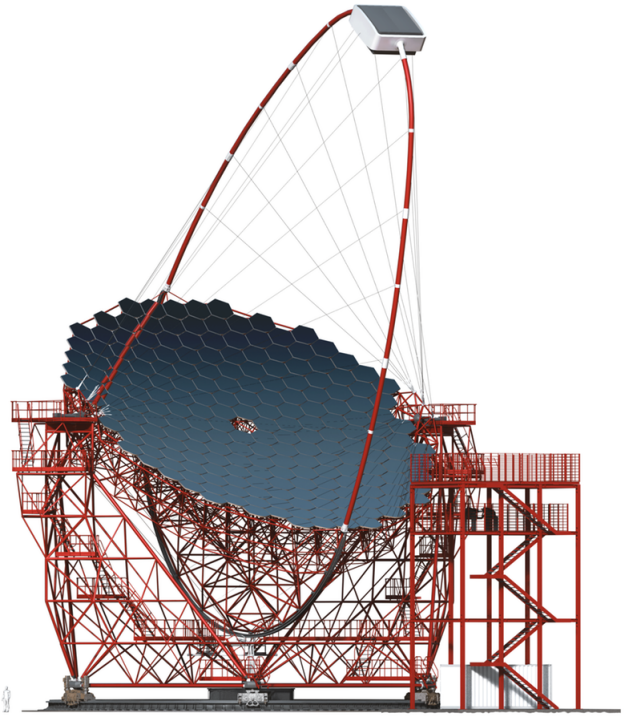
Les caméras installées sur ces télescopes seront bien plus rapides que celles des précédentes expériences (voir tableau 2.1).

Télescope	Type de Caméra	Nombre de pixels	Nombre de films par seconde	Images par film
LST	LST-CAM	1 855	15 000	30
MST	NECTAR-CAM	1 855	9 000	60
MST	FLASH-CAM	1 764	9 000	22
SCT	SCT-CAM	11 328	2 500	60
SST	ASTRI-CAM	2 368	600	1
SST	DC-CAM	1 300	600	25
SST-1M	GCT-CAM	2 048	600	80

TABLE 2.1 – Les différentes caractéristiques des caméra de CTA.

2.2.2.1 Les grands télescopes (Large Size Telescopes, LST)

Les grands télescopes (Large Size Telescopes, LST) sont conçus par huit pays en collaboration : le Brésil, la France, l'Allemagne, l'Inde, l'Italie, l'Espagne, la Suisse et la Suède. Les LST (voir figure 2.2) sont très sensibles aux basses énergies, entre 20 et 200 GeV. Ils ont un miroir segmenté de 23 mètres de diamètre, d'une surface de 400 m^2 et d'une distance focale de 28 m qui concentre la lumière Cherenkov dans une caméra. Un télescope LST mesurera 45 m de hauteur et pèsera environ 100 tonnes.



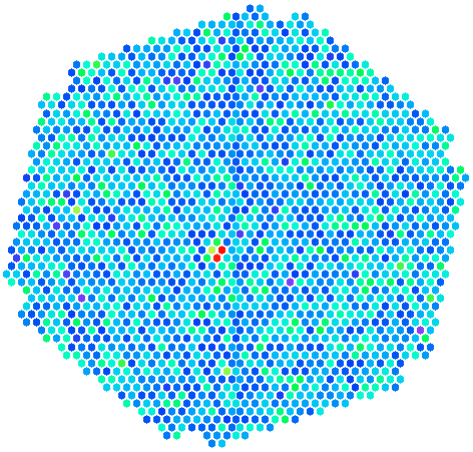
Poids	100 t
⌀ miroir	23 m
Surface du miroir	400 m^2
Distance focale	28 m
Temps repositionnement	20 s

FIGURE 2.2 – Grand télescope de CTA (Large Size Telescope, LST).

Ils pourront se repositionner en moins de 20 secondes. Cette capacité de repositionnement rapide couplé à la sensibilité permettra à CTA d'étudier des phénomènes courts et à faible magnitude comme : les transients galactiques, les noyaux actifs de galaxie à fort décalage vers le rouge et les sursauts gamma.

Les LST-CAM sont les caméras des grands télescopes (voir figure 2.3). Elles ont un champ de vue d'environ 4.3 degrés. Leur conception les rend compactes et légères (moins de deux tonnes). Ces caméras sont constituées de 1 855 photomultiplicateurs divisé en 265 tiroirs qui les rendent simple d'accès et donc facilite leur maintenance. Ils sont équipés d'un concentrateur optique (ou cône de Winston), leur pic d'efficacité quantique atteint les 40%.

Les taux de déclenchement de ces caméras ont été améliorés en prenant en compte la topologie des gerbes de particules et l'évolution temporelle du flash Cherenkov dans la caméra. Un algorithme fut développé spécialement pour détecter les flashs de lumière Cherenkov provenant de petites gerbes. Les caméras des LST seront également interconnectées entre



Nb pixels	1855
Nb films/s	15000
Nb images/film	30

FIGURE 2.3 – Caméra des grands télescopes de CTA (LST-CAM).

elles. Ce dispositif de gestion des coïncidences en ligne propre aux LST permettra de réduire le taux de coïncidences fortuites d'un facteur 100.

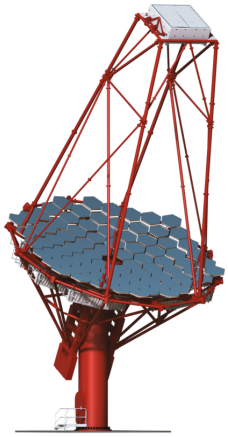
Elles peuvent enregistrer jusqu'à 15 000 films de 30 images par seconde. Les données des pixels sont encodés en 32-bits. Elles produisent un flux de données de 3.339 GO/s par caméra. Cette caméra est fabriquée au *Paul Scherrer Institute* en Suisse et est utilisée par quelques expériences, dont les télescopes de l'expérience MAGIC.

Un prototype de LST est actuellement en cours de construction à *l'Observatorio del Roque de los Muchachos* à La Palma (Espagne). Il sera normalement terminé à l'été 2018 et sera mis en service à la moitié de l'année 2019.

2.2.2.2 Les télescopes moyens (Medium Size Telescopes, MST)

Les télescopes moyens (Medium Size Telescopes, MST) rassemblent six pays : le Brésil, la France, l'Allemagne, la Pologne, la Suisse et les Etats-Unis. Les MST (voir figure 2.4) font le pont entre les hautes et les basses énergies, entre 100 GeV et 5 TeV. Ces télescopes seront capables de se repositionner en 90s (ce qui est 4.5 fois plus lent que les LST).

Ils ont un miroir segmenté de 12 mètres de diamètre composé de 90 miroirs hexagonaux accusant une surface totale de 88 m^2 et une distance focale de 16 m.

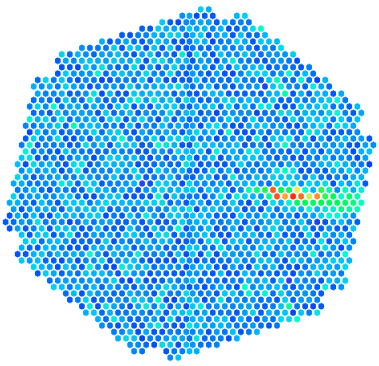


Poids	82 t
⌀ miroir	12 m
Surface du miroir	88 m^2
Distance focale	16 m
Temps repositionnement	90 s

FIGURE 2.4 – Moyen télescope de CTA (Medium Size Telescope, MST).

Deux types de caméras sont utilisés sur ces télescopes :

• Les NectarCAM sont très similaires au LST-CAM. Elles sont composées de 1 855 photomultiplicateurs, et ont un angle de vue de 8 degrés (figure 2.5 gauche). Elles enregistrent jusqu'à 9 000 films de 60 images par seconde digitalisées en 32-bits, ce qui représente un flux de données de 12 GO/s par caméra.

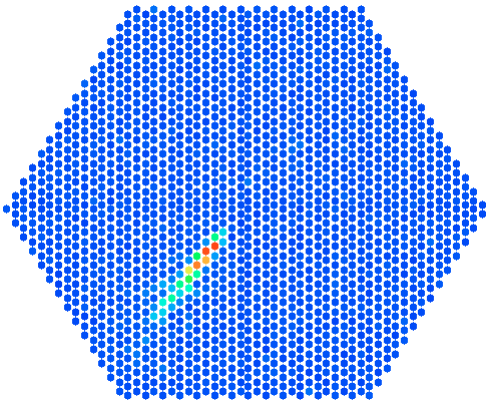


Nb pixels	1855
Nb films/s	7000
Nb images/film	60

FIGURE 2.5 – Une caméra NECTAR-CAM des moyens télescopes de CTA.

• Les FlashCAM, ont 1 764 photomultiplicateurs, et un angle de vue de 8 degrés (figure 2.6 droite). Elles enregistrent jusqu'à 9 000 films de 22 images par seconde, digitalisées en 12-bits et produisent un flux de données de 12 GO/s par caméra.

Au total, 40 MST seront construit sur les deux sites, 25 pour le site sud et 15 pour le site nord.



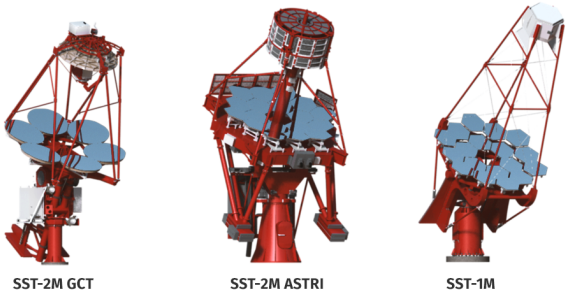
Nb pixels	1764
Nb films/s	6000
Nb images/film	22

FIGURE 2.6 – Une caméra FLASH-CAM des moyens télescopes de CTA.

2.2.2.3 Les petits télescopes (Small Size Telescopes, SST)

Les petits télescopes (Small Size Telescopes, SST) ne seront présent qu'au site sud (Paranal). Avec 70 télescopes, les SST surpasseront tous les autres télescopes en nombre. Ils s'étaleront sur quelques kilomètres carrés autour des autres télescopes de sorte à détecter de grandes gerbes, créées par des rayons gamma de hautes énergies. Les SST sont sensibles de quelques TeV à 300 TeV. Malgré leurs petites tailles, ils se repositionneront en pas moins de 60 secondes, soit trois fois plus lentement que les LST.

Il y aura trois types de petits télescopes (voir figure 2.7) :

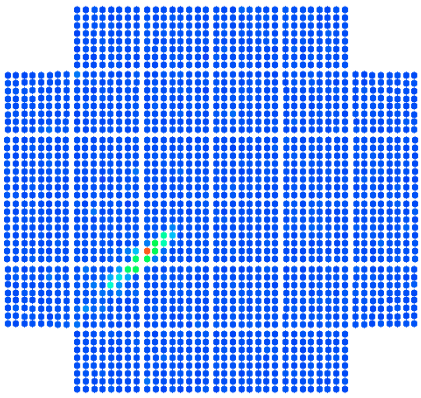


Type	GCT	ASTRI	SST-1M
Poids	11 t	19 t	8.6 t
⌀ miroir principal	4 m	4.3 m	4 m
⌀ miroir secondaire	2 m	2.3 m	-
Surface du miroir	8.9 m^2		7.5 m^2
Distance focale	5 m		
Temps repositionnement	60 s	60 s	60 s

FIGURE 2.7 – Petits télescopes, SST.

• Les GCT seront fournis par l'Australie, la France, l'Allemagne, le Japon, les Pays-Bas, l'Angleterre et les Etats-Unis. Ils ont une structure Schwarzschild-Couder composée de deux

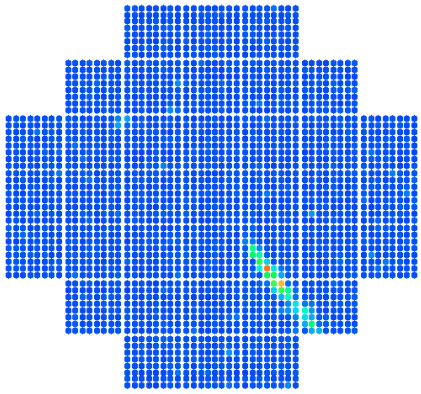
miroirs. Le miroir principal a un diamètre de 4.0 m et le second de 2.0 m. Cela représente une surface effective de 8.9 m^2 avec la prise en compte de l'occultation. Les GCT-CAM ont 2 048 photomultiplicateurs et un champ de vue de 9.2 degrés (voir figure 2.8). Elles enregistrent 400 films de 80 images par seconde digitalisées en 16-bits et produisent un flux de données de 2.0 GO/s par caméra.



Nb pixels	2048
Nb films/s	400
Nb images/film	80

FIGURE 2.8 – Caméra GCT des petits télescopes.

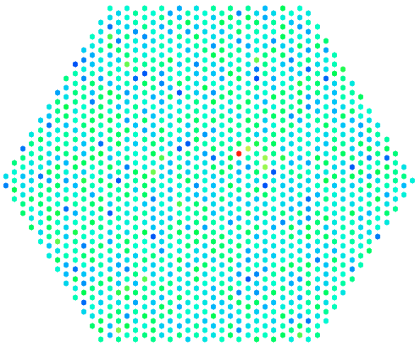
• Les ASTRI sont construites par le Brésil, l'Italie et l'Afrique du Sud. Ils ont aussi une structure Schwarzschild-Couder composée de deux miroirs, de 4.3 m et 2.3 m respectivement. Les ASTRI-CAM sont composées de 2 368 photomultiplicateurs en silicone (SiPMs) et ont un champ de vue de 9.6 degrés (voir figure 2.9). Elles ne prennent que 300 images par seconde digitalisées en 16-bits et produisent un flux de données de 2.0 GO/s par caméra.



Nb pixels	2368
Nb films/s	300
Nb images/film	1

FIGURE 2.9 – Caméra ASTRI-CAM des petits télescopes.

• Les SST-1M sont développées par la Suisse et la Pologne. Ils n'ont qu'un seul miroir de 4.0 m de diamètre d'une surface de 7.5 m^2. Leur conception est proche des FLASH-CAM mais l'utilisation de photomultiplicateurs silicone permet de réduire leur taille. Elles sont constituées de 1 300 photomultiplicateurs et ont un champ de vue de 9.1 degrés (voir figure 2.10). Elles enregistrent 600 films de 25 images par seconde digitalisées en 16-bits et produisent un flux de données de 3.2 GO/s.



Nb pixels	1300
Nb films/s	600
Nb images/film	25

FIGURE 2.10 – Caméra SST-1M des petits télescopes.

2.2.2.4 Les télescopes Schwarzschild-Couder (SCT)

Le télescope Schwarzschild-Couder (Schwarzschild-Couder Telescope, SCT) est une alternative au MST (voir figure 2.11). Sa conception en double miroirs est supposée plus efficace qu'un MST classique. Généralement un télescope Cherenkov classique est équipé d'un miroir segmenté composé de miroirs sphériques fixés sur une structure parabolique. Cette configuration créée des déformations appelées aberrations géométriques. La construction en double miroirs résout ce problème. Cela permet d'avoir un télescope plus compact (la distance focale est de seulement 5.58 m contre 16 m pour un MST), avec une structure plus simple à installer et à entretenir.

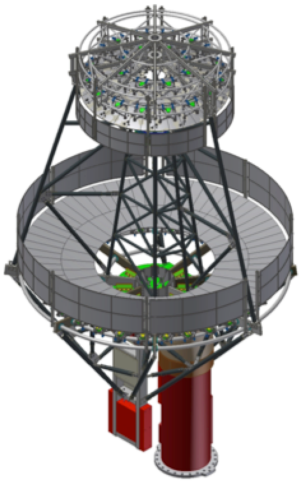
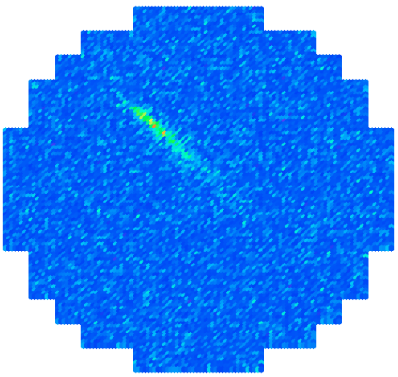


FIGURE 2.11 – Telescope Schwarzschild-Couder, SCT.

Des instituts américains sont en charge de leur conception, mais cette solution n'a pas encore été validée par le consortium de CTA pour le moment. Les SCT ne seront pas présents dans la première version du site, mais pourront être ajoutés lors de son évolution.

Les caméras des SCT surpassent les autres en nombre de pixels (voir figure 2.12). Elles enregistrent 2 500 films de 60 images de 11 328 pixels par seconde.



Nb pixels	11328
Nb films/s	2500
Nb images/film	60

FIGURE 2.12 – Illustration de la caméra des SCT SCT-CAM.

2.2.3 Le traitement des données

Le traitement des données de CTA sera multiple. Trois analyses différentes se succéderons pour traiter ces données, chacune a son propre objectif.

La première analyse, sera une analyse en temps réel (**on-line**). Elle devra analyser le flux de données en provenance des télescopes (27 GO/s pour le site sud et 17 GO/s pour le site nord, en prenant en compte les taux de déclenchement). Un événement sera analysé en moins de trente secondes après sa détection. Contrairement aux analyses de données en temps réel des expériences comme H.E.S.S. ou MAGIC, elle ne permettra pas uniquement de vérifier le bon déroulement de la prise de données. Cette analyse devra être suffisamment précise pour lancer des alertes pertinentes aux autres observatoires le cas échéant.

La deuxième analyse, sera une analyse sur site (**on-site**). Les données seront analysées une seconde fois, le jour suivant la prise de données. Cette analyse détectera des phénomènes intéressants encore plus précisément que l'analyse temps réel. Cela permettra de ne laisser passer aucune activité inhabituelle et jouera un rôle important dans la réduction du volume de données pour les transférer hors des sites (une ligne à 1 GO/s est disponible pour chacun des sites). Cette réduction se basera sur de la compression sans perte et sur une compression avec perte qui pourra exploiter le fait que 1 événement sur 1000 est produit par un rayon gamma en moyenne et que tous les pixels d'une image ne sont pas tous indispensables. Pour l'instant, aucune méthode n'a été choisi par le consortium de CTA. Néanmoins, elle devra être très performante en terme de temps de calcul puisqu'elle traitera les données de CTA au milieu du désert ou au sommet d'une montagne, ce qui n'est pas un endroit aisé pour monter un centre de calcul.

Ces deux analyses ont de fortes contraintes tant physique que informatique. Notamment dues au fait que les sites sont très isolés et que les ressources y sont limitées.

La troisième analyse de données (**off-site**) sera effectuée dans les centres de calculs qui stockeront les données de CTA ou sur la Grille [1]. Ils ne sont pas encore définis, mais seront au

1. Grille de calcul du CERN.

nombre de quatre et stockeront chacun la moitié des données dans le but d'éviter toute perte grâce à cette redondance. Cette analyse sera encore plus précise que les deux précédentes.

Le consortium de CTA a décidé que le langage principal de cette analyse, **ctapipe** [30], serait le Python [2] et que les parties les plus critiques seraient développées en C++.

Les informations des parties précédentes provienent du site officiel du projet CTA [31].

2.3 Les défis

Le projet CTA est une somme de défis tous plus ambitieux les uns que les autres, autant physiques que techniques, mécaniques ou informatiques.

2.3.1 Les défis généraux

Le premier est de gagner un ordre de grandeur en sensibilité et d'augmenter l'intervalle d'énergie de détection de l'expérience. Cela implique également une amélioration de la résolution angulaire (erreur sur les directions des rayons gamma reconstruits), de la résolution en énergie (erreur entre l'énergie reconstruite et l'énergie du rayon gamma incident) et de la discrimination (séparation des rayons gamma des autres rayons cosmiques qui sont considérés comme du bruit). Cette amélioration facilitera la détection de sources, ce qui permettra d'effectuer des études de populations par opposition à l'étude de sources au cas par cas qui est pratiquée pour le moment. Elle permettra aussi d'étudier des phénomènes transitoires et de créer des alertes vers les autres observatoires.

La notion d'alerte fonctionne dans des deux sens. CTA devra être capable de répondre aux alertes des autres observatoires. Cela implique que les télescopes devront être conçus pour repointer en moins de 20 secondes. Concevoir et construire des télescopes de 100 tonnes, équipés d'une caméra d'une tonne fixée à 28 mètres de leur axes de rotation et qui devront se repositionner rapidement représente un défi mécanique sans précédent.

Un grand nombre de télescopes, composés de caméras plus sensibles contenant plus de pixels produira une formidable quantité de données. De part son statut d'observatoire, les données de CTA devront être publiques. Les données devront donc être traitées le plus rapidement possible sur leurs sites respectifs puis compressées avant d'être diffusées.

2.3.2 Les défis de calcul

Les données seront analysées en temps réel. Ce qui signifie qu'un événement devra être analysé en moins de trente secondes après sa détection. Cette nouveauté permettra de vérifier au fur et à mesure de l'observation l'activité de la région du ciel visée. Elle permettra, le cas

2. Le Python est de plus en plus utilisé en physique des particules depuis quelques années et est très prisé des nouveaux physiciens. Il est apprécié pour son apparente simplicité et aux nombreuses contributions de ces utilisateurs (notamment des outils comme *IPython* ou *Jupyter*).

échéant, de prévenir les autres observatoires sur Terre que quelque chose d'intéressant se passe dans le ciel.

Les ressources de calcul sur site devront donc être particulièrement bien exploitées. De plus, la totalité des données stockées sera réanalysée tous les ans. C'est dans ce cadre que le calcul haute performance apporte une solution : À savoir adapter et optimiser une analyse performante du point de vue de la physique pour que les temps de calcul soient réalistes.

Nous suivrons l'optimisation des différentes analyses de données, leurs adaptations de l'expérience H.E.S.S. au projet CTA, tout au long de ce manuscrit.

2.4 La place du calcul haute performance

Le calcul haute performance (HPC, High Performance Computing) est un domaine de l'informatique dont le but est d'obtenir les meilleures performances de calcul sur une architecture donnée, typiquement atteindre le pic de performance de l'architecture utilisée. Obtenir de très bonnes vitesses de calcul implique d'atteindre le pic de performances le plus souvent possible pendant toute la durée du calcul. Cette discipline doit tirer parti de toutes les architectures disponibles : CPU, GPU, FPGA, multi-cores, many-cores et de toutes leurs combinaisons.

La difficulté majeure est de s'assurer que les algorithmes utilisés seront efficaces sur une architecture donnée. Pour ce faire, une excellente connaissance de l'architecture ciblée est nécessaire. Si un algorithme n'exploite pas suffisamment l'architecture, il est important de savoir si il peut être remplacé par un algorithme plus efficace, ou si des parties de cet algorithme peuvent être améliorées par des fonctions offrant de meilleures performances.

De nos jours, les données sont partout, chaque dispositif produit ses propres données. Le volume de données est gigantesque et augmente avec le temps. Des plateformes comme Youtube stockent chaque minute des centaines d'heures de vidéo supplémentaires.

Les expériences courantes sont dans ce cas et les nouvelles augmenteront cette production de données. L'idée générale est d'avoir de plus en plus de capteurs beaucoup plus sensibles que ceux de la génération précédente.

Cette nouvelle échelle impacte un grand nombre d'applications :

- Les expériences de physique (physique des particules, astrophysique, physique des plasmas, mécanique des fluides).
- Les prévisions météorologiques (modèles d'océans, d'atmosphère, d'ouragans ou de tornades).
- Les expériences de biologie (simulation de cellules, du cerveau, ou d'arbres).
- L'industrie (simulation de réservoirs sous-terrain de Total, simulation d'avion d'Airbus, etc).

La principale difficulté est que les centres de calcul nécessaires au traitement de ces données ne peuvent pas suivre ce rythme. L'approche classique qui consistait à augmenter le nombre de jobs pour analyser les données d'une expérience ne permet plus d'obtenir un

résultat dans un temps raisonnable car la seule conséquence de cette démarche est que les jobs attendent de plus en plus longtemps en queue avant d'être exécutés.

Il semblerait que ce problème n'aie pas de solution. A-t-on besoin de nouveaux ordinateurs ? d'ordinateurs spécifiques ? ou plus puissants ? La réponse est : non pour le moment.

En effet, l'utilisation moyenne du CPU de toutes les applications dans le monde est inférieure à 4% du pic de performance [32]. Donc, la puissance de calcul existe mais n'est pas utilisée. Les algorithmes utilisés couramment doivent être améliorés pour exploiter au mieux ces centres de calcul.

Une analyse plus rapide économise du temps de calcul et d'attente pour l'utilisateur mais le coût électrique est également diminué. De plus, 1 000 CPUs utilisés à 4% de leurs pics de performances ont la même puissance de calcul que 40 CPUs utilisés à 100%. Évidemment, le coût électrique ou celui nécessaire à la maintenance n'est pas le même pour 40 CPUs que pour 1 000. Au final, pour certaines expériences, les utilisateurs pourraient utilisés leur propres ordinateurs portables à la place d'un centre de calcul ce qui économiserai du temps de calcul pour de nouveaux utilisateurs potentiels.

Le traitement des données de CTA nécessitera une importante puissance de calcul. Les données stockées en Europe seront analysées sur la Grille ou dans des centres de calcul spécifiques. Ils ne sont pas encore définis, mais certains seront probablement adaptés dans ce but. Dans ce cas, la problématique consistera à analyser efficacement une augmentation linéaire de quelques Péta-octets de données par an. Depuis que de tels problèmes existent, le HPC évalue différentes solutions destinées à les résoudre. De puissants supercalculateurs ont été construits dans ce sens (voir figure 2.13).

La problématique du traitement d'une grande quantité de données implique une gestion efficace des entrées/sorties de sorte que l'application ne soit plus memory-band [3]. Dans le cas contraire, nous retombons dans la sous exploitation des CPUs évoqué précédemment. Des algorithmes, comme Map-reduce [33] utilisés dans HADOOP [34] ou SPARK [35], ont été développés dans ce but et testés sur des supercalculateurs.

Les supercalculateurs les plus puissants du monde sont catalogués dans un classement mondial appelé TOP-500 [36] (voir tableau 2.2). Ils permettent de tester des algorithmes dans des conditions extrêmes (grand nombre de noeuds de calcul, d'interconnections, de communications, etc). Ces évaluations expérimentent des comportements jusqu'à lors théoriques qui fournissent de précieuses informations pour concevoir les futurs supercalculateurs.

Dans le cas d'analyse **off-site**, la consommation électrique n'est pas une préoccupation décisive puisque les centres de traitement des données de CTA seront choisis très probablement à proximité de grandes villes. En revanche, l'analyse des données de CTA concernant les analyses en temps réel (**on-line**) et sur site (**on-site**) devront être performantes en terme de calcul mais également en terme de consommation électrique puisqu'elles seront utilisées au milieu du désert (Paranal, Chili) ou au sommet d'une montagne (La Palma, Espagne). Sur ces sites, les ressources de calcul disponibles seront en accord avec les limitations du réseau électrique. Depuis quelques années, une nouvelle branche dérivée du HPC, appelée GREEN-

3. Désigne un programme dont la vitesse d'exécution est limitée par les entrées/sorties.

FIGURE 2.13 – En haut à gauche : le supercalculateur Sunway TaihuLight. En haut à droite : le supercalculateur Tianhe 2. En bas, le supercalculateur Titan 2.

Rang	Pays	Système	Coeurs	Puissance (kW)
1	Chine	Sunway TaihuLight	10,649,600	15,371
2	Chine	Tianhe-2 (MilkyWay-2)	3,120,000	17,808
3	USA	Titan	560,640	8,209
4	USA	Sequoia	1,572,864	7,890
5	USA	Cori	622,336	3,939
6	Japon	Oakforest-PACS	556,104	2,719
7	Japon	K computer	705,024	12,660
8	Suisse	Piz Daint	206,720	1,312
9	USA	Mira	786,432	3,945
10	USA	Trinity	301,056	4,233

TABLE 2.2 – Le classement TOP-500 du 14 novenbre 2016 des plus puissants supercalculateurs dans le monde.

HPC, a pour objectif d'obtenir les meilleures performances de calcul mais en prenant en compte la consommation électrique. Ils sont classés dans le GREEN-500 [36] (voir tableau 2.3), et évaluent des solutions techniques que l'on retrouve dans les systèmes embarqués (très utilisés en biologie) et tentent de répondre à une problématique de taille posée par les supercalculateurs classiques : comment atteindre l'Exa-scale (10^{18} opérations flottantes par seconde) avec une consommation électrique raisonnable ? Un grand nombre d'architectures

hybrides (CPU+GPU, CPU+FPGA voir même CPU+GPU+FPGA) sont développées dans ce sens. Les retours sur expériences de ces architectures seront des plus utiles pour déterminer la composition des centres de calcul qui seront utilisés sur les sites.

Rang	MFLOPS/W	Site	Puissance Totale(kW)
1	9462.1	NVIDIA Corporation	349.5
2	7453.5	Swiss National Supercomputing Centre	1312
3	6673.8	Advanced Center for Computing and Communication	150.0
4	6051.3	National Supercomputing Center in Wuxi	15371
5	5806.3	Fujitsu Technology Solutions GmbH	77
6	4985.7	Joint Center for Advanced High Performance Computing	2718.7
7	4688.0	DOE/SC/Argonne National Laboratory	1087
8	4112.1	Stanford Research Computing Center	190
9	4086.8	Academic Center for Computing and Media Studies Kyoto University	748.1
10	3836.6	Thomas Jefferson National Accelerator Facility	111

TABLE 2.3 – Le classement GREEN-500 du 14 novenbre 2016 des supercalculateurs les plus efficaces du point de vue energétique dans le monde.

Les physiciens ont besoin du calcul haute performance dans leurs analyses. Mais ils ne peuvent pas développer la même chose pour toutes les expériences et n'ont pas les connaissances spécifiques en programmation. Des librairies communes doivent être développées par une collaboration entre physiciens et informaticiens pour allier aussi bien des performances de physique que d'informatique. Le projet ASTERICS, présenté dans la section suivante, a pour objectif de résoudre ce problème.

2.5 Le projet ASTERICS

Le projet ASTERICS (Astronomy ESFRI and Research Infrastructure CluSter [37]) fait parti du projet ESFRI horizon 2020. Son objectif est de fournir des librairies, ou des retours d'expériences qui permettront l'analyse de données pour différentes expériences de physique telles que CTA (rayons gamma), SKA (ondes radio), LSST(infrarouge), E-ELT(lumière visible suite du VLT, Very Large Telescope), LIGO-VIRGO EGO (ondes gravitationnelles) ou KM3Net (neutrinos).

Les analyses de données bas niveau de ces expériences ont de nombreux points communs comme des calculs de réductions, de barycentres, des multiplication matrice-matrice, etc. D'autres points communs de plus haut niveau existent tels que le calcul de cartes d'acceptance, de spectres et bien d'autres.

Les objectifs du projet ASTERICS sont multiples :

— Former les chercheurs et les informaticiens aux techniques de la programmation parallèle et à l'utilisation efficace des processeurs récents.
— Maximiser la réutilisabilité et l'interopérabilité des solutions développées.
— Adapter et optimiser les grandes bases de données par la coopération des centres de calcul et de l'industrie.
— Étudier et démontrer l'intégration des solutions proposées par le biais d'outils de fouille de données et d'analyses statistiques sur des volumes de données conséquents (Péta-octets).

Tous ces développements sont effectués dans le but de créer des standards et des librairies open-source pour les données multi-messagers/multi longueurs d'ondes, et ce afin de proposer des solutions communes pour traiter des flux de données en provenance de grandes bases de données. Ils se baseront sur des algorithmes d'analyses avancées pour le traitement des données et le contrôle de leur qualité.

Nous nous sommes fixés un objectif supplémentaire qui consiste à assurer que les fonctions fournies seront les plus rapides possible, dans le but d'améliorer également la vitesse d'exécution des analyses.

2.6 Objectif de la thèse

L'objectif de ce travail de thèse est de fournir un ensemble de librairies et d'outils réutilisables permettant de développer des analyses HPC. Ces librairies et ces outils feront partis des travaux délivrés au projet ASTERICS fournis par le LAPP. Leur application première sera l'optimisation de l'analyse de données de l'expérience CTA, tant informatique que physique :

— La définition et la génération d'un format de données hautes performances (voir chapitre 3).
— Le développement d'une méthode de compression rapide et efficace (voir chapitre 4).
— L'optimisation de l'analyse de données classique, Hillas (voir chapitre 5).
— L'adaptation et l'optimisation d'une analyse plus avancée basée sur la comparaison d'images (voir chapitre 6).

L'essentiel du chapitre 2 –

Le projet CTA sera la plus grande expérience d'astronomie gamma au sol d'ici à 2021. Il vise à gagner un ordre de grandeur en sensibilité par rapport aux expériences existantes. Il sera constitué d'un site nord à La Palma (Espagne) composé de 19 télescopes et d'un site sud dans le désert du Paranal (Chili) composé de 100 télescopes dans leurs premières versions respectives.

Le flux de données produit sera sans précédent en astronomie gamma. Cela change la manière classique de les traiter, de les compresser ou de les stocker. Le format de données, les méthodes d'analyse et les techniques de compression devront s'adapter à cette nouvelle échelle.

Le projet ASTERICS a été créer dans le but de rassembler les différentes communautés de l'astronomie gamma et de réutilisé des outils existants ou en cours de développement. Son objectif est également de former les physiciens et les informaticiens aux nouvelles techniques de programmation de l'informatique hautes performances (HPC).

Le travail présenté dans ce manuscrit fait parti intégrante du projet ASTERICS et utilise comme cas test le projet CTA. Les prochains chapitres présenteront respectivement les optimisations du format de données, de leur compression ainsi que les deux méthodes d'analyse les plus efficaces de l'expérience H.E.S.S. (la plus grande expérience d'astronomie gamma actuelle).

Chapitre 3

Générateurs de format de données

Sujet du chapitre 3 –
Un format de données bien adapté permet d'obtenir d'excellentes performances combiné au calcul haute performance. Ce chapitre, décrit chronologiquement l'amélioration du format de données utilisé par CTA à l'aide d'un générateur de format de données.

Sommaire

CTA est une expérience qui produira une quantité de données sans précédent en astronomie gamma (voir chapitre 2). Ces données devront être stockées et traitées le plus rapidement possible sur sites et être ensuite transférées en Europe où elle y seront analysées plus finement. La totalité des données acquises sera réanalysée tous les ans avec la nouvelle version de l'analyse de données. Le format de données de CTA devra donc être particulièrement efficace en terme de lecture, d'écriture et de facilité de calcul. Ces calculs ne pourront être performants qu'avec l'aide du calcul haute performance (HPC). Par conséquent, ce format de données devra être compatible avec les impératifs du HPC.

3.1 Situation de CTA

CTA est une expérience actuellement en cours de conception et de construction. Les caméras sont en phase de prototypage et leur données ne sont pas encore convenablement exploitables. Seules des simulations Monte-Carlo des télescopes sont utilisées pour le moment, elles se basent sur le format de données Simtel. Le model de données de CTA est très flou [38] et le format de données n'est pas non plus défini. Les différentes communautés provenant d'expériences comme H.E.S.S. ou MAGIC cherchent à réutiliser au maximum ce qu'elles ont déjà développées, notamment leurs formats de données qui permettent de réutiliser directement leurs analyses. Chacune d'elles a donc développées un convertisseur entre les données Simtel et leur format de données ce qui leur permet d'améliorer leur analyse en l'absence d'un format de données officiel.

La possibilité d'utiliser le format de données d'une expérience existante a été envisagée. Ces formats sont nombreux car quasiment chaque équipe possède le sien mais les plus utilisés sont :

- Simtel : utilisé pour décrire les simulations Monte-Carlo des télescopes Cherenkov au sol, dont CTA. Ce format est développé en C et est maintenu depuis 1992.
- Le format de données de H.E.S.S. ou celui de MAGIC sont tous les deux basés sur la librairie Root [25] (C++) utilisée en physique des particules. Ils ont été développés dans les années 2000.
- Le format de données de l'expérience FACT, ZFits [39].
- Les formats de données développés par les groupes en charge de l'analyse de données en temps réel sur GPU Nvidia K80 ou ARM.

Nous estimons que ces formats de données ne sont pas adaptés à la nouvelle échelle que CTA représente. Ils ne permettent pas d'utiliser efficacement le HPC et ne sont donc pas adaptés pour CTA. Comme nous l'avons mentionné précédemment (voir chapitre 2), l'analyse de données de CTA sera principalement écrite en Python. Les points les plus critiques seront néanmoins développés en C++ pour garantir une exécution efficace[1]. Nous pensons que le format de données proposé devra être adapté pour ces deux langages, et satisfaire aux exigences du HPC. Naturellement, le nombre de format de données adaptés à ces nouvelles

1. Le langage C++ est plus bas niveau que le Python et est donc plus performant.

contraintes est maintenant extrêmement réduit. Nous avons donc développé notre propre format de données afin que celui-ci permette le HPC.

Développer un format de données complet qui soit à la fois simple d'utilisation et extrêmement efficace est une gageure. D'autant que de nombreux tests devront être effectués pour déterminer de manière sûre le format de données adéquate pour CTA. Cela impliquerait d'écrire tous ces formats de données et de comparer leur performances ! Il n'est certainement pas raisonnable de réaliser tout ce travail à la main. Une méthode automatisée permettrait d'évaluer un plus grand nombre de possibilités tout en économisant du temps de développement.

Une équipe en charge de l'acquisition des données de CTA, développeur de ZFits, a déjà fait des tests dans ce sens en utilisant le générateur de format de données *Protocol Buffer* développé par Google. Ce générateur permet, à l'aide d'un fichier de configuration assez complexe, de générer un format de données en C++, Python, Java, Javascript, PHP, etc. Tous ces langages offrent des interfaces différentes mais permettent de lire le même fichier binaire. Cette compatibilité est très appréciable, et est même au-delà du cahier des charges de CTA (Python et C++).

Nous avons travaillé en collaboration avec cette équipe pour évaluer le format de données généré à l'aide de *Protocol Buffer* du point de vue du HPC et de l'attente que pourrait avoir un utilisateur de CTA. Malheureusement, le format de données généré ne permet pas une vectorisation efficace car les tableaux ne sont pas alignés sur des frontières de registres vectoriels. De plus, il utilise l'introspection, qui ajoute de la flexibilité dans le développement mais qui ralenti la lecture et l'écriture des fichiers binaires. Aucune documentation ne décrit le code généré ce qui peut être gênant pour l'utilisateur de ce format de données. De plus, tenter de lire ce code peut s'avérer d'autant plus déplaisant puisqu'il est extrêmement complexe au point qu'un informaticien aura beaucoup de difficultés à l'appréhender. Cette difficulté est accentuée par le fait que le code généré est également très mal indenté. Toute cette complexité a tendance à induire en erreur les environnements de développement les plus évolués comme *KDevelop*.

Le code généré est également très difficile à utiliser. Nous prenons ici l'exemple du C++, qui sera utilisé pour les fonctions HPC. Dans ce cas, toutes les données binaires sont stockées dans des std::string qui est une classe de la librairie standard du C++ utilisée habituellement pour stocker des chaînes de caractères. Outre le fait que cela rend leur alignement quasiment impossible, l'utilisation d'un tableau de données ne peut se faire qu'en utilisant un *dynamic_cast*. Cela implique que l'utilisateur ne peut récupérer une donnée que si il connaît à l'avance son type (car tous les pointeurs de données renvoyés sont des char*). Cette utilisation est potentiellement une source d'erreurs importantes qui les rendra très difficiles à corriger.

Toutes ces difficultés impliquent que ce code généré sera inutilisable par un physicien. Or, il est impératif que la communauté des physiciens ne soit pas exclue du développement du format de données car ce dernier est un outil important pour eux. La conception de ce format de données devrait donc se faire à part égale entre les informaticiens et les physiciens.

L'équipe de développement s'en est rendue compte et a conçue une interface par dessus le code généré par *Protocol Buffer* pour le rendre plus accessible. Malheureusement, cela ne fait que déplacer le problème, car si un code généré doit être caché systématiquement sous une interface, le développement obligatoire de cette interface efface tout l'intérêt d'utiliser un générateur de format de données.

Le programme *Protocol Buffer* n'est pas une exception. D'autres, comme *Avro* ou *Thrift* ont des utilisations similaires.

3.2 Objectifs

CTA traitera un très grand nombre de données. Leur format devra donc être performant en lecture et en écriture. Pour cela, une utilisation de structures de tableaux est préférable à une arborescence en tableaux de structures.

Les données des tableaux de types simples doivent pouvoir être alignées sur des frontières de registres vectoriels automatiquement. Le format de données devra adapter l'alignement des tableaux, matrices et tenseurs en fonction de l'architecture ciblée et du type de données à aligner. Il est important de veiller à ce que les données soient alignées en utilisant bien l'architecture adéquate car cela peut être pénalisant dans le cas de petites matrices possédant des pitchs [2]. Dans ce cas, si l'alignement choisi est plus grand que la taille des registres vectoriels (ce qui est compréhensible par soucis de simplicité) une matrice munie d'un pitch aura une taille mémoire plus importante. Cette fonctionnalité est décisive pour CTA car les images traités ne possèdent que quelques milliers de pixels et sont donc stockées dans de petites matrices.

L'utilisation du générateur de format de données devra être aisée ainsi que celle du format de données généré. Les fonctionnalités de ce générateur seront conçues pour offrir à l'utilisateur une constante simplicité. L'utilisation de tableaux sera fortement mise en avant tendis que des options telles que l'introspection ou l'héritage de classe seront éliminées. Ceci contribuera à simplifier d'avantage la configuration du format de données généré, son utilisation et également le développement du générateur proprement dit.

Une documentation du code généré sera également fournie pour simplifier la tâche de l'utilisateur. Le code généré sera facilement lisible et compréhensible par des physiciens (utilisateurs finaux de ce format de données). Il sera aussi conçu pour aider les développeurs qui utilisent des environnements de développement comme *KDevelop*.

3.3 Conceptions et architectures logicielles

Notre générateur de format de données est développé en C++. Ce langage allie flexibilité, robustesse et maintenabilité. Sa composante objet joue un rôle important dans la définition de la représentation interne du générateur.

2. Un pitch est un ensemble d'éléments supplémentaires ajoutés à la fin de toutes les lignes d'une matrice de sorte à ce que ces dernières soient toutes alignées sur une frontière de registre vectoriel.

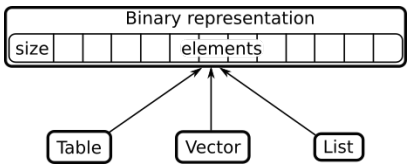


FIGURE 3.1 – Représentations binaires des tableaux, des listes et des vecteurs de données.

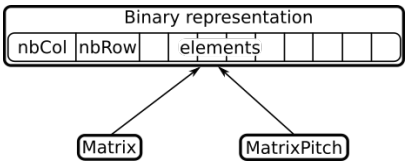


FIGURE 3.2 – Représentations binaires des matrices (avec et sans pitch).

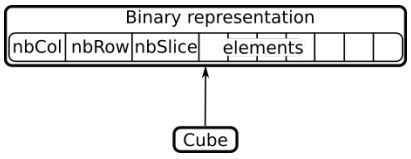


FIGURE 3.3 – Représentation binaire des tenseurs à trois dimensions (aussi appelés Cube).

Par soucis de simplicité, le générateur de format de données utilise des fonctions bas niveaux fournies dans la ***PLIBS_8*** [40] (cette librairie contient l'ensemble des contributions en calcul haute performance de cette thèse, sous forme de librairies et de programmes réutilisables). Cette méthode permet de simplifier son développement tout en assurant une cohérence des fonctionnalités proposées. Les différentes fonctionnalités présentes dans la ***PLIBS_8*** ont été vérifiées à l'aide de tests unitaires pour assurer leur pertinence et leur robustesse. L'avantage de cette méthode est que ces fonctions bas niveau sont utilisables sans recourir au générateur. Lorsque celles-ci sont mises à jour, tous les codes qui les utilisent se voient améliorés, y compris le format de données généré. L'utilisateur n'a jamais besoin de recréer son format de données en cas de mise à jour. Dans ces conditions le générateur n'est qu'un moyen de développer plus rapidement un format de données que l'on pourrait obtenir manuellement. Ce qui, une fois de plus, contribue à simplifier l'approche avec cet outil aussi bien en tant qu'utilisateur que programmeur.

Les fonctionnalités principales sont :

— La lecture et l'écriture des fichiers binaires.
— La compression des données, avec ou sans perte (voir chapitre 4).
— La conversion entre deux classes du format de données généré.
— La conversion des données en messages binaires pouvant être transmis sur le réseau.

Permettre le stockage de données binaires implique d'assurer une cohérence entre les différentes manières de les décrire. Les figures 3.1, 3.2, et 3.3 illustrent la description binaire des tableaux, des listes, des vecteurs, des matrices (avec ou sans pitch) et des tenseurs à trois dimensions. Cette manière de faire offre une grande flexibilité. Les tableaux (alignés automatiquement par la ***PLIBS_8*** ou non) sont compatibles avec les vecteurs (std::vector) et les listes (std::list). Il est donc possible d'écrire un fichier binaire en utilisant un format flexible de type vecteur, et de le lire avec un format plus optimisé de type tableau. Une matrice est toujours stockée le la même façon, quelle possède un pitch ou non, puisque le pitch dépend d'une caractéristique de l'architecture hôte et du type de donnée utilisé. Les tenseurs à trois dimensions seront très utiles dans CTA pour stocker les données provenant des caméras (voir section 5.1.2).

Le format de données généré doit également permettre de charger des fichiers à partir du langage Python pur. Toutefois, un appel de fonction en Python pur est très lent (typiquement 300 cycles contre $1-3$ cycles pour le C++). La seule façon de résoudre ce problème est d'écrire le format de données en C++ puis de le wrapper. La figure 3.4 illustre cette démarche. Chaque classes générée possède un wrapper qui permet de la lire et de la sauvegarder en binaire via une fonction Python. L'organisation de ces appels implique que le wrapper ne sera utilisé qu'une seule fois quelque soit la données désirée. Cela limite considérablement la perte de temps due aux appels de fonctions Python.

L'appel des fonctions permettant de compresser les données ou de les convertir en messages fonctionne également sur ce principe.

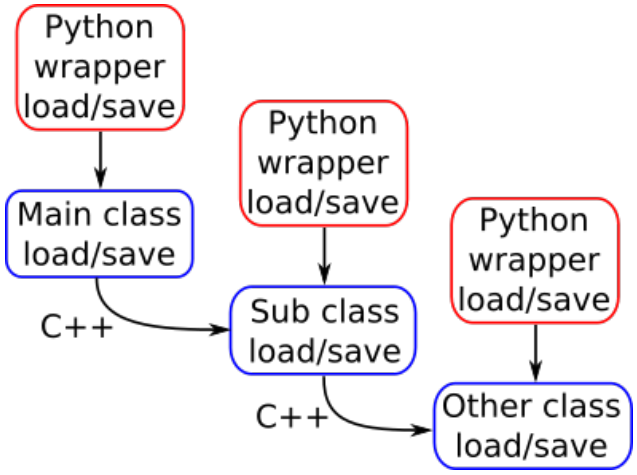


FIGURE 3.4 – Illustration de l'appel du wrapper Python. Chaque niveau du format de données a son propre wrapper. Tous les appels suivants un wrapper sont en C++ pour assurer une exécution rapide.

3.3.1 Première version

La première version de ce générateur de format de données était à l'origine une version d'appoint. Cela permettait à notre équipe d'améliorer sa version de l'analyse de données HPC de CTA sans devoir attendre une décision concernant le format de donnée final de cette expérience. Son temps de vie, initialement prévu à quelques mois, se prolongea sur plus de deux ans.

La figure 3.5 montre l'architecture de ce générateur. Un fichier de configuration est lu par le front-end, lui-même composé de deux parties. La première, ***MultiConfigParser***, gère les différentes versions du format de données. La seconde, ***ConfigParser***, convertie la version par défaut dans la représentation interne appropriée. Pour cela, le informations décrites dans le fichier de configuration sont réparties dans les différentes classes qui décrivent les back-end. Ces back-end permettent d'écrire toutes ces données dans le langage ciblé (C++, Python pur ou wrapper Python).

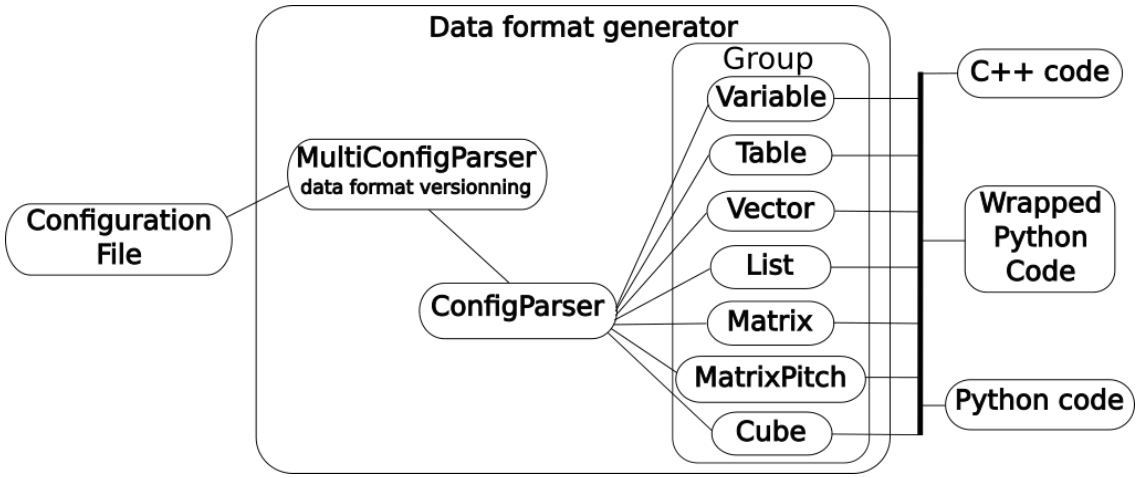


FIGURE 3.5 – Architecture de la première version du générateur de formats de données.

La figure 3.6 illustre la représentation interne du générateur. Dans celle-ci, la classe la plus générale décrit une variable simple, elle définie l'ensemble des back-end par défaut à l'aide de fonctions virtuelles. Ces fonctions seront surchargées dans les autres classes afin de décrire l'ensemble des back-end de toutes les structures de données fournies par le générateur.

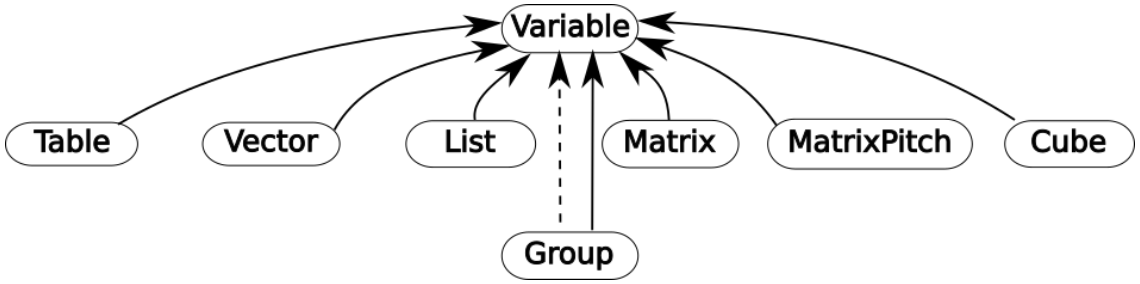


FIGURE 3.6 – Représentation interne de la première version du générateur de formats de données. Les traits pleins représentent l'héritage entre les classes. Les traits pointillés décrivent l'utilisation d'une classe sous forme d'attribut dans une autre.

La représentation interne utilise le design patern : patron de classe. Elle permet une grande flexibilité puisque toutes les classes héritant de la classe ***Variable*** peuvent être stockées dans une liste de pointeurs de ***Variable***s. Ce qui réduit le nombre de listes à utiliser pour décrire un nombre arbitraire de classes. De plus, si une nouvelle classe est ajoutée pour décrire un nouveau type de stockage, elle sera incorporée dans le programme sans aucun développement supplémentaire de par sa parenté avec la classe ***Variable***.

Néanmoins, cette méthode est à utiliser avec parcimonie puisque qu'elle dégrade les performances du programme. En effet, le patron de classe transforme le flux d'instruction en flux de données ce qui rend leur préchargement plus difficile à prédire pour le processeur. Cela est dû au fait que les commandes présentes dans le cache des instructions (L1i) sont dépendantes des données dans leur propre cache (L1d).

Ce programme fut développé pendant plus de deux ans. De nombreuses fonctionnalités n'étaient pas définies au départ et on été ajoutées au fur et à mesure des besoins. Ces tests ont notamment permis d'évaluer les performances d'un format de données en Python pur afin de le comparer à des techniques plus évoluées telles que swig [41] ou pybind11 [42] (voir section 3.5).

Une première ébauche d'une compression interne au format de données généré à également été développée. Cela à permis de montrer la faisabilité d'un tel concept, et l'avantage qu'offrait l'automatisation du développement d'une sous-couche logicielle de compression comparée à un développement manuel.

Des tests de transmission de données sur le réseau ont pu être menés afin d'améliorer les performances du programme d'auto-équilibrage de l'analyse de données développée au LAPP [43]. Ce framework permet de manipuler des processus qui communiquent entre eux via la librairie ***ZeroMQ*** [44]. Il est donc crucial que les données échangées se convertissent en messages binaires.

Tous ces développements ont mis en lumière le fait que ce programme devenait de plus en plus difficile à mettre à jour et à maintenir. En effet, chaque nouvelle fonctionnalité nécessite un back-end, défini par un ensemble de fonctions virtuelles. Ce back end doit être développé pour chaque classe de la représentation interne. De plus, certains d'entre eux doivent être mis à jour comme les *getter*s, les *setter*s ou encore la fonction qui définie la copie de la classe générée. Ceci était principalement dû au fait que ces back-end n'étaient pas complètement indépendants. Dans ce cas, l'ajout de nouvelles fonctionnalités nécessite de plus de plus de développements. Cela est accentué par le fait que les fichiers sources deviennent de plus en plus imposants, rendant leur lecture complexe (malgré l'utilisation d'une convention très stricte). Bien évidemment ce rythme devint impossible à suivre.

Néanmoins, cette méthode et l'utilisation du patron de classe permirent d'obtenir très rapidement une première version opérationnelle. Cette version très simpliste (uniquement composée des *getter*s, des *setter*s, d'un opérateur égal, d'un constructeur de copie, des fonctions de lecture et d'écriture de fichiers binaires pour le langage C++) permit d'augmenter sensiblement la vitesse de développement consacrée aux optimisations de l'analyse de données de CTA.

Cette étude fut partagée avec la collaboration CTA sous la forme d'une note interne [45] décrivant en détail la construction d'un format de données haute performance. Elle permit d'apporter de nouveaux éléments indispensables au choix du format de données qu'utilisera CTA. Le débat était relancé.

3.3.2 Deuxième version

La première version du générateur de format de données permit de hiérarchiser les fonctionnalités nécessaires à CTA. Le partage de notre note interne sur les propriétés HPC d'un format de données apporta de nouveaux arguments sur son choix dans CTA.

En réponse à notre note interne, le responsable de l'analyse de données de CTA, rédigea également une note [46]. Il a pris en compte nos remarques et définit l'ensemble des propriétés que devra fournir le format de données quel qu'il soit.

Dès lors, les choses se sont clarifiées et l'objectif du format de données de CTA était mieux défini. Voici une liste exhaustive des propriétés désirées :

1 - La maintenance et la longévité du format de données est essentielle. L'utilisation d'un standard est envisageable bien que ce ne soit pas le cas pour l'instant. La documentation doit être suffisante pour réécrire un lecteur de toutes pièces si l'API originale est perdue (bien que cela soit peut probable grâce aux méthodes modernes de gestion de projet comme *github* [47] ou *gitlab* [48]). Les personnes en charge de ce développement doivent être clairement identifiées et devront le maintenir pendant au moins 30 ans (le temps de vie de CTA). La présence de dépendances extérieures peut être problématique sur le long terme, tendis que de multiples implémentations seront gage de sûreté *Mais de multiples implémentations seront plus difficiles à maintenir qu'une seule.*

2 - Le schéma utilisé pour décrire les données doit aussi être clairement défini. Il est préférable que cette description soit présente dans les fichiers binaires. Cette propriété d'auto descriptivité sera d'autant plus importante tant que le format de données ne sera pas complètement définitif. Comme nous l'avons indiqué précédemment, un format de données basé sur des tableaux ou des tenseurs sera préféré car cela simplifie son utilisation et améliore ses performances. Mais la définition des tenseurs ou des tableaux doit être souple au point que les types décrivant le nombre d'éléments sur chaque dimension peuvent être indiqués par l'utilisateur. Par défaut, les dimensions sont stockées dans des long unsigned int (size_t) mais l'utilisateur pourra si il le désire les changer en tout autres types (unsigned int, unsigned short, unsigned char, etc). Le schéma doit pouvoir évoluer sans altérer la lecture des fichiers contenant d'anciennes versions du format de données. Nous ajouterons même que cette évolution doit avoir un impact minimal, tant sur le temps de développement de l'utilisateur qui désire changer la configuration que sur les performances de lecture et d'écriture du format de données.

3 - L'architecture de la machine cible doit également être prise en compte. Dans l'idéal, le format de données doit être indépendant de l'hôte, architecture (endianness), compilateur, OS (Linux, MacOS) et du langage (C++, Python). La détection de la corruption d'un fichier y est mise en avant. Et la possibilité de restaurer un fichier corrompu est d'autant plus importante que les données enregistrées seront utilisées pendant toutes la durée de l'expérience. Les fichiers de données auront probablement un volume de quelques GO, par conséquent le format de données devra permettre de ne pas charger complètement un fichier en RAM. Le format de données devra également supporter les impératifs du HPC. Enfin, une lecture parallèle des fichiers et éventuellement une écriture parallèle seront appréciables.

4 - Les performances du format de données seront décisives, tant en lecture qu'en écriture, avec ou sans compression. Elles concernerons les temps de chargement des volumes de données et leur temps de traitement associés.

Quelques propriétés étaient déjà présentes dans l'ancien générateur de format de données. Bien que le format de données générées ne soit pas un standard (par définition), voir paragraphe **1**, l'utilisation fréquente de tableaux et la possibilité de gérer des versions répondaient à une majorité des propriétés du paragraphe **2**. Toutes les remarques du paragraphe **3** étaient elles aussi satisfaites. Les performances du format de données généré, demandées dans le paragraphe **4**, sont décrites dans la section 3.5. La suite détaille comment répondre aux autres exigences de la note sur le format de données de CTA.

L'ensemble des propriétés décrites ci-dessus permet de cerner d'avantage le travail à effectuer. Le développement d'un nouveau générateur de format de données est maintenant une évidence. Puisque de plus en plus de temps de développement est nécessaire pour ajouter des fonctionnalités dans le précédent. Celui-ci devra être bien plus complet. Son développement devra être d'autant plus flexible que l'ensemble des fonctionnalités demandées prendra un certain temps de développement.

Contrairement au précédent, toutes les fonctionnalités développées pour le langage C++ devront aussi être disponibles en Python (via un wrapper). Le langage Python pur sera quant

à lui abandonné [3], au profit des wrappers.

3.3.2.1 Compatibilité C++/Python

La compatibilité de lecture des fichiers binaires est indispensable entre le C++ et le Python. Les vitesse de lecture et d'écriture des fichiers en Python sont supérieures en utilisant un wrapper. Nous avons effectués des tests de performances sur les différentes méthodes de wrappers existantes.

Le programme *swig* permet de wrapper rapidement un format de données déjà écrit en C++. Les vitesses de lecture et d'écriture sont comparables à celles du C++. Malheureusement, tous les tableaux de données sont copiés lors de leur renvoie par un *getter*. Renvoyer une variable avec cette méthode est 75 fois plus lente que du Python pur.

La librairie ***pybind11*** [42] fournie une interface pour wrapper du C++. Elle permet d'obtenir des performances comparables à l'API Python mais ajoute une dépendance au format de données généré.

Le Python fournie une API permettant de wrapper des fonctions C++. Cette API est extrêmement complexe à utiliser mais permet d'obtenir les meilleures performances. Elle permet également de ne pas copier les attributs renvoyés et d'aligner, manuellement, les tableaux, matrices et tenseurs. L'utilisation de pitch, appelés *strides*, est aussi possible.

Afin de garantir qu'aucune copie se sera effectuée après la lecture des fichiers, le wrapper sera lui-même le format de données final utilisé pour le Python. Cela implique que tous les conteneurs de type complexe seront transformés en listes Python dans le format de données généré.

Le type de structure de donnée est donc légèrement différent entre le C++ et le Python même si le format de données est identique en binaire. Cependant, tous les tableaux, matrices ou tenseurs de types simples sont identiques et ont exactement les mêmes propriétés HPC. Les formats de données générés seront donc performants dans les deux langages.

3.3.2.2 Représentation interne

La représentation interne de ce générateur doit permettre une grande flexibilité dans le développement tout en garantissant une constante simplicité. L'accroissement des temps de développements dédiés au générateur précédent était principalement dû au fait que les différents back-end n'étaient pas totalement indépendants. Ils devaient également tous être définis en même temps. L'expérience acquise sur le générateur précédent montre que le patron de classe n'est pas une bonne solution sur le long terme, il ne sera donc pas utilisé dans ce nouveau programme.

Si le patron de classe est exclu du projet, quid de l'héritage et des fonctions virtuelles ?

3. Nous étions arrivé au point que le format de données en Python pur (déjà optimisé) chargeait les données en trois heures tandis que le format C++ faisait de même en 40 secondes (sur huit cœurs). Ce qui était bien trop long.

L'héritage assurait une abstraction qui permettait de stocker des types différents dans la même liste. Cette propriété était appréciable d'autant qu'elle était indépendante du nombre de types à traiter. Les fonctions virtuelles permettaient quant à elles de définir les différents back-end des langages et des types de données utilisés. La demande de simplicité de la nouvelle représentation interne implique de trouver une nouvelle méthode de définition des back-end. Les fonctions virtuelles seront donc supprimées de cette représentation interne. Dans ce cas, l'héritage n'apporte plus de fonctionnalités utiles. Il sera donc également supprimé.

Le nouveau générateur de format de données sera principalement axé sur le C++ et les wrappers Python. Nos tests de performances (voir section 3.5) montrent que l'utilisation des wrappers Python (avec l'API Python) fournissent de bien meilleures performances que le Python pur. Or, l'API Python est écrite en C et l'interface de wrappers Python de la PLIBS_8 est développée en C++.

La représentation interne doit donc décrire le même langage (C++) pour les deux back-end (C++ et wrappers Python). La solution est donc de développer une représentation interne pour manipuler des sources C++.

La figure 3.7 illustre la représentation interne du nouveau générateur de format de données. Sa partie principale décrit un fichier source C++ de manière général, composé de classes, de structures, de fonctions et de variables globales. Quelques classes supplémentaires permettent de stocker les options de génération demandées et de définir des conversions entre deux classes générées.

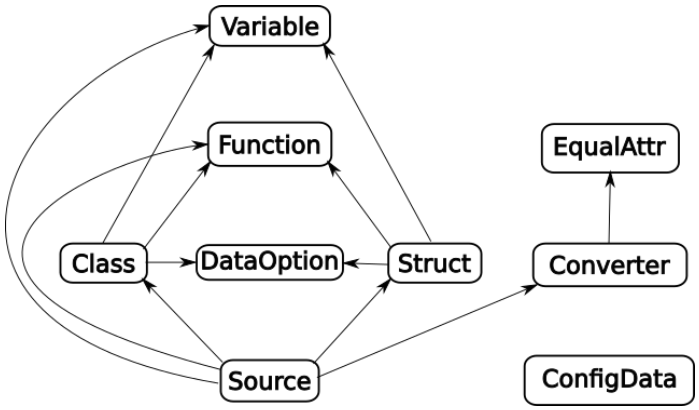


FIGURE 3.7 – Représentation interne de la deuxième version du générateur de formats de données. Aucun héritage n'est utilisé entre les classes. Les traits pleins décrivent l'utilisation d'une classe sous forme d'attribut dans une autre.

Cette représentation interne est produite par le générateur de format de données précédent. Cela augmente la rapidité de développement et conserve la flexibilité. La description du code généré utilise principalement des chaînes de caractères qui améliore la flexibilité et simplifie le développement des back-end.

L'ensemble des classes contiennent une chaîne de caractères de documentation au format *doxygen* qui sera écrite dans le format de données généré. Cela garantie que toutes les classes générées seront documentées.

La classe *Source* décrit à la fois les en-têtes et les sources C++ des fichiers à générer. Elle contient des vecteurs (std:vector) de classes, structures, fonctions et variables globales. Des *include*s ou du code supplémentaire peuvent être également ajouter dans le but de simplifier la création de fichiers wrappers.

La classe *Class* permet de manipuler une classe. Elle est préférablement utilisée pour stocker la représentation interne destinée au back-end C++. Elle décrit une classe habituelle en C++ composée de vecteurs d'attributs et de fonctions, publiques, protégés et privés, ainsi qu'un vecteur de classes amies (utilisé pour la gestion des classes compressées, voir section 3.3.2.5). Des options de génération sont également présentes via une classe ***DataOption***. Les différentes versions de cette classe décrites par un vecteur de ***Class***, de sorte que tous les formats propres à chaque classe soient connus.

La classe *Struct* est le pendent de la classe ***Class*** pour les wrappers Python via l'API C de Python. Elle pourra décrire des structures C++ mais sera principalement utilisée dans le back-end Python. Elle est composée de vecteurs d'attributs et de fonctions, ces dernières étant présentes pour simplifier la gestion de la représentation interne. Quelques attributs supplémentaires renseignent le nom du module wrapper, si il existe, ainsi qu'une chaîne de caractères qui précède la définition de tout attribut (*PyObject_HEAD* dans le cas des wrappers Python).

La classe *Function* décrit une fonction C++. Elle se compose classiquement d'un tableaux d'arguments, du type renvoyé et le corps de la fonction est décrit par une unique chaîne de caractères. Des attributs supplémentaires indiquent au back-end wrapper Python si la fonction est une méthode à ajouter dans les fonctions appelables de Python, et un dernier attribut détermine si la fonction a des paramètres nommés ou non.

La classe *Variable* est la plus importante de la représentation interne. Elle est utilisée comme attribut de classe ou de structure, paramètre de fonction ou variable globale dans les autres classes axées C++. Elle contient des attributs qui décrivent une variable classique en C++ : nom, type, constant, pointeur, référence, externe.

La classe ***Variable*** décrit aussi l'ensemble des structures de données fournie par le générateur : variables, tableaux, matrices, tenseurs à N dimensions avec ou sans pitch, alignés ou non. Elle doit être semblable à une arborescence (voir figure 3.8). Pour cela, elle contient un vecteur de ***Variable***, de cette manière la classe principale décrit le pointeur de données et les classes contenues décrivent les dimensions. Un vecteur (std::vector) est suffisant dans ce cas car il n'est pas nécessaire de supprimer des éléments.

L'alignement et le pitch sont tous deux calculés en prenant en compte l'architecture de la machine hôte. Cette fonctionnalité est fournie par la ***PLIBS_8***. Les caractéristiques de l'architecture sont déterminées automatiquement par l'analyse du fichier */proc/cpuinfo* sur Linux et l'appel du programme *sysctl* sous MacOS. Ces appels créent un fichier d'en-tête C++ qui contient des macros décrivant l'ensemble des extensions trouvées. De cette manière les librairies développées s'adaptent à l'architecture hôte.

Une valeur par défaut peut être définie lors de l'utilisation d'une variable comme argument

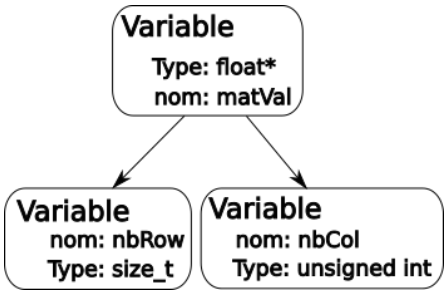


FIGURE 3.8 – Arborescence de la classe ***Variable*** pour décrire une matrice.

d'une fonction. Une valeur d'initialisation, pouvant être différente de la valeur par défaut, peut également être définie et sera utilisée lors de la création de la fonction d'initialisation d'une classe. Cette valeur d'initialisation peut être définie par l'utilisateur via le fichier de configuration du format de données qu'il désire produire.

Les derniers attributs de cette classe concernent plus particulièrement la compression de données avec ou sans perte du format généré. Une première variable décrit le nombre de bits a conserver lors de l'écriture du format de données en binaire (les N premiers ou les N derniers bits d'un type simple). Cette compression par décalage de bits est *a priori* avec perte dans le cas de nombres à virgule flottante et peut être sans perte dans le cas de nombres entiers. Elle fut, à l'origine, développée pour comparer l'algorithme de compression polynomial, décrit dans le chapitre 4, à un algorithme plus conventionnel et simple à développer. Un dernier attribut est présent pour désactiver la conversion d'une variable contenant un type complexe en classe Python. En effet, il est tout à fait possible qu'un utilisateur compose des classes générées avec d'autres qui n'ont pas leur équivalent Python. Les classes utilisées pour stocker le format de données compresser en sont un exemple (voir section 3.3.2.5). De cette manière, les classes existent bien dans le format de données Python, mais sont inaccessibles pour l'utilisateur. Elles ne sont utilisables qu'en C++ pur ou à l'intérieur d'un wrapper Python. Cette méthode qui consiste à cacher des variables à l'utilisateur permet d'éviter des erreurs de manipulation.

L'ensemble des classes, exceptée celle qui décrit les variables, contiennent la configuration textuelle utilisée pour créer le format de données. Cela permet de garantir l'auto descriptivité demandée par les recommandations du format de données de CTA. Lors de l'écriture d'une classe générée dans un fichier binaire, la configuration utilisée pour créer la classe sera écrite dans l'en-tête du fichier sous forme textuelle. Cette en-tête ne contiendra que les classes utiles permettant de lire le fichier en question. Ce qui limitera la taille de l'en-tête. De plus, il est indispensable que cette en-tête soit lisible par des programmes standard tels que *cat* ou *sed*. Cela garantira leur lecture sans imposer l'utilisation d'un programme développé spécialement dans ce but. Il est possible d'utiliser le programme *cat* pour afficher la configuration du fichier binaire. Néanmoins, il écrira la totalité du fichier dans le terminal, ce qui n'est pas le but recherché. L'utilisation du programme *head* peut résoudre partiellement ce problème, en indiquant le nombre de lignes à lire au début du fichier. Mais ce nombre de lignes devra a priori être ajusté ce qui peut être source d'erreurs suivant les besoins de l'utilisateur. Nous avons

donc opté pour qu'un séparateur décrive la fin de la configuration, de cette manière, celle-ci sera toujours extraite en intégralité sans données superflues. Il est donc préférable d'utiliser la commande *sed '/ENDOFCONFIG/q' fichierBinnaire* pour afficher la configuration.

3.3.2.3 Architecture logicielle du générateur

La représentation interne est la clé de voûte de ce nouveaux générateur. La figure 3.9 montre l'architecture de ce générateur. Un fichier de configuration est lu par le front-end qui le transforme en représentation interne.

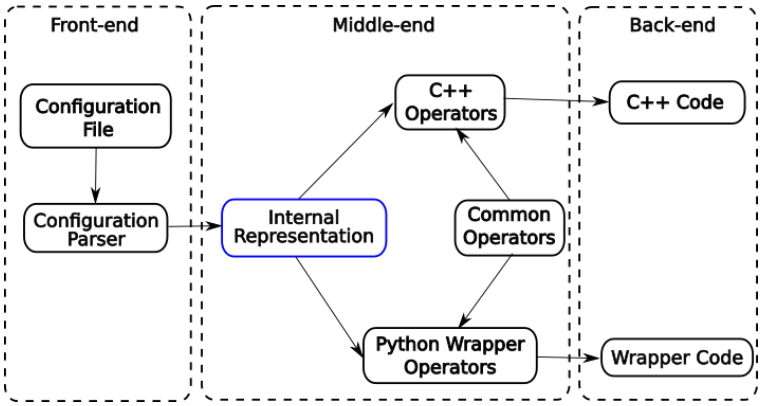


FIGURE 3.9 – Illustration de l'architecture de la deuxième version du générateur de formats de données.

Cette représentation interne sera modifiée par des fonctions (ou opérateurs) appelées chacune à son tour. Tous les opérateurs sont indépendants et chaque opérateur a un rôle unique. Ils sont conçus pour être appelés dans un ordre quelconque. Les opérateurs les plus simples créent les constructeurs, les *getter*s, les *setter*s, l'opérateur égal, la fonction et le constructeur de copie d'une classe. Ils sont néanmoins composés d'opérateurs élémentaires, qui permettent de copier, d'allouer ou de renvoyer un élément d'un type quelconque (en prenant en compte son alignement et son pitch si nécessaire). Leur forte interopérabilité permet de les composer pour créer des opérateurs destinés au wrapper Python à partir de ceux du C++ ou d'autres encore plus complexes. Cette propriété implique que de moins en moins de développements sont nécessaires au fur et à mesure que le nombre d'opérateurs disponibles augmente.

Un utilisateur n'a pas à savoir si une structure de données doit être alignée ou non, avec un pitch ou non, c'est le rôle du format de données. Par conséquent, des opérateurs ont été ajoutés pour créés des fonctions d'allocation et de désallocation de toutes les structures qui nécessitent une allocation. Ces fonctionnalités sont particulièrement utiles pour les utilisateurs qui développent en Python. En effet, ils n'ont pas à allouer eux mêmes leurs tableaux avec ***numpy*** qui n'aligne pas les tableaux sur les architectures supérieures à SSE4. Un module Python a tout de même été développé dans la ***PLIBS_8*** dans le but d'aligner des tableaux ***numpy***, mais il est tout de même préférable que l'utilisateur final n'ait pas à ce préoccuper de cela.

Un opérateur permet d'ajouter des fonctions de lecture de données pour ne charger qu'une seule partie d'un fichier en mémoire. Cela répond à une problématique de CTA définie plus tôt.

La majorité des opérateurs définis ajoute des fonctions dans les classes ou les structures de la représentation interne. Cependant, quelques opérateurs ajoutent des classes supplémentaires dans le but de simplifier le travail de l'utilisateur du générateur. C'est ainsi qu'un opérateur permet de convertir une classe en structure de tableaux (voir section 3.3.2.4) et qu'un autre permet de créer un format de données compressé (voir section 3.3.2.5).

Les derniers opérateurs concernent la création des modules Python indispensables au wrapper. Un opérateur ajoute un fichier qui définie un module Python et un autre écrit le fichier de configuration *setup.py* qui permettra d'installer le format de données généré comme un module Python.

Le back-end est unique (contrairement à la version précédente) puisque le format de données C++ et le wrapper Python utilisent tous les deux des sources C++. Cela simplifie grandement le développement de ce générateur car, dans ce cas, le back-end n'a pas besoin d'évoluer. Quelques modifications ont été ajoutées au tout début pour permettre la sauvegarde d'un wrapper Python mais cela n'a quasiment rien changé à son écriture.

3.3.2.4 Structures de tableaux

Il est généralement plus simple de comprendre un tableaux de structure qu'une structure de tableaux dans un format de données. Cependant, les données organisées en structures de tableaux sont plus efficaces que celles en tableaux de structures en terme de temps d'accès et de vectorisation (voir figure 3.10). Un opérateur a donc été ajouté dans ce but : convertir une classe en structure de tableaux. Cette nouvelle classe est donc comparable à un tableau où chaque élément serait la classe précédente.

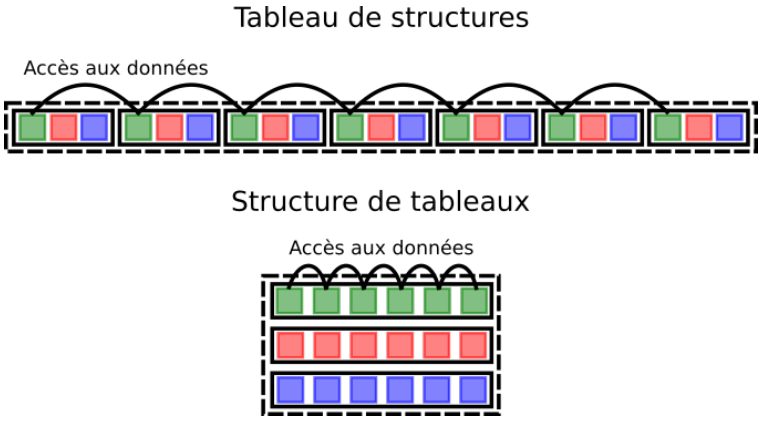


FIGURE 3.10 – Tableau de structures et structures de tableaux.

Cet opérateur fut relativement simple à développer grâce à l'interopérabilité des autres opérateurs. Tous les attributs de la classe de départ se voient ajouter une dimension. Néanmoins, la conversion de chaque type nécessite à conserver les propriétés HPC demandées par la classe initiale. Un tableaux d'éléments alignés est converti en matrice où toutes les

lignes sont alignées (grâce à un pitch). Une matrice alignée est transformée en tenseur à trois dimensions aligné et ainsi de suite.

Chaque tableau est accessible séparément. Il est également possible de renvoyer une classe de départ (qui correspond à un élément de chaque tableau de la classe supplémentaire), ou d'initialiser un élément de tous les tableaux avec la classe de départ. Cette fonctionnalité nécessite une copie des données car leur structure est différente mais elle contribue à simplifier l'utilisation de la classe supplémentaire.

3.3.2.5 Format de données compréssé

La question de la compression des données est cruciale dans CTA (voir chapitre 4). Le générateur de format de données permet donc d'incorporer des algorithmes de compression à l'intérieur même du format de données généré.

La compression la plus simple, décrite précédemment, utilise un décalage de bits. L'utilisateur spécifie dans le fichier de configuration le nombre de bits qu'il souhaite conserver dans le format de données binaire. Il peut ainsi conserver les N premiers ou les N derniers bits d'un type simple (type standard du C). Cette compression peut comporter des pertes dans le cas de nombres à virgule flottante et peut être sans perte dans le cas de nombres entiers. Elle n'est effective que lors du stockage des fichiers binaires. Il sont décompressés lors du chargement des données en RAM. Cette fonctionnalité est intéressante et a permis de faire des tests (voir chapitre 4), mais est limitée si les fichiers utilisés sont plus volumineux que la RAM disponible. Elle sera, dans ce dernier cas, combinée à d'autres méthodes, permettant de ne charger qu'une partie du fichier en RAM.

Une autre méthode est utilisée pour compresser des données sans perte. Un format de données parallèle est créé pour accueillir les données compressées. Dans ce cas, la compression est effectuée au plus bas niveau des données. Ainsi, uniquement les tableaux (au sens large) de types simples sont compressés.

Lors de la compression, tous ces tableaux sont convertis dans des tableaux de données binaires (typiquement renvoyé par des algorithmes de compression). Lors d'une décompression, c'est le contraire. Les algorithmes proposés pour l'instant sont le LZMA [49] (référence dans le domaine de la compression de données binaires sans perte) et la compression polynomiale (exposée dans le chapitre 4 de ce manuscrit).

Une dernière méthode existe dans le but d'exploiter au maximum les possibilités de l'algorithme LZMA. Elle permet de compresser des tableaux quelconques (ils peuvent contenir d'autres classes générée) par blocs. L'utilisateur peut spécifier que les tableaux d'une classe doivent se compresser par blocs, et peut indiquer le nombre d'éléments contenus par défaut dans un bloc. La compression et la décompression se font comme précédemment mais le taux de compression atteint est bien plus important (voir chapitre 4).

Par soucis de simplicité, la compression et la décompression d'une classe s'effectue, en C++, par l'appel de l'opérateur égal ou du constructeur de copie entre la classe de départ et son miroir compressé. Ces opérations sont également disponibles en Python par le biais d'un appel de fonction (la surcharge de l'opérateur égal n'est pas permise en Python). Ainsi, la

classe compressée peut se décompresser et inversement.

3.4 Description du langage proposé

Nous proposons ici un langage de description de format de données dont les types sont dérivés du C++.

Le langage utilisé pour décrire le format de données généré a été conçu pour être le plus simple possible. L'exemple suivant montre la déclaration d'une classe composée des types décrits précédemment :

```
1  ///@brief An other class
2  Gibi{
3    ///Gibi's chapeau
4    int chapeau;
5  }
6
7  ///@brief Test class
8  Shadok {
9    ///Age of the Shadok
10   int age;
11   ///Table of aligned value
12   Table(int, nbValue, AUTO) tabValue;
13   ///Matrix of aligned values
14   Matrix(float, nbRow, nbCol, AUTO) matValue;
15   ///Cube of aligned values
16   Matrix(float, nbSlice, nbRow, nbCol, AUTO) cubeValue;
17
18   Table(Gibi, nbGibi) tabGibi;
19   Matrix(Gibi, nbRowGibi, nbGibi) matGibi;
20   Cube(Gibi, nbSliceGibi, nbRowGibi, nbGibi) cubeGibi;
21 }
```

On notera que dans cette définition, la matrice **matValue** à le même nombre de lignes et de colonnes que le tenseur à trois dimensions **cubeValue**. Le format de données vérifiera donc systématiquement lors d'une lecture de fichier que cette propriété est bien respectée.

Il est également possible de définir d'autres types de stockage pour le nombre d'éléments par dimension :

```
Gibi{
  int chapeau;
}

Shadok {
  Table(int, nbValue[unsigned int], AUTO) tabValue;
  Matrix(float, nbRow[unsigned short], nbCol[unsigned int], AUTO) matValue;
  Matrix(float, nbSlice[unsigned char], nbRow[unsigned short], nbCol[unsigned int],
      AUTO) cubeValue;

  Table(Gibi, nbGibi[unsigned int]) tabGibi;
  Matrix(Gibi, nbRowGibi[unsigned short], nbGibi[unsigned int]) matGibi;
  Cube(Gibi, nbSliceGibi[unsigned char], nbRowGibi[unsigned short], nbGibi[unsigned
      int]) cubeGibi;
}
```

La compression par décalage de bits ne peut se faire que sur des types simples :

```
Shadok{
  Table(int[27], nbValue, AUTO) tabValue;
  Table(float[-16], nbValue, AUTO) tabFloat;
}
```

Dans ce cas, nous choisissons de sauvegarder les 27 bits de poids faible d'un entier sur 32 bits et les 16 bits de poids fort d'un float.

Une gestion des versions de classes est également possible :

```
Version(0,default){
  Gibi{
    int chapeau;
  }
}

Version(1){
  Gibi{
    int chapeau;
    float extraValue;
  }
}

Shadok{
  Table(Gibi, nbGibi) tabGibi;
}
```

Ici, deux versions de ***Gibi*** sont disponibles. Une version par défaut avec un seul attribut (version **default**), et une autre avec deux. Le format de données choisira automatiquement la bonne et produira le format demander par défaut.

La conversion entre deux classes générées est également possible :

```
Gibi{
  int chapeau;
}
Shadok {
  int age;
}
Converter(Shadok, Gibi){
  age = chapeau;
}
```

Dans ce cas, une classe ***Gibi*** est convertie en classe ***Shadok*** par la génération d'une fonction de conversion *plib_converter* qui complète la PLIBS_8. Cette fonctionnalité sera utile lors de reconstitution des événements (voir section 5.6.1).

3.5 Performances

Les tests de performances présentés dans cette section ne concernent que les vitesses de lecture et d'écriture du format de données généré pour tous les langages des back-end.

Le format de données utilisé pour les test de performances est le même que celui utilisé dans toute l'analyse de données. Il décrit les images des caméras des télescopes sous forme de tableaux. Un fichier d'observation contient un tableaux de télescopes. Un télescope contient un tableaux d'événements. Un événement contient le signal enregistré dans la caméra sous forme de tableaux d'entiers (int) alignés. Ce format de données contient évidemment d'autres données, qui ne seront pas détaillées ici, car elles représentent moins d'un pourcent du volume totale des données lues.

Les tests ont été effectués avec 17 fichiers. Le volume de chaque fichier est d'environ 600 MO ce qui représente un volume totale d'environ 10 GO.

Les tableaux 3.1 et 3.2 comparent ces vitesses de lecture et d'écriture au programme *dd*. Deux architectures différentes sont utilisée.

- La première est plus ancienne mais est plus représentative des machines utilisées pour le calcul. Elle ne possède pas de disque SSD et a une RAM standard.
- La seconde est plus récente, elle utilise l'extension AVX2 et a permis de d'évaluer les vitesses d'écriture sur disques SSD.

Les vitesses de lecture et d'écriture des formats de données C++ et wrapper Python sont supérieures à celle du programme *dd*. Car nous optimisons la taille de la mémoire tampon de la librairies standard du C, avec la fonction *setvbuf*, pour que les temps de chargements des fichiers en RAM soient plus courts. Le bus de lecture est donc particulièrement bien utilisé. Au contraire, le format de données en Python pur est particulièrement lent en lecture (5.3 fois plus lent que le format C++) et extrêmement lent en écriture (11.59 fois plus lent que le format C++). L'utilisation du format de données en wrapper Python donne des résultats comparables au C++.

	dd	Format C++	Pure Python	Wrapper Python
Lecture(RAM)	991 MO/s	1150 MO/s	214 MO/s	1233 MO/s
Lecture(RAM) écriture(disque dur)	65.51 MO/s	81.16 MO/s	7 MO/s	74.70 MO/s
Écriture(disque dur)	70.14 MO/s	87.32 MO/s	7, 48 MO/s	79.52 MO/s

TABLE 3.1 – Temps en lecture, écriture et lecture-écriture du format de données généré , C++, Python, wrapper Python comparé au programme *dd*. Ces tests de performances ont été réalisés sur un Intel core i5 M 560 utilisé avec une architecture SSE4.

	dd	Format C++	Pure Python	Wrapper Python
Lecture (RAM)	1494 MO/s	1861 MO/s	481.93 MO/s	2302.58 MO/s
Lecture(RAM) écriture(SSD)	120.68 MO/s	376.13 MO/s	11.39 MO/s	258.30 MO/s
Écriture (SSD)	131.29 MO/s	471.37 MO/s	11.66 MO/s	290.94 MO/s

TABLE 3.2 – Temps en lecture, écriture et lecture-écriture du format de données généré, C++, Python, wrapper Python comparé au programme *dd*. Ces tests de performances ont été réalisés sur un Intel core i5 M 560 utilisé avec une architecture AVX.

Notons le fait que le format de données en wrapper Python est légèrement plus rapide en lecture que le format C++. Soit 7% avec SSE4 et jusqu'à 23% avec AVX2. Nous pensons que cet effet est dû au fait que Python réutilise la mémoire qu'il a préalablement alloué ce qui accélère les allocations.

Les vitesses d'écriture sont plus intuitives. Le format de données wrapper Python est 7.9% plus lent que le format C++ avec SSE4 et plus de 31% plus lent avec AVX2.

Les tests sur différentes architectures montrent que des différences faibles de performances sur d'anciennes architectures, environs 7% de différence entre les vitesses de lecture et d'écriture sur SSE4, sont amplifiées par les nouvelles, plus de 20% d'écart entre les vitesses de lecture et d'écriture sur AVX2.

Les résultats concernant l'optimisation de calculs seront présentés dans le chapitre 5.

3.6 Conclusion

Ce chapitre montre la nécessité d'utiliser un format de données performant pour l'expérience CTA. Deux générateurs de format de données ont été développés pour ce travail. Le premier, devait être temporaire, afin de fournir un format de données performant au groupe CTA-LAPP et lui permettre de développer son analyse de données HPC. Une note interne, décrivant les propriétés d'un format de données HPC, découla de ce travail. En réponse à cette note, le responsable de l'analyse de données de CTA rédigea également une note où il y détailla les recommandations du format de données pour CTA. Un deuxième générateur de format de données fut créé dans le but de répondre complètement aux exigences, attendues par CTA.

Ce travail a montré que le problème des entrées/sorties de CTA dû au temps de lecture et d'écriture des formats de données précédents pouvait être résolu. De ce fait, l'optimisation de l'analyse de données de CTA pouvait être effectuée à l'aide de l'informatique hautes performances.

Ce travail a également permis de préciser les propriétés que devrait avoir le futur format de données de CTA. Actuellement, quatre formats de données, ou assimilés, sont envisagés par la collaboration :

- **Simtel** : le format de données historique des simulations.
- **ZFits** : un format compressé basé sur le format de Fits de la NASA.
- **HDF5** : le Hierarchical Data Format [50] est un outil bien adapté au HPC qui a été conçu pour manipuler des fichiers bien plus volumineux que la RAM (Random Access Memory) disponible sur une architecture. Il ne décrit pas un format de données à proprement parlé mais fournit un ensemble de fonctionnalités qui simplifies la gestion de fichiers binaires volumineux.
- **PDataGenerator** 1 et 2 : les générateurs de format de données que nous avons présenté dans ce chapitre. Pour l'instant, notre format de données généré n'est pas un standard, par définition. Un futur travail consistera à ajouter un opérateur pour générer un format de données basé sur HDF5.

La problématique d'un format de données efficace est commune à toutes les expériences à venir. CTA est ici un cas test, mais tout l'intérêt d'un générateur est de permettre son utilisation dans d'autres situations. Le groupe de recherche sur la simulation de plasma au sein du *Laboratoire Leprince-Ringuet*, LLR, utilise d'ores et déjà notre générateur ainsi qu'une partie des fonctions HPC de la ***PLIBS_8***.

L'ensemble des formats de données utilisés dans ce manuscrit ont été générés, via ce travail. De cette manière, l'effet des entrées/sorties est limité et le HPC peut être utilisé pour optimiser l'analyse de données de CTA. Il reste cependant un défi très important pour CTA que nous avons à peine évoqué dans ce chapitre, la compression des données. Celle-ci devra être rapide et efficace, deux aspects apparemment incompatibles. Le chapitre 4 décrit comment

nous avons abordé ce sujet tout en l'incorporant dans les générateurs de format de données que nous avons décris tout au long de ce chapitre.

L'essentiel du chapitre 3 –

Un format de données efficace est indispensable pour CTA car il permet d'utiliser le calcul haute performance. Deux générateurs de codes ont été développés dans ce but. Un premier fut conçu pour tester et explorer différentes techniques d'optimisation et de compression. Le second, bien plus avancé, lui succéda pour satisfaire les exigences de CTA concernant le format de données. Ce travail a mis en lumière l'importance d'un format de données efficace et contribua à l'affinement des objectifs du format de données de CTA. Les générateurs ainsi que les fonctions HPC de la ***PLIBS_8*** sont déjà utilisés par nos collègues du LLR qui travaillent sur la simulation de plasma.

Chapitre 4

Compression de données sans perte

Sujet du chapitre 4 –
Ce chapitre traite du défi de la compression des données dans les futures grandes expériences de physique et plus particulièrement le cas de CTA. La problématique est d'obtenir un algorithme de compression suffisamment performant en terme de taux de compression et de vitesse d'exécution.

Sommaire

Les expériences scientifiques actuelles produisent une grande quantité de données et les nouvelles en produiront d'avantage. Ce raz-de-marée de données représente quelques Péta-octets (PO) de données par an [51]. Mais cela n'équivaut qu'à une partie de la quantité produite par les grandes expériences radio telles que SKA, Alma ou LOFAR. L'expérience SKA produira à elle seule 2 TO/s [52]. Cette augmentation du flux de données cause des bouleversements techniques à tous les niveaux, comme le stockage des données, leur analyse, leur dissémination et leur préservation. Ce chapitre traite du problème de compression des données.

La manière classique d'acquérir les données en physique est de digitaliser un signal analogique en entiers non signés. Ces signaux sont en général dominés par un bruit blanc. Dans ce chapitre, nous proposons une approche pour compresser des données dominées par un bruit blanc dans un temps plus court que les algorithmes classiques avec un taux de compression raisonnable. D'autres expériences ont récemment résolues ce problème de compression de données [53], [39] mais pour de faibles flux de données. Avec l'augmentation du flux de données, l'amélioration de la vitesse et du taux de compression deviennent essentiels pour atteindre de bonnes performances.

Les meilleurs algorithmes de compression sont typiquement utilisés pour les images (JPEG), les vidéos (H264) ou la musique (MP3). Il permettent des taux de compression supérieurs à 10. Cependant, ils compressent en éliminant une partie des données (compression avec perte), ce qui n'est pas acceptable dans le cas où les données doivent être précises et intactes. Dans le meilleur des cas, une compression sans perte seule est nécessaire.

Dans le cadre de ce travail, la compression sans perte doit être utilisée. Il existe différentes méthodes de compression sans perte. Les plus courantes sont LZMA [49], LZ78 [54], BZIP2 [55], GZIP [56], Zstandard [57] ou l'algorithme d'Huffman car ils ont les meilleurs taux de compression. Cependant, leurs vitesses de compression imposent de lourdes contraintes vis à vis du volume de données considéré.

La compression de caractères telle que CTW (Context Tree Weighting) [58], LZ77 [59], LZW [60], la transformée de Burrows-Wheeler, ou PPM [61], ne peuvent pas être utilisées efficacement sur des données physiques puisqu'elles n'ont pas les mêmes caractéristiques, comme les occurrences ou les répétitions de caractères.

4.1 Situation de CTA

L'expérience CTA, la nouvelle génération d'observatoire d'astronomie gamma au sol, générera plus de 360 PO de données brutes non compressée d'ici 2030 (voir chapitre 2). CTA aura un site d'observation dans chaque hémisphère et représente plus de 120 télescopes au total. Une compression efficace des données est essentielle afin de les transférer des sites jusqu'aux centres de stockage en Europe.

CTA produira en fait 210 PO de données brutes par an, mais ce volume sera réduit à 4 PO par an pour les deux sites à cause de contraintes technologiques et budgétaires. Il y a donc un besoin à la fois de compression sans perte et de compression avec pertes afin de réduire les données et d'assurer une vitesse de lecture et d'écriture raisonnable.

La vitesse d'écriture doit être proche du temps réel puisqu'il n'y aura pas de mémoire tampon suffisamment grande sur site. La vitesse de décompression doit également être soutenue puisque la totalité des données accumulées sera réanalyser chaque année. Cela implique que la quantité de données qui sera lue sur disques (ou sur bandes) et décompressée augmentera tous les an (4 PO, 8 PO, 12 PO...).

Dans CTA comme dans beaucoup d'expériences quelque soit le domaine, les données acquises par digitalisation peuvent être décrites par deux composantes : une distribution poissonnière qui représente le signal, dominée par une distribution gaussienne représentant le bruit. Ce dernier est généralement du bruit blanc (voir figure 4.1).

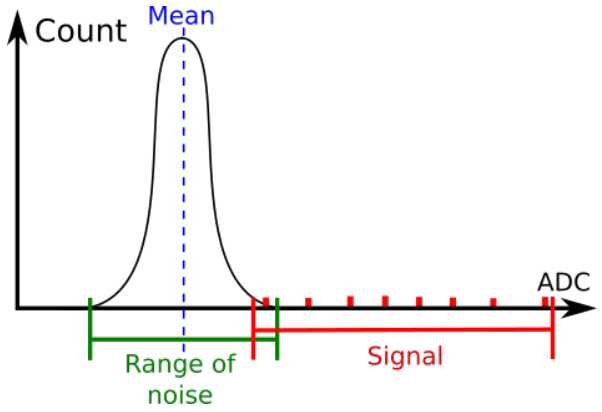


FIGURE 4.1 – Exemple de signal digitalisé dans de nombreuses expériences de physique. Dans la majorité des cas, le bruit blanc (une distribution gaussienne) domine de signal (généralement une distribution poissonnière). Donc, la plus grande partie du signal que l'on veut compresser suit une distribution gaussienne.

Ce chapitre propose une méthode de compression basée sur ces caractéristiques (distribution gaussienne ajoutée à une distribution poissonnière). En combinant cette méthode avec le meilleur algorithme de compression, comme LZMA, la vitesse finale de l'algorithme est fortement accélérée tout en gardant son taux de compression intact. Sa vitesse et son taux de compression en font un candidat compétitif pour la compression de données scientifiques dans des cas réels d'utilisation.

4.2 La compression polynomiale

L'intervalle de valeur d'un unsigned int, $[\![0, 2^{32}[\![$, définie un ensemble mathématique $\mathbb{Z}/d\mathbb{Z}$, appelé anneau, où $d = 2^{32}$. Dans ce cas, le signal digitalisé définie également un anneau, sa valeur minimale est v_{min} et sa maximale est v_{max}, donc l'anneau correspondant est définie comme $\mathbb{Z}/b\mathbb{Z}$ avec $b = v_{max} - v_{min} + 1$.

Dans la grande majorité des cas $b < d$, il est donc possible de stocker plusieurs valeurs dans le même unsigned int (voir figure 4.2). Une approche polynomiale peut être utilisée pour effectuer cette compression. Ajouter différentes valeurs dans le même entier puis les recalculer grâce aux puissances d'une base, qui est donnée par l'intervalle des valeurs à compresser.

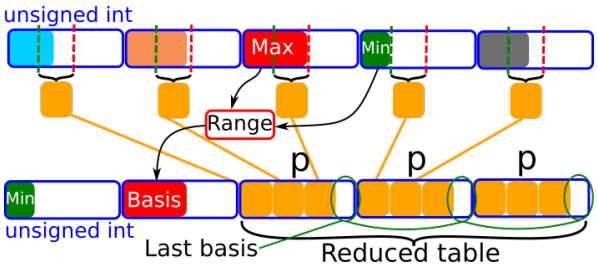


FIGURE 4.2 – Illustration du principe de la compression de données.

4.2.1 Méthode de compression basique

Soit un vecteur de données comportant N éléments, $\mathbf{v} \in \mathbb{N}^N$, son minimum, v_{min} et son maximum v_{max} associés. Si l'anneau que représente les données est plus petit que celui d'un unsigned int, il est possible de stocker plusieurs valeurs dans un seul unsigned int. Plus la base est petite et plus le taux de compression est important. Comme les données sont comprises dans l'intervalle $[\![v_{min}, v_{max}]\!]$, l'intervalle entre 0 et v_{min} est inutile. Grâce à cela, les données a compresser peuvent être réduites en leur soustrayant cette valeur minimale et ainsi former une base plus petite. Le minimum peut être stocké qu'une seule fois avant les données compressées. La base de compression B est définie par : $B = v_{max} - v_{min} + 1$. Elle permet de décrire $(v_{max} - v_{min})$ valeurs différentes. Le taux de compression p, est donné par le nombre de bases B qui peuvent être stockées dans un seul unsigned int (dans $[\![0, 2^{32}[\![$) :

$$p = \left\lfloor \frac{\ln\left(2^{32} - 1\right)}{\ln B} \right\rfloor \tag{4.1}$$

Les éléments compressés sont donnés par :

$$s_j = \sum_{i=1}^{p} v_{i+p\times(j-1)} \times B^{i-1} \quad \text{for } 1 \leq j < \frac{n}{p} \tag{4.2}$$

Une division de polynôme permet de décompresser les données.

4.2.2 Méthode de compression avancée

L'inconvénient de la méthode de compression basique est qu'il y a systématiquement un espace inutilisé à la fin de chaque unsigned int (voir figure 4.2). Le cas idéal utilise absolument tout l'espace de stockage disponible de sorte qu'il n'y ait plus de vide, comme l'illustre la figure 4.3. La méthode de compression avancée permet de tendre vers ce cas idéal. De plus, raccourcir le temps de lecture et d'écriture permet des vitesses de compression et de décompression supérieures.

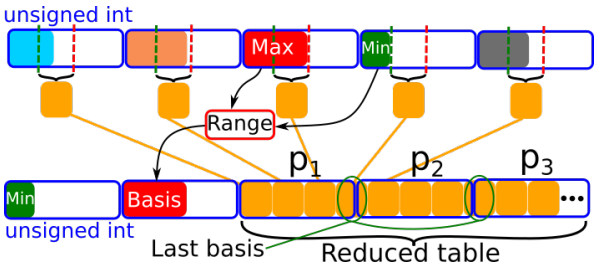


FIGURE 4.3 – Illustration du principe de compréssion avancé.

Un meilleur taux de compression peut être atteint en divisant la dernière base (voir figure 4.2 et figure 4.3), utilisée pour paquer des données dans le même unsigned int, en créant dans deux autres bases, R et R' de sorte à avoir $B \leq R \times R'$. Dans ce cas, la base R est stockée dans l'unsigned int courant et la base R' est stockée dans le prochain (voir figure 4.4). Cette configuration assure un ordre des données plus efficace en terme du préchargement de données CPU à l'étape de décompression, pour permettre une décompression plus rapide.

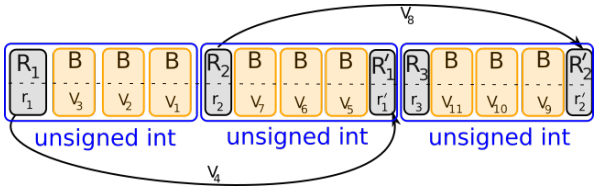


FIGURE 4.4 – Cette figure montre comment les valeurs du vecteur **v** sont stockées dans le vecteur compressé. La première ligne décrit les bases utilisées pour stocker les différentes valeurs. La seconde ligne montre les variables utilisées pour stocker ces différentes valeurs en prenant en compte leurs bases respectives. Pour augmenter le taux de compression, la dernière base d'un élement compressé, B, est découpée en bases R et R' cela permet d'améliorer la capacité de stockage d'un unsigned int au mieux. Les valeurs à gauche d'un unsigned int sont stockées avec une petite puissance de la base B. Les valeurs à droite d'un unsigned int sont stockées avec une grande puissance de la base B.

Les données sont accumulées de la plus grande puissance de la base B à la plus petite. Cela garantie que la décompression produira des données contigües.

Le scindement d'une valeur à compresser, v, en deux est réalisé en découpant la base de départ B en deux bases, R et R' et la valeur v en deux variables r et r'. Les variables r et r' sont stockées dans deux éléments compressés consécutifs (unsigned int).

Le calcul des bases R et R' n'est possible que lorsque le nombre de bases B pouvant être stockées dans un unsigned int est connu. Le nombre de bases qui peuvent être stockées dans le premier élément compressé unsigned int, p_1, est donné par l'équation suivante :

$$p_1 = \left\lfloor \frac{\ln\left(2^{32}-1\right)}{\ln B} \right\rfloor \tag{4.3}$$

La première base scindée R_1 est donnée par :

$$R_1 = \left\lfloor \frac{2^{32}-1}{B^{p_1}} \right\rfloor \tag{4.4}$$

La base R_1 doit être complétée pour stocker un élément $e \in [\![0, B[\![$, par :

$$R'_1 = \left\lfloor \frac{B}{R_1} \right\rfloor + (1 \text{ si } B \mod R_0 \neq 0) \tag{4.5}$$

Alors $R_1 \times R'_1 \geq B$. Toutes les bases R_i et R'_i sont associées pour stocker des valeurs r_i et r'_i respectivement. Le premier élément compressé, s_1, peut être écrit comme suit :

$$s_1 = r_1 + R_1 \times \left(\sum_{k=1, k\neq p_1}^{p_1} v_k B^{p_1-k} \right) \tag{4.6}$$

Où :

$$r'_1 = \left\lfloor \frac{v_{p_1}}{R_1} \right\rfloor \tag{4.7}$$

$$r_1 = v_{p_1} - r'_1 \times R_1 \tag{4.8}$$

La valeur r_1 est associée à la base R et la valeur r'_1 est associée à la base R'.
Le nombre de base B qui peuvent être stockées dans le deuxième élement compressé, p_2, est donné par :

$$p_2 = \left\lfloor \frac{\ln\left(2^{32}-1\right) - lnR'_1}{\ln B} \right\rfloor \tag{4.9}$$

La base partielle restante R_2 peut être écrite comme :

$$R_2 = \left\lfloor \frac{2^{32}-1}{R'_1 B^{p_2}} \right\rfloor \tag{4.10}$$

La valeur du deuxième élément compressé peut être calculé comme suit :

$$\begin{aligned} s_2 &= r_2 \\ &\quad + R_2 \times \left(\sum_{k=1,k\neq p_1}^{p_1} v_{p_0+k} B^{p_1-k} + r'_1 B^{p_1} \right) \end{aligned} \tag{4.11}$$

L'équation 4.5 peut être utilisée pour calculer R'_2.

La compression d'un vecteur complet de données peut être effectuée en utilisant une série mathématique pour calculer les bases scindées de chaque élément compressé. En fixant la première base scindée $R'_0 = 1$ à la première étape, les séries mathématiques utilisées pour compresser un vecteur d'entiers non signés peuvent être écrites comme suit (pour $0 < i \leq n_p$, où n_p est le nombre d'éléments compressés) :

$$\begin{aligned} p_i &= \left\lfloor \frac{\ln\left(2^{32}-1\right) - lnR'_{i-1}}{\ln B} \right\rfloor \\ R_i &= \left\lfloor \frac{2^{32}-1}{R'_{i-1}B^{p_i}} \right\rfloor \\ R'_i &= \left\lceil \frac{B}{R_i} \right\rceil \\ q_i &= i - 1 + \sum_{k=1,k\neq i}^{i} p_k \quad \text{ou 0 si } i = 0 \\ r'_i &= \left\lfloor \frac{v_{q_i+p_i}}{R_i} \right\rfloor \\ r_i &= v_{q_i+p_i} - r'_i \times R_i \\ s_i &= r_i + R_i \times \left(\sum_{k=1,k\neq p_i}^{p_i} v_{q_i+k+1} B^{p_i-k} + r'_{i-1} B^{p_i} \right) \end{aligned}$$

Où p_i est le nombre de bases B qui peuvent être stockées dans le i$^{\text{éme}}$ élément compressé, R_i et R'_i sont les bases scindées, r_i et r' leur valeurs associées respectives, q_i est utilisé pour savoir combien d'éléments ont été compressés depuis le i$^{\text{éme}}$ élément compressé, finalement s_i est la valeur du i$^{\text{éme}}$ élément compressé.

4.2.3 Méthode de compression par blocs

Nous obervons que plus l'étalement du signal est faible et plus de taux de compression est important. La méthode de compression avancée est particulièrement efficace pour compresser du bruit blanc avec un faible étalement. La conséquence directe de cela est que si le bruit gaussien ou le signal poissonien est trop étalé, l'efficacité de la compression diminue.

Son efficacité peut cependant être améliorée en divisant le vecteur à compresser en plus petits ensembles pour diminuer l'impact de la plus grande valeur sur le taux de compression global.

L'efficacité de cette méthode, et la détermination de la taille des blocs seront discutées dans la section 4.3.1.

4.3 Éxpériences et analyses

Afin de tester et d'évaluer les performances de la méthode de compression polynomiale proposée, des distributions aléatoires seront utilisées dans ce qui suit. Ces distributions sont en accord avec les valeurs simulées sur des caméras Cherenkov de CTA(voir figure 4.5).

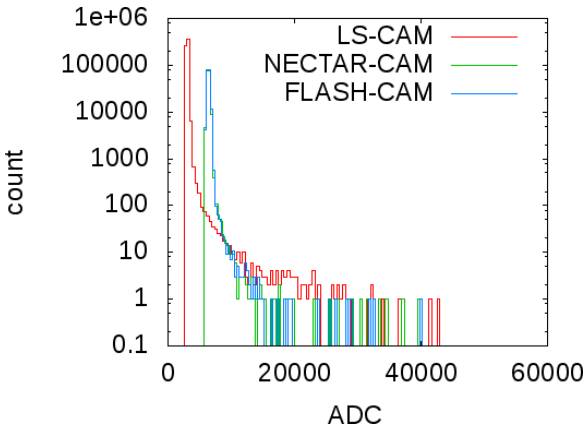


FIGURE 4.5 – Illustration de la distribution typique de signal obtenue dans quelques caméras utilisées dans CTA [62].

4.3.1 Compression sur des distributions simulées

Les données peuvent être décrites par une distribution aléatoire gaussienne avec un écart type donné (le bruit blanc dans le signal des caméras) et en lui ajoutant une distribution uniforme dans l'intervalle du signal de la caméra (le signal physique).

Soit un ensemble, $\mathcal{A}$, de distribution de valeurs des pixels des caméras :

$$\mathcal{A}(\mu, \sigma, x, y, a, N) = \mathcal{N}(\mu, \sigma)^{N-a} \cup \mathcal{U}(x, y)^{a} \tag{4.12}$$

Où :
— μ : la moyenne du bruit gaussien
— σ : l'écart type du bruit gaussien
— (x, y) : l'intervalle de la distribution de valeurs selon un loi uniforme
— a : le nombre de valeurs tirées suivant cette loi uniforme (signal)
— N : le nombre total d'éléments dans le vecteur à compresser

($\mathcal{N}$ est la distribution normale, le bruit simulé, et $\mathcal{U}$ décrit une distribution uniforme, le signal simulé.)

Un exemple de simulation est présenté dans la figure 4.6.

Nous avons testé les distributions $\mathcal{A}$ avec $\mu = 3000$, $\sigma \in [100, 10\,000]$, $x = 2\,000$, $y \in [\![20\,000, 100\,000]\!]$, $a \in [\![1, 1\,000]\!]$ et $N \in [\![1855, 10\,000]\!]$, ce qui correspond aux distributions que l'on attend dans les données de CTA.

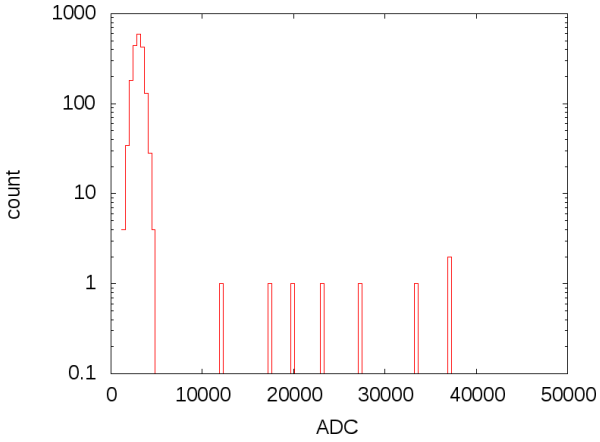


FIGURE 4.6 – Distribution typique utilisée pour tester la compression de données dans l'ensemble $\mathcal{A}\,(3\,000, 500, 2\,000, 45\,000, 9, 1855)$. Dans ce cas, la distribution gaussienne représente 99% des valeurs des pixels et la distribution uniforme ne représente qu'un pourcent de cette distribution.

L'implémentation de la compression polynomiale par blocs a été testé sur différentes distributions. Ce test détermine l'influence de la distributions des paramètres sur le taux de compression.

Comme la méthode de compression polynomiale utilise les propriétés statistiques des données pour les compresser, les tests ne peuvent être effectués que sur des ensembles de distributions.

La figure 4.7 montre le taux de compression pour 1000 vecteurs avec $\mathcal{A}\,(3000, 500, 2000, 45\,000, 4, 1855)$ pour calculer les fluctuations (courbe rouge). La variation de l'écart type, σ, de la gaussienne a une forte influence sur le taux de compression final, de l'ordre de 25% dans le cas où $\sigma = 1000$ à $\sigma = 500$ pour la meilleure taille de bloc. L'influence de l'intervalle du signal est plus faible, 5% ou 10% suivant la taille des blocs utilisés, et 5% pour la meilleure taille de bloc. Le choix de la taille des blocs est également important, le taux de compression est moindre si les blocs sont trop grands, le taux de compression diminue de 25% pour $\sigma = 500$ et de 30% pour $\sigma = 1000$. Dans ce cas, utiliser des blocs de 154 éléments permet un taux de compression de 2.47054 ce qui est 17% plus important que le taux de compression atteint avec la méthode basique.

La figure 4.8 montre que le rapport signal sur bruit a une forte influence sur le taux de compression si l'étalement des valeurs est faible. Au contraire, si l'étalement du bruit est important (courbe verte), le rapport signal sur bruit a une influence plus faible sur le taux de compression final.

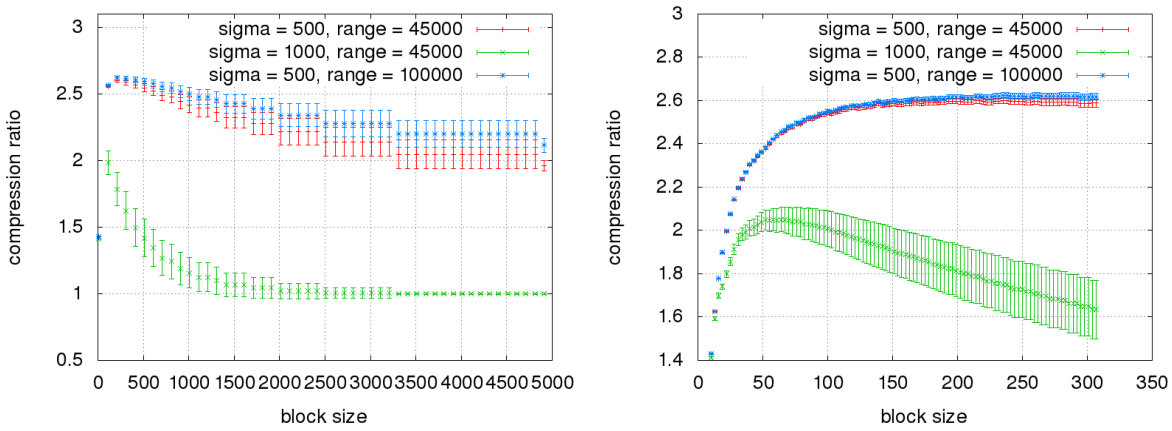


FIGURE 4.7 – **À gauche** : illustration du taux de compression en fonction du nombre d'éléments par bloc utilisé pour compresser un vecteur de données Les points rouges (+) indiquent le taux de compression pour une distribution avec $\sigma = 500$ et $range = 45\,000$, dans $\mathcal{A}(3\,000, 500, 2\,000, 45\,000, 4, 10\,000)$. Les points bleus (∗) donnent le taux de compression pour une distribution avec $\sigma = 500$ et $range = 100\,000$, dans $\mathcal{A}(3\,000, 500, 2\,000, 100\,000, 4, 10\,000)$. Les points verts (×) montrent le taux de compression pour une distribution avec $\sigma = 1\,000$ et $range = 45\,000$, dans $\mathcal{A}(3\,000, 1\,000, 2\,000, 45\,000, 4, 10\,000)$. Les valeurs obtenues à droite donnent le taux de compression de la méthode de compression polynomiale avancée (un seul bloc). **À droite** : le même graphique zoomé.

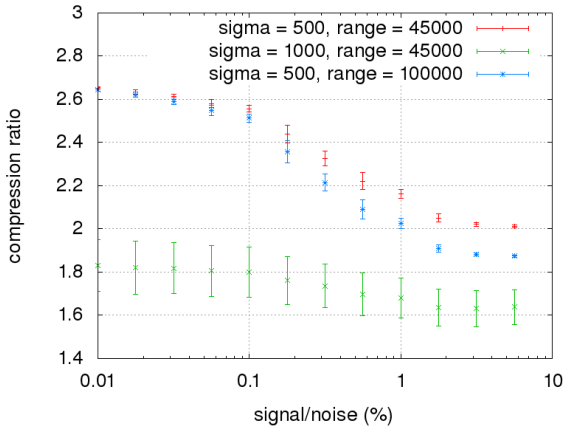


FIGURE 4.8 – Illustration de l'influence du rapport signal sur bruit sur le taux de compression final pour un vecteur de 10 000 valeurs. Les points rouges (+) donnent le taux de compression pour une distribution avec $\sigma = 500$ et $range = 45\,000$. Les points bleus (∗) montrent le taux de compression pour une distribution avec $\sigma = 500$ et $range = 100\,000$. Les points verts (×) indiquent le taux de compression pour une distribution avec $\sigma = 1\,000$ et $range = 45\,000$. Le taux de compression est constant à partir de 10%.

4.3.2 Compression polynomiale sur des simulations CTA

La section 4.3.1 décrit les taux de compression obtenus avec la compression polynomiale par blocs sur des distributions simulées. Cependant, ce test ne reflète par fidèlement les taux de compression potentiellement obtenues avec des données physiques qui sont généralement des superpositions de plusieurs distributions dans le cas où plusieurs capteurs sont utilisés.

Nous avons testé cette compression sur les données de CTA. Les données de CTA décrivent de la lumière émise par des gerbes électromagnétiques captée par des caméras fixées sur des télescopes au sol. Seules les images des télescopes qui ont déclenchés sont sauvegardées. Chaque télescope produira un fichier contenant ses propres images (appelés aussi événements). Un événement est composé d'une en-tête, utilisée pour décrire des propriétés comme l'horodatage, et aussi des données caméra : le signal intégré et/ou le signal vidéo (waveform). Cela représente typiquement quelques dixaines de millier d'octets. Une sélection des pixels sera probablement appliquée pour diminuer le flux de données ce qui constitue une première compression avec perte.

Comme décrit dans le chapitre 3, le futur format de données de CTA devra avoir plusieurs propriétés. Tous les fichiers devront pouvoir être lus par partie et permettre l'accès à leur en-têtes sans nécessiter une étape de décompression complète. Donc, la compression par blocs est permise et la compression d'un fichier complet est a priori rejetée. Ce qui est en faveur de l'approche proposée ici.

Ce test a été effectué avec les fichiers Monte-Carlo de CTA production 3 [63] qui simulent l'observation de gerbes Cherenkov dans l'atmosphère. Dans ce cas, les capteurs sont des caméras Cherenkov. Ces simulations sont utilisées pour évaluer les résultats scientifiques de CTA dans des conditions idéales, elles sont donc raisonnablement réalistes. Elles ont été converties dans un format de données qui permet des optimisations multi-niveaux (voir chapitre 3), qui stocke les valeurs des pixels en 16 bits pour permettre une compression rapide. Les fichiers de test concernent 624 télescopes et contiennent 22 000 images, soit 7.6 GO de signal vidéo (voir chapitre 2) et 474 MO de signal intégré [1]. Chaque image est composée d'un bruit et d'un signal qui ressemble à une ellipse (voir figure 4.9).

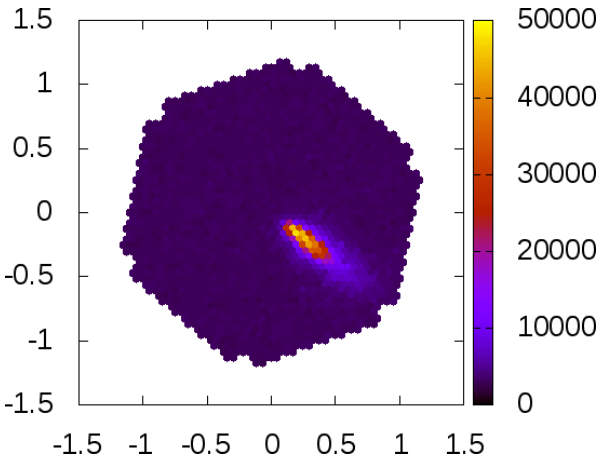


FIGURE 4.9 – Illustration de la forme ellipsoïdale des gerbes enregistrées dans la caméras des simulations Monte-Carlo. L'échelle de couleurs représente le nombre de photons détectés dans un pixel.

La suite présente comment adapter la compression polynomiale au prérequis de CTA. Tous les tests on été effectués avec un Intel core i5 ayant une fréquence d'horloge de 2.67 GHz avec l'extension SSE4 sans disque SSD.

1. Le signal intégré s'obtient en sommant les valeurs du signal vidéo à chaque temps.

4.3.2.1 Test sur les données vidéo de CTA

Pour améliorer la compression du signal vidéo et permettre l'exploitation des techniques du calcul haute performance (le préchargement des données CPU et la vectorisation) nous stockons les données dans des matrices. Un élément $M_{i,j}$ d'une matrice correspond au i^{éme} temps du j^{éme} pixel de la caméra courante. L'alignement sur les lignes permet une meilleure compression car cela augmente le nombre de séquences de valeurs similaires. Cette configuration permet également d'optimiser le traitement des images (voir chapitre 5).

Notre compression polynomiale réduit un fichier test de 7.6 GO en un fichier compressé de 2.8 GO (taux de compression de 2.71) en 5 min 36 s. Le tableau 4.1 compare nos résultats avec les méthodes classiques. Nous testons les algorithmes BZIP2 et LZMA sur les fichiers complets.

Notre méthode est 3.5 fois plus rapide que BZIP2 et obtient un meilleur taux de compression.

Les deux tests de l'algorithme LZMA utilisent le programme *7z*.

	Compression taux	Compression Temps	Taille (GO)	Décompression Temps	Compression Temps RAM	Décompression Temps RAM
Sans compression	1	0	7.6	0	0	0
Polynomial avancée	2.71	5 min36.025s	2.8	2 min35 s	2 min56 s	2 min30 s
BZIP2	2.62	19 min 18.247 s	2.9	2 min23.676 s	-	-
LZMA (-mx=9 -mfb=64 -md=200m)	6.52	2 h14 min44 s	1.166	1 min49.689 s	2 h03 min44 s	2 min03 s
LZMA (-mx=1 -mfb=64 -md=32m)	5.88	14 min00 s	1.293	2 min10 s	15 min42 s	2 min09 s
Poly. + LZMA (9 16 32)	5.02	10 m49 s	1.513	2 min13 s	4 min57 s	2 min15 s

TABLE 4.1 – Taux et temps de compression de l'algorithme de compression polynomiale comparé à LZMA (état de l'art dans le domaine). Les fichiers de tests sont des simulations contenant la totalité du signal vidéo de la production PROD_3 (run 3998) de l'expérience CTA. Le processeur utilisé est un Intel core i7 M 560 avec 19 GO de RAM installé sous Fedora.

Nous avons tout d'abord chercher à obtenir le meilleur taux de compression possible avec la commande 7z a -t7z -m0=lzma -mx=9 -mfb=64 -md=200m -ms=on. Ainsi, le fichier de test est compressé avec un taux de compression de 6.52 en 2 h14 min 44 s. Nous avons également testé la compression rapide de LZMA (7z a -t7z -m0=lzma -mx=1 -mfb=64 -md=32m -ms=on) et nous avons obtenus un taux de compression de 5.88 en 14 min 00 s.

Finalement, nous obtenons un taux de compression de 5.02 en seulement 10 min 49 s en combinant la compression polynomiale et le LZMA. Les temps de décompression des différents algorithmes sont très similaires. Cependant, la compression et la décompression d'un fichier complet ne sont pas en accord avec les recommandations du format de données (les événements ou les blocs doivent être compressés séparément des en-têtes).

Une solution consiste à compresser plusieurs événements dans des blocs, le niveau de granularité est alors augmenté. Avec LZMA, cette méthode peut atteindre un taux de compression de 6 en compressant 300 événements consécutifs par bloc, ce qui représente moins de 0.02 seconde de signal d'une caméra LST-CAM. Dans ce cas, chaque bloc contient approximativement 100 MO de données (suivant la caméra). Pour réaliser cette étude, nous

avons utiliser les propriétés de compression fournies par le nouveau générateur de format de données (voir section 3.3.2.5).

4.3.2.2 Test sur les données intégrées de CTA

Les données intégrées sont obtenues par réduction du signal vidéo. Cette réduction peut être faite sur l'ensemble des pixels contenant du signal vidéo ou seulement sur une partie. Dans notre cas, nous réduisons les matrices de la section 4.3.2.1 en vecteurs pour utiliser le HPC. Les éléments des vecteurs produits sont décrits en 24 ou 32 bits suivant les caméras. La compression polynomiale atteint un taux de compression de 3.74 en 3.7 s sur un fichier test de 474 MO.

La figure 4.10 montre les différents taux de compression obtenus avec différentes tailles de blocs. Les variations du graphique montrent les différents taux de compression obtenus pour les différentes caméras dans le fichier. Statistiquement toutes les caméras n'ont pas le même taux de compression. Cela explique les fluctuations.

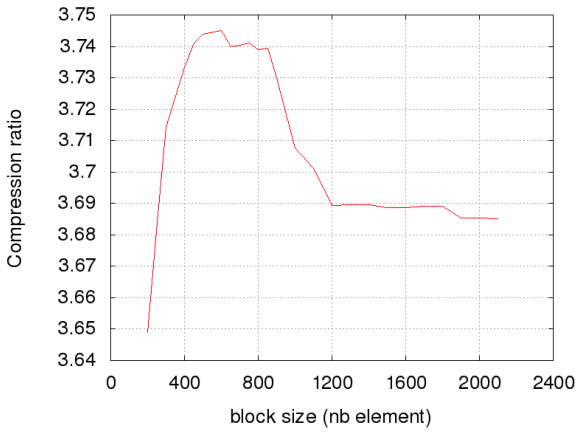


FIGURE 4.10 – Comparaison du taux de compression pour differentes tailles de blocs sur des fichiers Monte-Carlo de CTA PROD_3.

La différence entre les données Monte-Carlo et les distributions simulées vient des 7 types de caméras. Chaque type de caméra a une taille typique d'ellipse. Si un bloc contient la totalité de l'ellipse, il sera moins compressé que les autres. Il en résulte un meilleur taux de compression global.

Le tableau 4.2 compare la compression polynomiale avec les algorithmes LZMA et BZIP2.

	Taux de compression	Temps	Taille (MO)	Temps en RAM	Temps de Décompression en RAM
Sans compression	1	0	474	0	0
Compression polynomiale avancée	3.74	3.7 s	127	0.9 s	0.9 s
BZIP2	4.69	1 min 48 s	101	1 min0 s	6.23 s
LZMA (7z)	4.84	7 min 48.636 s	98	1 min 18 s	9.28 s
Compression polynomiale avancée + LZMA	4.84	**24.646 s**	**98**	1 min 20 s	11 s

TABLE 4.2 – Temps et taux de compression de la compression polynomiale ainsi que la taille des fichiers produits comparés à LZMA (état de l'art). Le fichier de simulation testé provient de la PROD_3 (run 497) de l'expérience CTA. La combinaison de notre méthode de compression polynomiale avancée avec LZMA permet d'obtenir le même taux de compression que le LZMA pur mais 19 fois plus rapidement. Le CPU utilisé est un Intel core i5 M 560 avec 8 GO de RAM installé avec un Ubuntu 16.4.

L'algorithme BZIP2 atteint un meilleur taux de compression (4.69) mais en 1 min48 s (29 fois plus lent que notre compression). Les meilleurs taux de compression sont obtenus avec l'algorithme LZMA (4.84), mais en 7 min48 s (126 fois plus lent que notre compression).

En combinant notre compression polynomiale avancée avec LZMA nous obtenons le même taux de compression que LZMA seul, mais 19 fois plus rapidement. De plus, l'utilisation de la compression polynomiale permet au LZMA de garder un profil plat car elle rassemble des petites valeurs pour en produire des grandes. Donc, la moyenne des valeurs augmente et la distribution de l'occurrence des octets s'aplatie (voir section 4.4).

La compression polynomiale atteint un meilleur taux de compression qu'un classique décalage de bits car l'espace libre dans un unsigned int est utilisé (voir figure 4.11). Les données à compresser sont encodée sur 32 bits. La compression polynomiale avancée obtient un bien meilleur taux de compression qu'un décalage de bits classique car elle sommes des valeurs et donc mutualise l'utilisation des bits de stockage. Elle utilise 14 630 bits dans notre exemple, contre 16 900 pour le décalage de bits, soit 13.4% de moins.

En extrapolant au flux de données annuel de CTA, 4 PO [64], l'utilisation de l'algorithme LZMA nécessiterait plus de 1750 coeurs par an juste pour la compression.

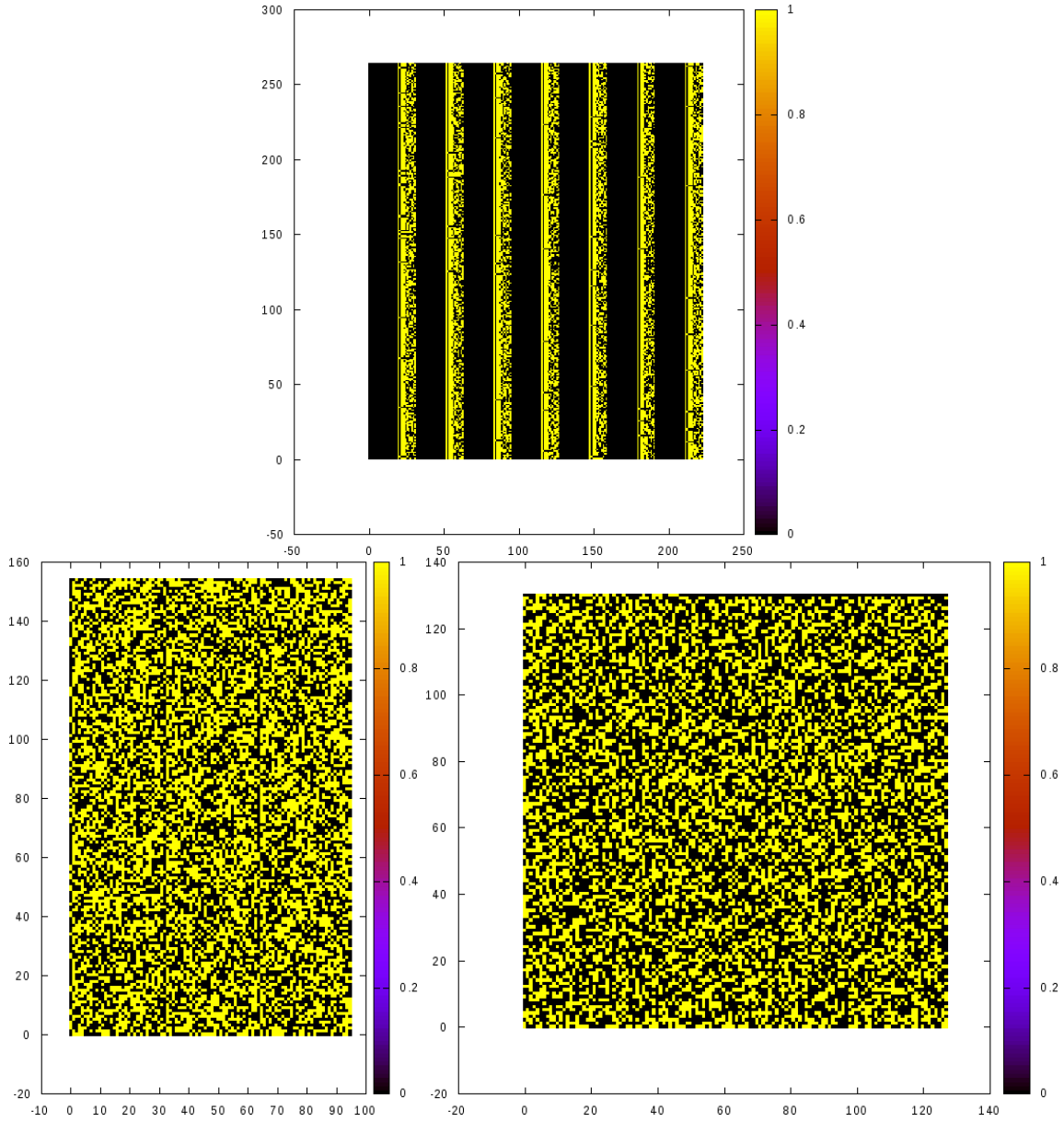


FIGURE 4.11 – Comparaison de l'agencement des données, en bits, d'un événement d'un fichiers Monte-Carlo de CTA PROD_3. **En haut**, les données non compressées, sur 32 bits. **En bas à gauche**, résultat de la compression polynomiale avancée. **En bas à droite**, compression par un décalage de bits classique.

4.4 Occurrence d'octets

Un test supplémentaire est de visualiser la distribution des différentes valeurs des octets présents dans le fichier initial et dans le fichier compressé. De ce point de vue, le but principal d'un algorithme de compression est d'assurer que l'occurrence de chaque octet est la même, à ce niveau le fichier ne peut pas être compressé d'avantage sans perte [49]. La figure 4.12 montre les différentes distributions d'octets dans le fichier initial et ceux compressés avec la compression polynomiale, LZMA et BZIP2 pour des données intégrées.

Cette figure montre que l'algorithme de réduction polynomiale est bien une compression. En fait, la compression polynomiale aplatie le profile de la distribution. La meilleure compression est obtenue avec l'algorithme LZMA sur un fichier qui a été préalablement compressé avec la compression polynomiale car son profile est plat. La combinaison de la compression polynomiale avec l'algorithme BZ2 ne permet pas d'obtenir un meilleur taux de compression ou une compression plus rapide.

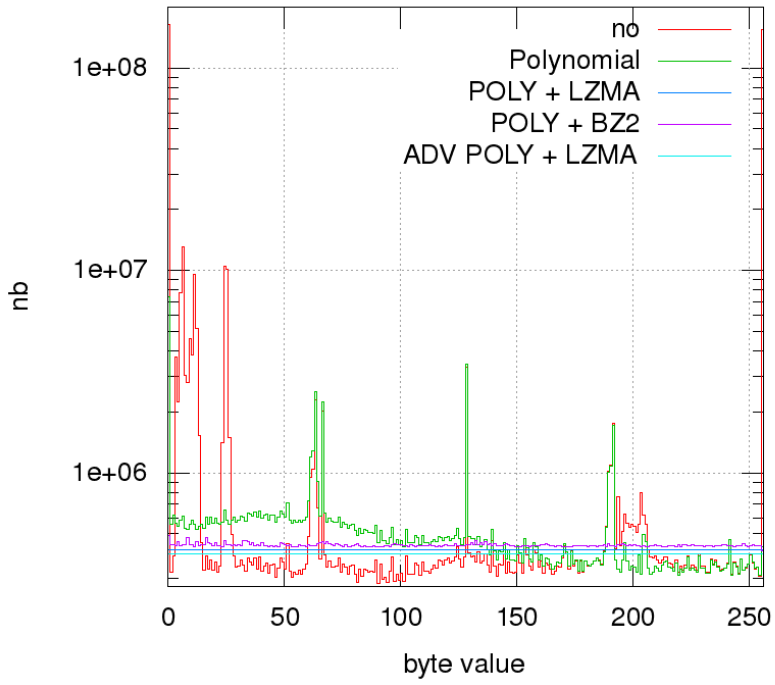


FIGURE 4.12 – Comparaison des différentes occurrences d'octets dans les fichiers binaires. En rouge, le profile du fichier décompressé (Monte-Carlo PROD_3 de CTA). En vert, le profile du fichier compressé avec la compression polynomiale. En bleu, le profile du fichier compressé correspondant avec l'algorithme LZMA. En violet, le profile du fichier compressé correspondant avec l'algorithme BZ2. En cyan, le profile du fichier compressé avec la compression polynomiale avancée et LZMA.

4.5 Conclusion

Ce chapitre introduit un nouvel algorithme de compression d'entiers non signés provenant d'un signal digitalisé et dominé par un bruit blanc.

Cette méthode est très rapide, elle aide le préchargement des données en mémoire et facilite la vectorisation. Elle peut être intégrée dans chaque format de données qui utilise des tableaux ou des matrices d'entiers. Cette méthode compresse préférentiellement des entiers, mais une adaptation de cet algorithme pourrait compresser des nombres flottant avec une précision fixe qui créée des entiers. Elle permet de compresser des matrices et des tableaux séparément de sorte à garder une structure de données similaire entre les données compressées et décompressées. Elle est intégrée, ainsi que LZMA, comme options de compression dans les générateurs de format de données présentés dans le chapitre 3. Le temps de décompression est typiquement deux fois moindre que le temps de compression. Cette méthode peut aussi être vectorisée afin d'augmenter sa vitesse de traitement des données.

Les tests sur les Monte-Carlo CTA montrent que la compression polynomiale est moins efficace que LZMA mais meilleure que BZIP2 sur les données vidéo. Utilisée comme pré-compression, elle accélère l'algorithme LZMA et réduit le temps de compression. Plus généralement, elle peut-être considérée comme une méthode de pré-conditionnement pour les algorithmes de compression comme LZMA.

La compression est très efficace et rapide sur les données intégrées. Utilisée comme pré-compression de LZMA, nous obtenons le même taux de compression que LZMA pur mais avec un temps d'exécution 19 fois plus court.

La simplicité de la méthode permet un développement aisé dans de multiples langages et la possibilité de l'utiliser sur des systèmes embarqués. Elle peut également réduire le volume de données produit par des capteurs ultra sensibles à l'aide de FPGA. Elle peut être utilisée comme méthode de pré-compression d'algorithmes tels de LZMA et les accélérer.

Ce travail a fait l'objet d'une publication [65] dans le journal *Computing and Software for Big Science*.

L'essentiel du chapitre 4 –

La compression des données est un des nombreux défis de CTA. Nous avons présenté dans ce chapitre une méthode originale de compression sans perte à la fois rapide et permettant d'obtenir un taux de compression raisonnable. Elle peut également accélérée des méthodes existantes, comme LZMA (jusqu'à un facteur 19 dans notre cas), tout en conservant un taux de compression élevé. Sa simplicité en fait également un candidat de choix pour la compression de données sur FPGA.

Chapitre 5

Optimisation de l'analyse standard Hillas

Sujet du chapitre 5 –
Ce chapitre montre comment optimiser l'analyse de données la plus simple de CTA. Elle consiste à calculer des paramètres d'ellipses à partir des images des caméras puis de les combiner pour déterminer les paramètres physiques de la particule initiale qui a créée la gerbe de particules.

Sommaire

Lorsqu'un rayon cosmique arrive sur Terre, il interagit avec l'atmosphère en produisant une gerbe de particules. Ces particules sont plus rapides que la lumière dans l'air. Elles émettent donc de la lumière Cherenkov.

Plusieurs télescopes enregistrent simultanément des images ou des vidéos des gerbes Cherenkov (voir figure 5.1). Cette simultanéité permet de reconstruire les événements en stéréoscopie.

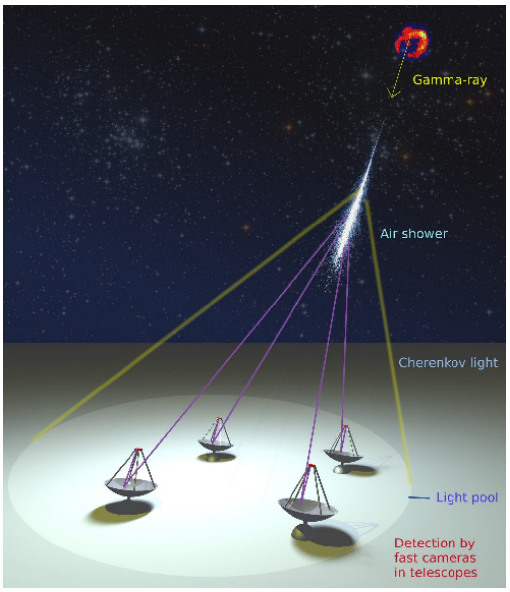


FIGURE 5.1 – Détection d'une gerbe Cherenkov par des télescopes en stéréoscopie.

Les photomultiplicateurs (PM) qui composent la caméra assurent la rapidité et la sensibilité de l'instrument. Ils sont conçus pour être sensibles à un seul photon. Cette propriété peut saturer leur signal de sortie (voie haut gain). Pour évité cela, il existe généralement une autre voie, moins amplifiée, la vois bas gain.

Un PM renvoie toujours du signal, même si il ne reçoit aucun photon. Ce bruit électronique est très variable dans le temps et sa moyenne est appelée piédestal. Le signal électrique émis lorsque le PM ne reçoit qu'un seul photon est appelé le gain du PM. La voie haut gain a une meilleure sensibilité que la voie bas gain mais son signal peut être saturé une fois digitalisé. Dans ce cas, la voie bas gain est utilisée à la place de la voie haut gain.

Seule une partie des photons qui entrent dans le PM sont détectés. Ce ratio est la sensibilité du PM.

Un PM est décrit par son gain, son signal, son piédestal et sa sensibilité pour chacune des voies haut et bas gain. Nous les appellerons pixels dans le reste de ce manuscrit.

Certains pixels peuvent avoir des défauts. Des problèmes d'alimentation ou d'usure causés par le sable ou les hautes températures. Un pixel peut être saturé, son gain ou son piédestal ont pu ne pas avoir été déterminés. Dans ce cas, le pixel est marqué comme étant inutilisable (broken pixel) et est temporairement exclu de l'analyse de données.

Si le signal dans la caméra est suffisant, il est enregistré dans un fichier de données brutes. Si plusieurs télescopes déclenchent en même temps, les événements sont également sauvegardés dans un fichier de données brutes. Les événements composés de plusieurs télescopes sont importants car ils permettent une analyse stéréoscopique. Dans ce cas, les gerbes de particules sont enregistrées selon plusieurs points de vue.

L'analyse de données consiste à utiliser les images enregistrées de différents télescopes afin de déterminer les paramètres physiques de la particule qui a créée la gerbe (voir figure 5.2). Deux types d'acquisitions existent.

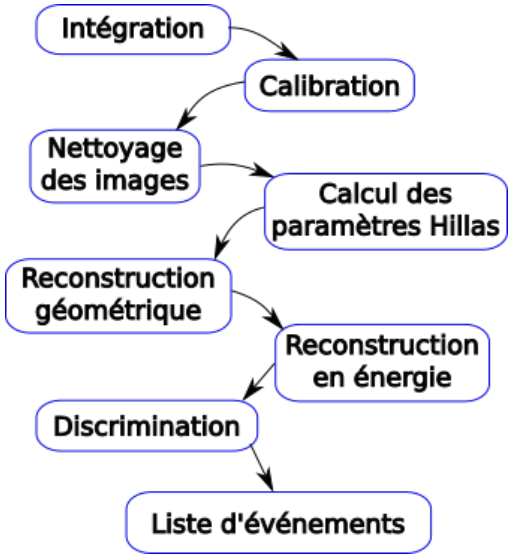


FIGURE 5.2 – Analyse de données Hillas.

La première est historique et est utilisée sur les télescopes H.E.S.S. I. La caméra capte le développement de la gerbe Cherenkov avec plusieurs images, puis les intègre électroniquement. Le signal obtenue est alors qualifié de signal intégré.

La deuxième a été testée avec l'installation du télescope H.E.S.S. II. Dans ce cas, la caméra enregistre toutes les images du développement de la gerbe Cherenkov. On parle alors de signaux vidéo ou waveform. **L'intégration** est alors faite plus tard dans l'analyse de données même (voir section 5.2). Cette étape permet d'augmenter le rapport signal sur bruit des données.

La calibration converti le signal électrique reçu (coups ADC) en nombre de photons détectés par la caméra (voir section 5.3). Cela rend les télescopes comparables entre eux.

Le nettoyage des images retire le bruit qui les compose (voir section 5.4). Le bruit des caméras est constitué d'une superposition du bruit due au PM et du ciel nocturne. Ce dernier est appelé Night Sky Background (NSB).

Les paramètres Hillas décrivent les images nettoyées en exploitant le fait que les gerbes Cherenkov sont semblables à des ellipses (voir section 5.5).

Ensuite, les événements doivent être reconstitués (voir section 5.6.1) afin de permettre la reconstruction stéréoscopique (voir section 5.6).

À partir de cette étape, seuls les événements constitués d'au moins deux télescopes sont conservés.

La reconstruction géométrique détermine la direction de la particule qui a créée la gerbe, ainsi que la position de l'impact de la gerbe au sol (ou paramètre d'impact) avec les paramètres Hillas des différents télescopes de l'événement (voir section 5.6.2).

La reconstruction en énergie détermine l'énergie de la particule initiale (voir section 5.6.4).

La discrimination évalue la probabilité que l'événement soit produit par un rayon gamma à l'aide de tous les paramètres cités précédemment. Cette étape est la discrimination (voir section 5.6.5).

Une liste d'événements est renvoyée par l'analyse. Chacun d'eux a une direction, une énergie et la probabilité d'être un rayon gamma. Avec cette liste, les astrophysiciens produisent des cartes du ciel, des courbes de lumière, des spectres, etc.

Le développement de l'analyse de données dans CTA regroupe de nombreuse personnes. La gestion des différentes versions de l'analyse est gérée via le programme *git* et un dépôt *github*.

5.1 Format de données

Les propriétés HPC d'un format de données ont déjà été abordées au chapitre 3. Cette section décrit plus en détail la structure du format de données utilisé dans l'analyse de données CTA-LAPP. Celui-ci est généré avec les générateurs de format de données décrits dans le chapitre 3. La section 5.1.1 introduit brièvement les méta données utilisées dans toute l'analyse de données. La section 5.1.2 décrit le format de données des signaux vidéo et la section 5.1.3 le format des données intégrées. Un format de données hybride est présenté dans la section 5.1.4. La section 5.1.5 présente le format de données des paramètres extraits des images, les paramètres Hillas. Le format de données des événements reconstruits est décrit dans la section 5.1.6. Finalement, la section 5.1.7 décrit les temporaires que nous utilisons dans notre analyse afin de minimiser les coûts dues aux allocations mémoire.

5.1.1 Méta données

Les méta données de l'analyse de données sont stockées dans une classe ***RunHeader*** qui décrit les en-têtes de tous les fichiers produits par cette analyse. Elle contient les directions de pointé des télescopes, leurs positions ainsi que leurs distances focal. Ces informations sont propagées à chaque étape de l'analyse de données. En effet, cette en-tête est commune à tous les fichiers produits par l'analyse de données quelque soit leur niveau. Dans ce cas, une telle classe simplifie d'utilisation du format de données car elle peut être copiée facilement.

5.1.2 Format de données vidéo

Les données vidéo décrivent le développement de la gerbe Cherenkov vue par une caméra. Chaque pixel de la caméra possède donc plusieurs valeurs dans le temps (entre 22 et 80 pour les différentes caméras de CTA qui enregistrent des signaux vidéo). Le signal temporel d'un pixel est semblable à une pulsation (voir figure 5.3). Comme tous les pixels ont le même nombre de valeurs dans le temps, il est naturel d'utiliser une matrice pour les stocker. Dans le format de données actuel de l'analyse de CTA, chaque ligne décrit un pixel et les colones ses différentes valeurs dans le temps. Cette méthode ne permet par une vectorisation efficace du traitement des pixels puisque ils doivent être traités les uns après les autres.

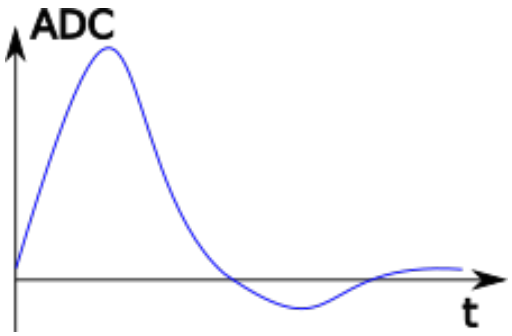


FIGURE 5.3 – Illustration d'un signal temporel obtenue sur un pixel de caméra sur quelques dixaines de nanosecondes.

Une méthode plus efficace consiste à transposer cette matrice. Dans ce cas, les lignes décrivent le temps et chaque colonne est un pixel. Cela permet une vectorisation beaucoup plus efficace et améliore également la compression des données (voir section 4.3.2.1). En effet, dans ce cas, chaque ligne décrit des données similaires, donc facilement vectorisable. De plus, les lignes sont bien plus grandes que dans l'autre cas (quelques milliers de valeurs contre quelques dixaines). La vectorisation sera donc bien plus efficace car le coup d'un tail [1] sera amoindri. Dans notre cas, la matrice est créée avec un pitch afin que toutes ses lignes soient alignées sur une frontière de registre vectoriel.

Le format de données est illustré par la figure 5.4. Un fichier d'observation est décrit par une classe ***SliceRun***. Elle contient un tableau de télescopes, représentés par la classe ***SliceTel***. Cette dernière contient l'identifiant du télescope, son type de caméra associé et son indice (qui permet d'obtenir sa position et sa distance focal grâce à la classe ***RunHeader***). Elle possède un tableau d'événements. Chaque événement, ***SliceEvent***, a un horodatage ainsi qu'un numéro et la matrice évoquée précédemment dont chaque élément est un entier non signé sur 16 bits.

5.1.3 Format de données intégré

Le format de données intégré suit le même schéma que le format de données vidéo. La matrice qui décrit les valeurs des pixels dans le temps est remplacée par un vecteur aligné. Les données intégrées sont représentées par des entiers sur 32 bits (int) car l'intégration d'entiers sur 16 bits pourrait saturer un short.

1. Ensemble d'instructions effectuées après un calcul vectorisé lorsque la taille du tableau à traiter n'est pas un multiple du nombre d'éléments d'un registre vectoriel, même si ce tableau est aligné.

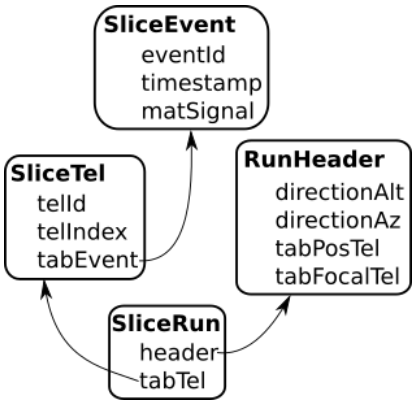


FIGURE 5.4 – Format de données utilisé pour l'analyse qui traite les donneés vidéo.

5.1.4 Format de données hybride

Un format de données intermédiaire entre le format intégré et le format vidéo est une possibilité. Cette méthode permet une compression avec perte des données puisque, quelques pourcents des pixels conservent leurs signaux vidéo (ceux qui décrivent la gerbe de particules) tandis que les autres sont intégrés (ceux qui contiennent du bruit). Ces derniers ne sont pas jetés car il est important d'étudier le bruit afin de le soustraire du signal.

Néanmoins, cette idée n'a pas été retenue par le consortium CTA pour le moment. Cette section décrit l'optimisation de ce format de données même si il n'est pas encore utilisé. La figure 5.5 montre l'agencement des données hybrides.

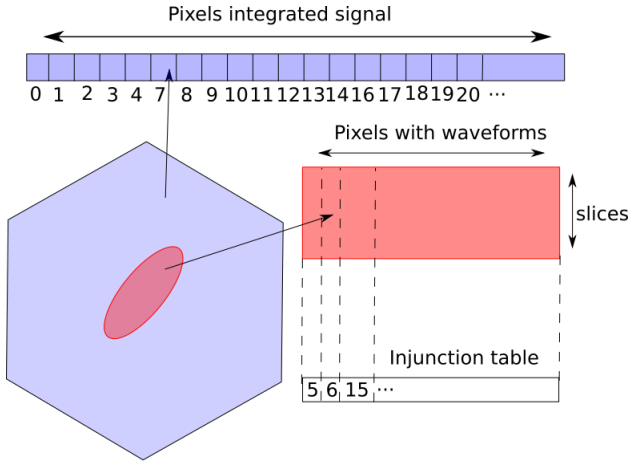


FIGURE 5.5 – Format de données utilisé pour l'analyse qui mélange les donneés vidéo et les intégrées.

Comme nous l'avons vu dans les sections précédentes, le format de données vidéo est stocké dans une matrice tandis que celui intégré est dans un vecteur. Nous conservons ces agencements initiaux mais en ajoutant une table d'injonction qui réfère les pixels sélectionnés (puisqu'ils seront *a priori* moins nombreux que les autres).

Cet agencement permet le traitement spécifique des deux types de données voir même une compression spécialisée. La table d'injonction permet quant à elle de reconstituer la caméra si nécessaire.

5.1.5 Format de données Hillas

Deux formats de données décrivent les paramètres Hillas calculés pour chaque image (voir figure 5.6). Le premier stocke les résultats du calcul des paramètres Hillas dans un format similaires au données brutes ; une classe ***TabHillas*** contient un tableau de télescopes ***HillasTel*** qui contient les paramètres Hillas, ***HillasEvent***, de chacune de ses images. Comme les télescopes ne contiennent que leurs propres paramètres Hillas aucune reconstruction stéréoscopique n'est possible. Le deuxième format de données décrit les paramètres Hillas en terme d'événements ; une classe ***TimeHillas*** contient l'ensemble des événements, ***TimeHillasEvent***, composés des paramètres des télescopes qui ont été conservés ***TimeHillasTel***.

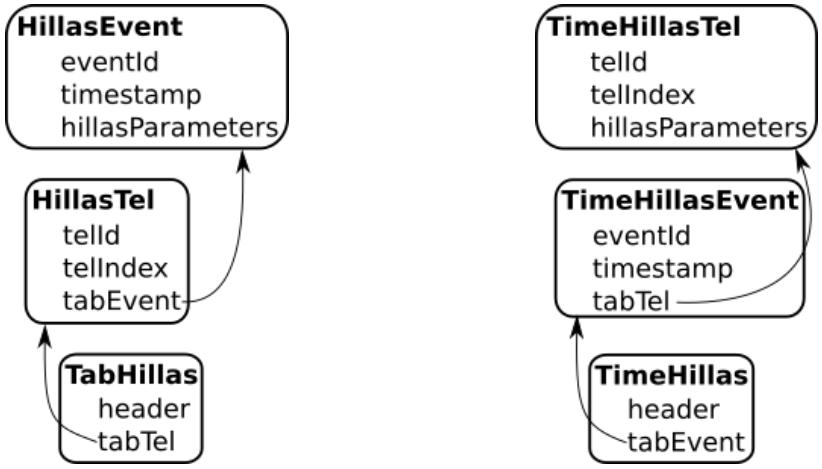


FIGURE 5.6 – **À gauche**, le format de données qui permet de stocker les paramètres Hillas par télescopes. **À droite**, le format de données qui décrit les paramètres Hillas par événement. Les en-têtes des deux format de données sont également constituées de la classe ***RunHeader***.

La conversion entre les deux formats de données qui décrivent les paramètres Hillas est possible grâce au générateur de format de données (voir chapitre 3) et sera détaillée dans la section 5.6.1.

5.1.6 Format de données des événements reconstruits

Le dernier format de données de l'analyse classique décrit les événements reconstruits par stéréoscopie (voir figure 5.7). Il est constitué d'une liste d'événements où chacun d'eux est composé de paramètres généraux : d'un numéro d'identification et un horodatage ; les paramètres physiques : la direction de la particule incidente (altitude, et azimut), son énergie, et la probabilité qu'il soit un rayon gamma ; enfin, des valeurs de contrôle décrivent le paramètre d'impact de la gerbe de particules au sol ($core_x$ et $core_y$), le nombre de télescopes impliqués dans l'événement (multiplicité), et une qualité de reconstruction qui indique à quelle étape de l'analyse l'événement a été rejeté. Ce dernier paramètre est utilisé pour calculer les différentes surfaces effectives présentées dans la section 5.6.3.

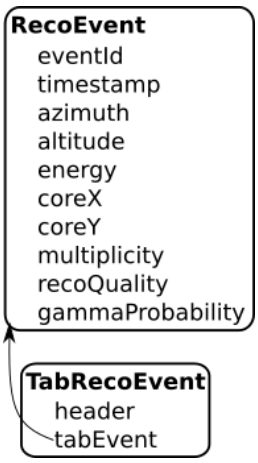


FIGURE 5.7 – Format de données des événements reconstruits.

5.1.7 Gestion des temporaires

Les allocations de la mémoire sont très coûteuses en temps. Nous utilisons donc une classe temporaire qui contient toutes les variables nécessaires à l'analyse de données. De cette manière, aucune allocation n'est superflue.

Des matrices décrivent l'image à traiter pour chercher plus efficacement les pixels voisins (voir section 5.2.1.3). Un tableau décrit le signal calibré calculé dans la section 5.3. L'étape de nettoyage des images optimisée utilise également un tableau et une matrice temporaire (voir section 5.4.2).

5.2 Intégration des données vidéo

L'intégration des données vidéo consiste à sommer le signal de chaque pixel dans le temps. Cette somme peut être réalisée de différentes manières, de sorte à maximiser le rapport signal sur bruit.

Si la totalité du signal temporel de chaque pixel est sommé, on parle d'intégration complète. Une telle intégration n'offre pas un excellent rapport signal sur bruit. En effet, le signal est minoritaire dans le temps car la durée de déploiement de la gerbe Cherenkov n'est que d'une dixaine de nanosecondes, contrairement au temps total d'acquisition qui est supérieur à 30 nanosecondes. Cela implique, que, dans le meilleur des cas, le signal ne représente qu'un tiers de l'ensemble des valeurs enregistrées par les pixels. Dans ces conditions, le rapport signal sur bruit ne peut pas être optimal.

La solution consiste à ne sommer qu'une partie du signal temporel, c'est une intégration partielle. La taille de la fenêtre d'intégration est choisie, conformément à la réponse typique d'un PM qui reçoit un seul photon. Cette courbe de référence, ou ref-shape, permet de déterminer une fenêtre d'intégration optimale. Chaque équipe en charge de construire une caméra fournie cette courbe de référence. La figure 5.8 montre la courbe qui correspond à une caméra LST-CAM, les caméras les plus rapides de CTA. Le pic ne représente bien que le tiers du temps total d'acquisition.

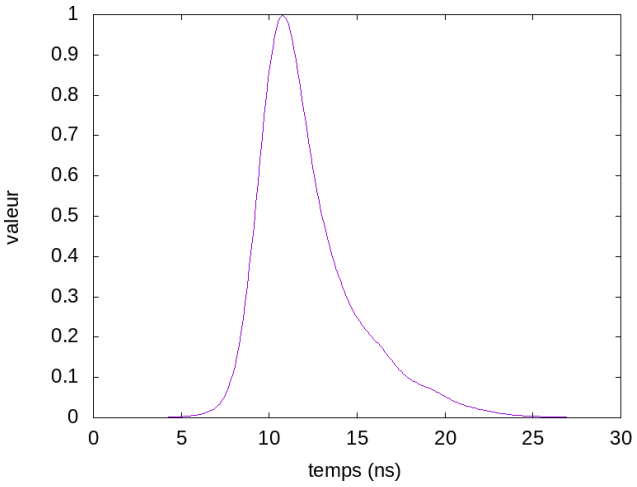


FIGURE 5.8 – Courbe de référence d'une caméra LST-CAM, les plus rapides de CTA. Toutes les courbes de références sont normalisées.

L'optimisation du rapport signal sur bruit consiste à sélectionner une partie suffisamment importante du pic de signal tout en excluant le bruit de fond du pixel. Dans la méthode classique, la fenêtre d'intégration est fixée, à l'oeil, et le temps entre de début de la pulsation et le maximum atteint par le signal est également fixé. Nous avons envisagé une méthode plus automatique. Un pourcentage du maximum de la pulsation est défini, typiquement 10%. L'ensemble du signal au-delà de ce seuil permet de définir la taille de la fenêtre d'intégration. Le temps entre de départ de l'intégration et le maximum est également calculé.

Le maximum du signal dans le temps varie énormément, il doit donc être calculé pour chaque pixel à chaque image.

5.2.1 Calcul du temps au signal maximal

De nombreuses méthodes permettent le calcul du temps au signal maximum (5 sont implémentées dans l'analyse officielle de CTA). Nous en détaillerons deux dans cette partie.

5.2.1.1 Maximum local par pixel

La méthode la plus simple calcule le maximum du signal pour chaque pixel et ajuste la fenêtre d'intégration sur celui-ci. Cette méthode, au demeurant très légitime, ne produit pas complètement l'effet escompté.

En effet, si la position de la fenêtre d'intégration est calculée en fonction du signal maximal contenu dans les pixels, ceux qui contiennent de l'information voient leur signal augmenter, mais ceux qui contiennent du bruit également puisque le bruit maximal sera sélectionné. La distribution du signal reçu par chaque pixel est alors biaisée, d'un biais b, illustré dans la figure 5.9. Dans ce cas, la distribution de bruit n'est plus centrée sur 0, comme l'est habituellement une distribution de bruit blanc. Or les coupures [2] présentes dans l'analyse de données utilisent cette propriété essentielle du bruit afin d'améliorer la qualité de la reconstitution. Si cette propriété n'est pas conservée, les résultats pourraient être fortement biaisés.

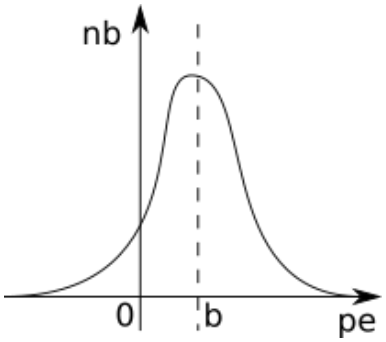


FIGURE 5.9 – Illustration du biais, b, de la méthode d'intégration qui utilise le maximum local de chaque pixel.

La qualité supplémentaire de l'intégration est donc de conserver la moyenne du bruit proche de 0. Il y aura toujours un léger biais positif, due au fait que la caméra ne déclenche que sur un excès de photons, mais il doit être minimal.

5.2.1.2 Maximum du voisinage

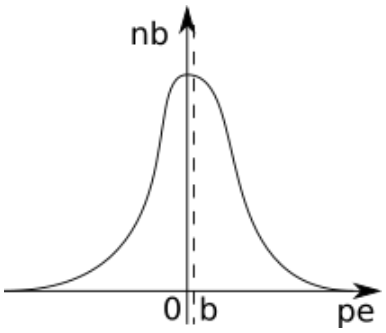


FIGURE 5.10 – Illustration du biais, b, obtenu avec la méthode d'intégration qui prend en compte les maxima des voisins de chaque pixel.

La méthode de calcul du maximum utilisée dans l'analyse de données prend en compte les voisins du pixel courant. Le temps du maximum est défini comme le temps où la somme des signaux des pixels voisins, ajoutée au pixel courant, est maximale.

L'intérêt de cette méthode est statistique. En effet, si un pixel contenant du signal est entouré de bruit, le signal prime, donc le maximum trouvé est bien influencé par le signal. À l'inverse, un pixel bruité entouré d'autres pixels bruités aura un maximum aléatoire, le bruit n'est donc statistiquement pas maximisé. Contrairement à la méthode précédente, le rapport signal sur bruit augmente, et le bruit n'est pas maximisé. Il en résulte une distribution de valeur bien plus centrée sur 0 (voir figure 5.10).

L'optimisation de cette méthode est bien plus complexe que la précédente. Elle nécessite une recherche rapide des pixels voisins suivit d'un calcul de maximum.

2. Une coupure désigne un seuil, supérieur ou inférieur, effectué sur une variable qui déterminera si la donnée en question doit être conservée ou non.

5.2.1.3 Recherche des pixels voisins

Les calculs prenant en compte les voisins d'un pixel sont récurrents dans l'analyse de données. Une manière efficace consiste à convertir les caméras en matrices. Cette conversion est triviale pour les caméras carrées. Ici, nous détaillons la conversion pour les caméras hexagonales (LST-CAM, NECTAR-CAM, FLASH-CAM et DC-CAM).

La figure 5.11 montre la déformation désirée du motif d'accès hexagonal en matrice. Ici, les six voisins (rouge) du pixel central (noir) sont conservés dans la matrice d'arrivée. Les voisinages plus éloignés sont eux aussi conservés

Les pixels en haut à droite et en bas à gauche ne sont pas considérés comme voisin dans la matrice bien que ce soit le cas en mémoire.

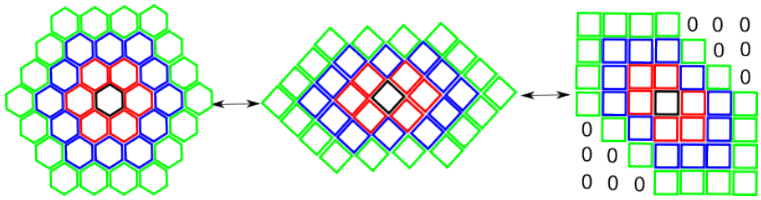


FIGURE 5.11 – Déformation d'un motif d'accès hexagonal en matrice.

Les sept types de caméras de CTA comptent quatre caméras hexagonales. La déformation du repère en matrice doit donc se faire automatiquement suivant le motif, ce qui simplifiera l'analyse par la suite.

Cette déformation automatique utilise un quadtree (voir définition 5.2.1). Les pixels de la caméra sont ajoutés un par un dans le quadtree et sont naturellement triés spatialement (voir figure 5.12). La caméra doit être centrée à l'origine du repère de départ.

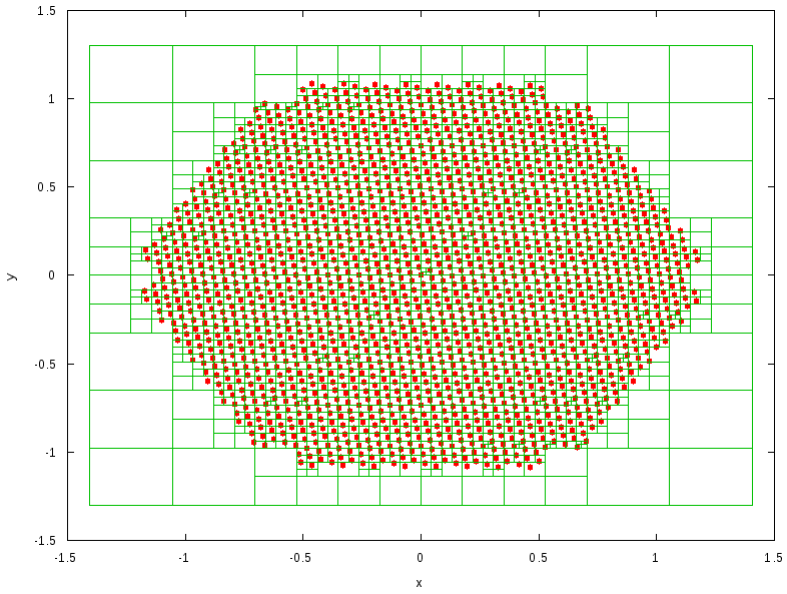


FIGURE 5.12 – Séparation des pixels de la caméras (points rouges) dans un quadtree en vert.

Définition 5.2.1 – **Les quadtrees**

Un quadtree est un arbre qui permet de séparer des éléments entre eux. Ils sont généralement utilisés en simulations sous leur forme à trois dimensions pour réduire la complexité des algorithmes, notamment le problème à N corps en interaction gravitationnelle quand N est très grand. Nous les utilisons pour séparer les pixels dans une caméra.

Les règles de construction d'un quadtree sont assez simples (voir figure 5.13). Une cellule ne peut contenir qu'une seule particule, cette cellule est alors rempli. Si une deuxième particule tombe dans une cellule rempli, celle-ci se coupe en quatre (ou huit pour un octree), puis les particules sont replacées dans les cellules filles. Ce processus se répète pour toutes les particules à ajouter dans le quadtree.

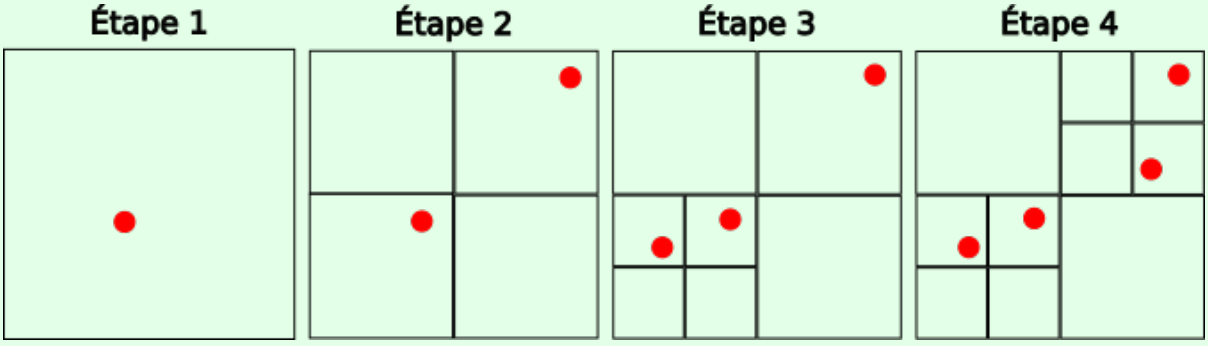


FIGURE 5.13 – Remplissage d'un quadtree avec des particules.

La figure 5.14 montre comment calculer les déformations dx et dy des pixels. Les pixels A, B et C sont triés spatialement par le quadtree : A et B sont sur la même ligne et A et C sont sur la même colonne. L'écart entre A et C donne donc la déformation en x et l'écart entre A et B la déformation en y. La déformation du repère qui transforme la caméra en matrice est donnée par :

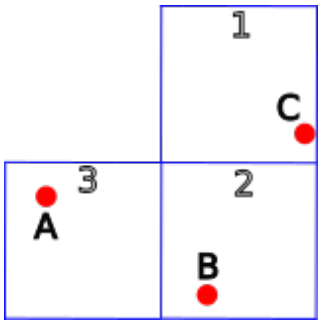


FIGURE 5.14 – Utilisation du quadtree pour déterminer les déformations du repère sur l'axe Ox et l'axe Oy. Les pixels sont les points rouges, les quadtree les cases bleues.

$$\delta_x = \frac{c_x - b_x}{\|c_y - b_y\|} \tag{5.1}$$

$$\delta_y = \frac{b_y - a_y}{\|a_y - b_x\|} \tag{5.2}$$

La déformation de la position de tous les pixels est alors donnée par :

$$\begin{pmatrix} p\prime_x \\ p\prime_y \end{pmatrix} = \begin{pmatrix} 1 & \delta_y \\ \delta_y & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} \tag{5.3}$$

Le repère déformé est alors utilisé pour créer une table d'injonction qui permet de construire une matrice à partir des données caméra, stockées dans un vecteur. La figure 5.15 montre la conversion d'une LST-CAM en matrice.

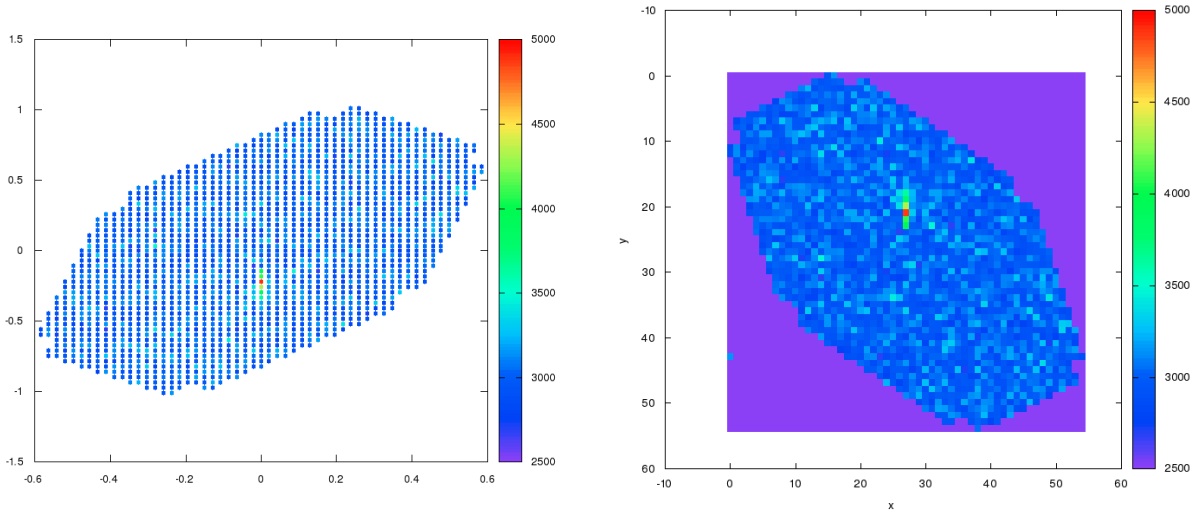


FIGURE 5.15 – **À gauche**, le repère des positions des pixels déformé par la formule 5.3. **À droite**, la matrice obtenue après conversion d'une caméra LST-CAM.

5.2.1.4 Optimisation du calcul du maximum de voisinage

L'optimisation du calcul du maximum du voisinage utilise l'optimisation de la recherche des pixels voisins (voir section 5.2.1.3). Les caméras sont converties en matrices pour plus d'efficacité. Les 6 ou 8 voisins des pixels, suivant les caméras (hexagonales ou carrées), sont pris en compte. Comme chaque ligne de la matrices qui décrit le signal temporel des pixels correspond au même temps pour tous, le calcul du maximum peut être vectorisé à l'aide d'un masque, fourni par les fonctions intrinsèques (voir définition 5.2.2).

Définition 5.2.2 – **Les fonctions intrinsèques**

Les fonctions intrinsèques représentent la limite entre le langage C et l'assembleur. Ce sont des fonctions C qui contiennent de l'assembleur. Cela permet de programmer quasiment en assembleur (sans les inconvénients d'un vrai assembleur) dans un programme C ou C++. Ces fonctions sont très utiles pour effectuer des optimisations que le compilateur ne permet pas.

Anecdote 5.2.1 – **Intégration manuelle ou automatique ?**

Au tout début de nos tests sur l'analyse, nous obtenions des résultats qui n'étaient pas en accord avec ceux attendus. Cela était dû au fait que nous utilisions le signal intégré fourni par les simulations. Nous avons appris plus tard qu'il ne fallait surtout pas l'utiliser.

5.3 Calibration des données

La calibration convertie des données électriques, coups d'ADC, en nombre de photons tombés sur le télescope. Elle permet de rendre les différentes images des télescopes comparables.

5.3.1 Calibration des données intégrées par les caméras

5.3.1.1 Méthode classique

La calibration des données intégrées est la plus simple. Dans l'expérience H.E.S.S. elle sert également à déterminer si la voie haut gain doit être utilisée ou seulement la voie bas gain, pour chaque pixel. Dans le cas de ces photomultiplicateurs (PM), deux images sont produites : l'image fortement amplifiée (voie haut gain) et l'image faiblement amplifiée (voie bas gain).

Dans ce cas, le nombre de photons reçus par le télescope, s, est donné par :

$$s = \begin{cases} \dfrac{(Adc_{hi} - Ped_{hi}) \cdot Gain_{hi}}{Flatfield_{hi}} & \text{si la voie haut gain n'est pas saturée} \\ \dfrac{(Adc_{lo} - Ped_{lo}) \cdot Gain_{lo}}{Flatfield_{lo}} & \text{si la voie haut gain est saturée mais pas la voie bas gain} \\ -1 & \text{sinon} \end{cases} \tag{5.4}$$

Où Adc_{hi}, Ped_{hi}, $Gain_{hi}$ et $Flatfield_{hi}$ sont respectivement le signal (ADC), le piédestal (bruit moyen du PM), le gain (nombre d'électrons produit par un seul photon dans le PM) et la sensibilité de le voie haut gain, et Adc_{lo}, Ped_{lo}, $Gain_{lo}$ et $Flatfield_{lo}$ sont respectivement le signal, le piédestal, le gain et la sensibilité de la voie bas gain.

Après l'étape de calibration, le nombre total de photons reçus par le télescope est calculé. Si il est trop petit, inférieur à quelques centaines pour les LST-CAM, le télescope est rejeté et ses données ne seront pas utilisée dans l'analyse.

5.3.1.2 Optimisation

L'équation 5.4 a été réécrite sans branchement de sorte à être plus performante. Cela garantit que le pipeline CPU sera utilisé efficacement (voir définition 5.3.1).

Définition 5.3.1 – **Pipeline CPU**

Le pipeline CPU désigne l'enchaînement des instructions effectuer dans le CPU pour faire un calcul (voir figure 5.16). Les vrais pipelines CPU ont bien plus d'étages mais cet exemple permet d'illustrer l'essentiel. En premier lieu, le CPU va chercher l'instruction à exécuter (**IF**, Instruction Fetch). Ensuite il la décode (**ID**, Instruction Decode). Il l'exécute avec l'unité arithmétique et logique (ALU, Arithmetic Logical Unit). Il effectue les appels mémoire nécessaires (**MEM**), puis stocke le résultat (**WB**, Write Bytes).

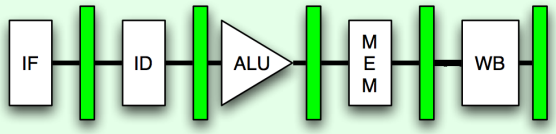


FIGURE 5.16 – Pileline CPU simplifié.

Une utilisation efficace du pipeline consiste à exécuter plusieurs instructions simultanément (une instruction est exécutée, une autre est décodée, une est chargée, etc). Pour que ce mode soit efficace, il ne doit pas y avoir d'ambiguïté sur les prochaines instructions à charger. Dans le cas d'un branchement, la prochaine instruction chargée ne pourra être certaine qu'après traitement complet de la précédente. Dans ce cas, le prédicateur de branchement parie sur la prochaine instruction à charger. Si il a raison, l'exécution a une constante efficacité, sinon le pipeline doit être vidé puis rempli de nouveau, ce qui coûte plusieurs centaines de cycles.

Éviter ce dernier cas est crucial pour garantir les performantes d'un programme. Donc le nombre de branchements doit être limité au maximum dans une fonction chaude (voir définition 5.3.2).

Définition 5.3.2 – **Fonction chaude**

Une fonction chaude est une fonction appelée souvent dans un programme. L'utilisation de programmes comme *valgrind*, *perf* ou *gprof* hiérarchisent les fonctions chaudes d'un programme de la plus coûteuse à la moins coûteuse en temps.

La méthode suivante calibre chaque pixel sans branchement et met à jour un vecteur de pixels morts (qui seront exclus de l'analyse). Quelques variables sont nécessaires, la couleur rouge indique quand ce sont *a priori* des booléens, :

— **k** : un vecteur de booléens, 1 si le pixel doit être gardé, 0 sinon
— Adc_{hi} : le signal digitalisé de la voie haut gain
— Adc_{lo} : le signal digitalisé de la voie bas gain
— Ped_{hi} : le piédestal de la voie haut gain
— Ped_{lo} : le piédestal de la voie bas gain
— $Gain_{hi}$: le gain de la voie haut gain
— $Gain_{lo}$: le gain de la voie bas gain
— $Flatfield_{hi}$: la sensibilité de la voie haut gain
— $Flatfield_{lo}$: la sensibilité de la voie bas gain

Le vecteur du signal calibré est **s**, et le vecteur final de pixels morts est **b**.
Premièrement, la voie à utiliser doit être déterminée, haut (équation 5.6) ou bas gain (équation 5.7) :

$$\xi = 1.6\,10^7 \tag{5.5}$$
$$k_{hi} = (Gain_{hi} > 0 \;\&\&\; Flatfield_{hi} > 0 \;\&\&\; Ped_{hi} < \xi \;\&\&\; Adc_{hi} < \xi) \tag{5.6}$$
$$k_{lo} = (Gain_{lo} > 0 \;\&\&\; Flatfield_{lo} > 0 \;\&\&\; Ped_{lo} < \xi \;\&\&\; Adc_{lo} < \xi) \tag{5.7}$$

Si un gain ou une sensibilité est inférieur à zéro, c'est que leur valeur est indéterminée. Les seuils déterminent si une voie doit être utilisée ou non (variable ξ). Ces valeurs sont arbitrairement hautes pour l'instant, car les données utilisées sont des simulations et qu'elles n'ont pas de notions de saturation. Ces coupures seront ajustées par la suite aux vraies valeurs de saturation des caméras.

La calibration est calculée avec les variables précédentes (-1 indique que la calibration n'a pu être effectuée). Les équations 5.8, 5.9, 5.10 et 5.11 sont composées de plusieurs parties. Les termes marrons contribuent quand la voie haut gain peut être utilisée. Les termes verts contribuent lorsque la voie haut gain est saturée mais que la voie bas gain ne l'est pas. Les termes gris contribuent quand aucune des voies ne peut être utilisée, dans ce cas, la valeur renvoyée est -1.

$$Gain_{calib} = k_{hi} \cdot Gain_{hi} + (1 - k_{hi}) \times (Gain_{lo} \cdot k_{lo} + (k_{lo} - 1)) \tag{5.8}$$
$$Flatfield_{calib} = k_{hi} \cdot Flatfield_{hi} + (1 - k_{hi}) \times (Flatfield_{lo} \cdot k_{lo} + (k_{lo} - 1)) \tag{5.9}$$
$$Ped_{calib} = k_{hi} \cdot Ped_{hi} + (1 - k_{hi}) \times (Ped_{lo} \cdot k_{lo} + (k_{lo} - 1)) \tag{5.10}$$
$$Signal_{calib} = k_{hi} \cdot Adc_{hi} + (1 - k_{hi}) \times (Adc_{lo} \cdot k_{lo} + (k_{lo} - 1)) \tag{5.11}$$

Le signal calibrée est donnée par :

$$s = k \cdot \frac{(Signal_{calib} - Ped_{calib}) \cdot Gain_{calib}}{Flatfield_{calib}} \tag{5.12}$$

Sur un ordinateur, une division est bien plus lente qu'une multiplication. C'est pour cela que les termes $\frac{Gain_{hi}}{Flatfield_{hi}}$ et $\frac{Gain_{lo}}{Flatfield_{lo}}$ de l'équation 5.12 sont calculés préalablement. Nous définissons :

$$GainFlatfield_{hi} = \frac{Gain_{hi}}{Flatfield_{hi}}, -1 \text{ si gain ou sensibilité inutilisables} \quad (5.13)$$

$$GainFlatfield_{lo} = \frac{Gain_{lo}}{Flatfield_{lo}}, -1 \text{ si gain ou sensibilité inutilisables} \quad (5.14)$$

Si on remplace ces expressions dans les équations 5.6 et 5.7 nous obtenons les équations 5.15 et 5.16 utilisées dans notre calibration :

$$k_{hi} = (GainFlatfield_{hi} > 0 \;\&\&\; Ped_{hi} < 16000000 \;\&\&\; Adc_{hi} < 16000000) \quad (5.15)$$

$$k_{lo} = (GainFlatfield_{lo} > 0 \;\&\&\; Ped_{lo} < 16000000 \;\&\&\; Adc_{lo} < 16000000) \quad (5.16)$$

Les équations 5.8 et 5.9 deviennent :

$$GainFlatfield_{calib} = k_{hi} \cdot GainFlatfield_{hi} \quad (5.17)$$

$$+ \; (1 - k_{hi}) \times (GainFlatfield_{lo} \cdot k_{lo} + (k_{lo} - 1)) \quad (5.18)$$

Donc, l'équation 5.12 devient :

$$s = k \cdot (Signal_{calib} - Ped_{calib}) \cdot GainFlatfield_{calib} \quad (5.19)$$

Nous devons également déterminer si le pixel doit être gardé pour les prochains calculs :

$$b = k \cdot (k_{hi} + (1 - k_{hi}) \times k_{lo}) \quad (5.20)$$

Les variables rouges sont, a priori, des booléens, mais la vectorisation n'est efficace que si elle est utilisée avec des tableaux de même type (dans notre cas des float). C'est pour cette raison que la calibration n'utilise que les float. Les test effectués avec le programme *maqao* [66] montrent que la vectorisation est totale dans notre cas.

Optimisation cachée 5.3.1 – **Utilisation pour CTA**

Il est fort probable que seul les signaux hauts ou bas gain soient enregistrés dans CTA et non les systématiquement les deux comme dans les expériences précédentes. En effet, si la voie haut gain est saturée pour un pixel, elle ne sera jamais utilisée. Il n'est donc pas utile de sauvegarder le signal associé. La sélection des voies sera probablement effectuée en amont, par la caméra. Celle-ci renverra alors directement un tableau décrivant quelle voie utiliser.

5.3.2 Calibration des données vidéo

La calibration des données vidéo suit le même schéma que dans la section 5.3.1. Les optimisations de la section 5.3.1.2 restent valables.

Néanmoins, une correction doit être appliquée si l'intégration des données vidéo n'est pas complète. Dans ce cas, la portion de signal conservée est bien plus faible que la portion de signal totale. Pour que les données soient toujours comparables entre différents télescopes la correction doit tenir compte de cette perte de signal.

Le facteur de correction est le rapport entre signal total du pixel et le signal contenu dans la fenêtre d'intégration (voir figure 5.17).

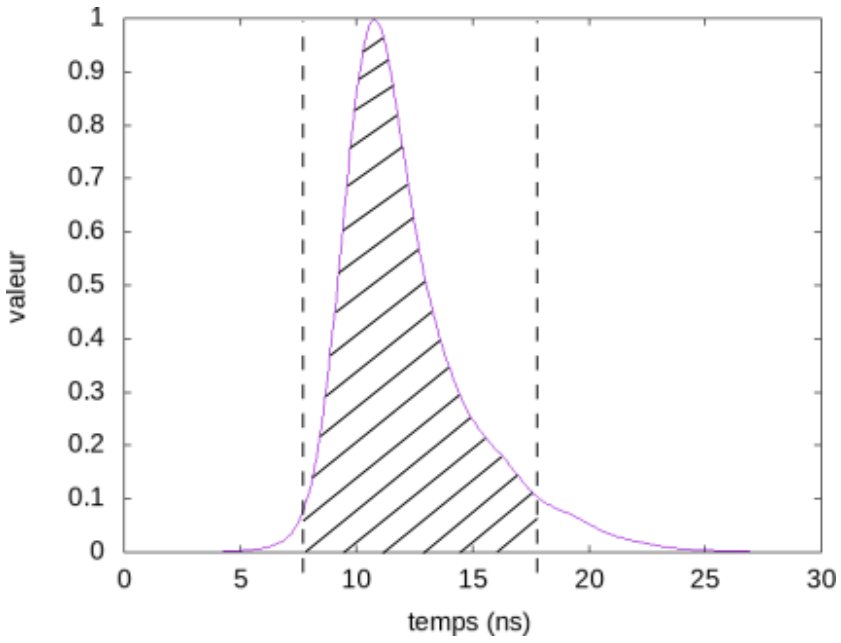


FIGURE 5.17 – Intégration du signal de référence sous la courbe, pour une caméra LST-CAM. La correction est le rapport entre l'intégralle totale de la courbe et l'intégralle (hachurée) de la fenêtre d'intégration, délimité par les deux trais pointillés verticaux.

Afin de prendre en compte cette correction dans le calcul optimisé de la calibration de la section 5.3.1.2, les équations 5.13 et 5.14 deviennent :

$$GainFlatfield_{hi} = \frac{Gain_{hi} \cdot \delta_{hi}}{Flatfield_{hi}}, -1 \text{ si gain ou sensibilité inutilisables}$$
$$GainFlatfield_{lo} = \frac{Gain_{lo} \cdot \delta_{lo}}{Flatfield_{lo}}, -1 \text{ si gain ou sensibilité inutilisables}$$

Où δ_{hi} et δ_{lo} sont les corrections des voies haut et bas gain calculées avec leurs signaux de référence respectifs.

Optimisation cachée 5.3.2 – **Calibration de l'analyse officielle de CTA**
Dans l'analyse officielle de CTA (ctapipe) [30], la calibration est effectuée en premier et ensuite l'intégration. Cet ordre est moins efficace car la quantité de données manipulée est plus importante. Si la matrice du signal vidéo contient N pixels et M temps. Les données vidéo sont décrites par $M * N$ short, ceux-ci sont convertis en $M * N$ float puis sont intégrés en N float. Pour une taille typique ou $N = 2000$ et $M = 30$, cette analyse utilise 368 000 octets. L'analyse optimisée que nous décrivons ici n'utilise que 132 000 octets, soit 2.78 fois moins de données. Cela est d'autant plus important compte tenu du nombre d'images à traiter.

5.4 Nettoyage des images

La reconstruction Hillas est extrêmement sensible au bruit contenu dans les images. L'étape de nettoyage des images est donc essentielle dans cette analyse.

5.4.1 Méthode de l'expérience H.E.S.S.

La méthode de nettoyage d'image décrite dans ce qui suit provient de l'expérience H.E.S.S.. Cette méthode prend en compte le signal des pixels voisins. Le signal d'un pixel n'est conservé que si il est supérieur à un seuil et si au moins un de ses voisins est supérieur à un autre seuil. Dans l'analyse de données de l'expérience H.E.S.S. chaque pixel a une liste de voisins. Cette technique supprime des pixels isolés de la caméra, qui ont du signal mais ne font pas partie d'une gerbe. La figure 5.18 illustre ce nettoyage d'image.

5.4.2 Nettoyage historique optimisé

Comme nous l'avons mentionné précédemment, les voisins de chaque pixels doivent être connus. Une liste de voisins ne peut pas être utilisée car cela ne permet pas une vectorisation efficace. La méthode utilisée consiste à convertir les images des caméras en matrices (voir section 5.2.1.3), de sorte que tous les voisins aient un motif d'accès identique pour tirer parti du pré-chargement des données CPU.

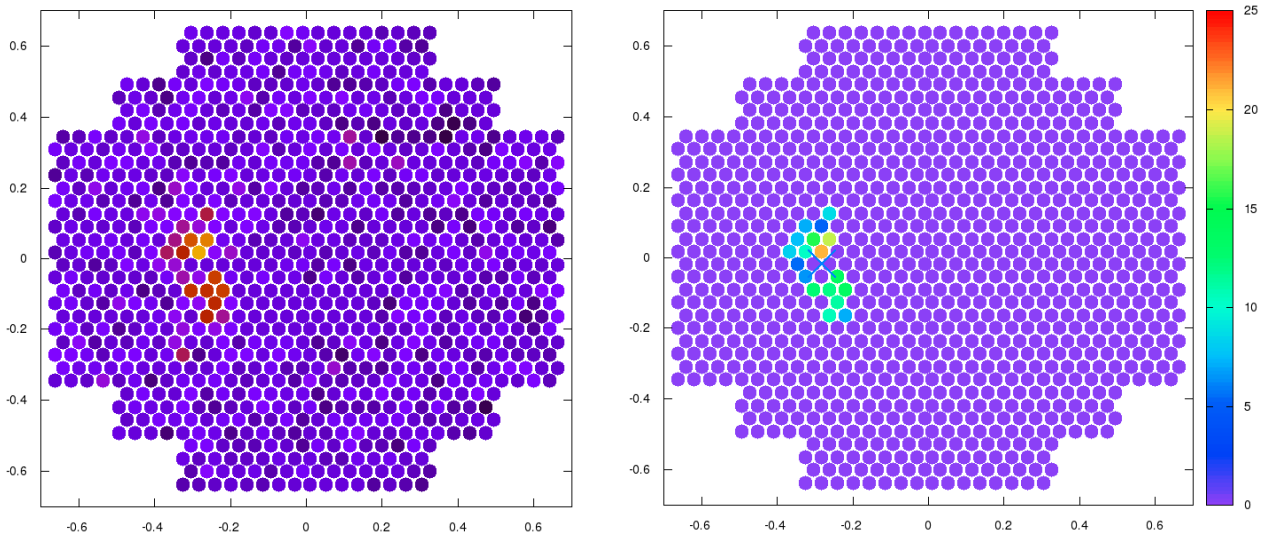


FIGURE 5.18 – Nettoyage des images calibrés. **À gauche,** l'image calibré que l'on désire nettoyer. **À droite**, l'image nettoyée.

Le nettoyage d'image met à jour le vecteur $\mathbf{b} \in [\![0,1]\!]^n$ qui décrit si un pixel est conservé (1) ou non (0).

Soit une matrice $\mathbf{b}_{quad}$ qui contient le vecteur $\mathbf{b}$ décrivant les pixels à prendre en compte dans la reconstruction. La matrice $\mathbf{c}_{quad}$ définie les pixels à conserver après l'étape de nettoyage. Le vecteur de l'image calibrée, $\mathbf{s}$, est également converti en matrice, $\mathbf{s}_{quad}$.

Les indices du pixel considéré sont i, j dans la matrice correspondante $m \times n$, $0 < i < m$ et $0 \leq j \leq n$. Le test peut être écrit comme suit :

$$keep_c = \left(\mathbf{s}_{quad}^{i,j} \geq s_1\right) \,\&\&\, \left(\mathbf{s}_{quad}^{i-1,j-1} \geq s_2 || \mathbf{s}_{quad}^{i-1,j} \geq s_2 || \mathbf{s}_{quad}^{i,j-1} \geq s_2 || \mathbf{s}_{quad}^{i,j+1} \geq s_2 || \mathbf{s}_{quad}^{i+1,j} \geq s_2 || \mathbf{s}_{quad}^{i+1,j+1} \geq s_2\right) \tag{5.21}$$

$$keep_n = \left(\mathbf{s}_{quad}^{i,j} \geq s_2\right) \,\&\&\, \left(\mathbf{s}_{quad}^{i-1,j-1} \geq s_1 || \mathbf{s}_{quad}^{i-1,j} \geq s_1 || \mathbf{s}_{quad}^{i,j-1} \geq s_1 || \mathbf{s}_{quad}^{i,j+1} \geq s_1 || \mathbf{s}_{quad}^{i+1,j} \geq s_1 || \mathbf{s}_{quad}^{i+1,j+1} \geq s_1\right) \tag{5.22}$$

$$\mathbf{c}_{quad}^{i,j} \;=\; \mathbf{b}_{quad}^{i,j} \,\&\&\, (keep_c \,||\, keep_n) \tag{5.23}$$

La formule 5.21 garantie qu'un pixel n'est conservé que si il a un signal supérieur à un seuil s_1 (7 dans le cas de H.E.S.S., 30 dans le cas de CTA), et si un de ses voisins a un signal supérieur à un seuil s_2 (5 dans le cas de H.E.S.S., 25 dans le cas de CTA).

La formule 5.22 conserve un pixel si son signal est supérieur à s_2 et si un de ses voisin est supérieur à s_1 car il serait rejeté par la formule 5.21 seule.

Les équations 5.22, 5.21 et 5.23 garantissent un calcul sans branchement. Cela permet une utilisation efficace du pipeline CPU (voir définition 5.3.1).

Une fois cette étape terminée, la matrice $\mathbf{c}_{quad}$, donnée par la l'équation 5.23, doit être convertie en vecteur, $\mathbf{c}$ pour décrire quels pixels doivent être utilisés dans la caméra. La figure 5.19 illustre le nettoyage d'image obtenu par cette méthode.

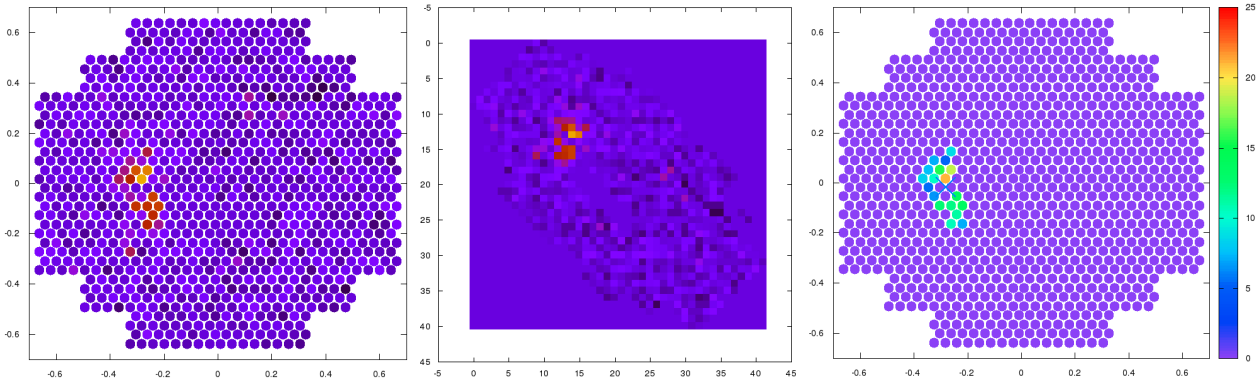


FIGURE 5.19 – Nettoyage d'image. **À gauche**, l'image calibré à nettoyer. **Au centre**, la caméra convertie en matrice. **À droite**, l'image nettoyée obtenue.

5.4.3 Nettoyage par transformée en ondelettes

Ce travail a été effectué par un stagiaire de master II que j'ai eu le privilège de co-encadrer. Nous avons surtout échangé sur les méthodes d'optimisation tant du calcul, des résultats et des tests décrits dans cette section. Notamment en ce qui concerne le travail présenté en section 5.4.3.3 qui a donné lieu à une note interne dans CTA [67].

Les transformées en ondelettes sont communes en traitement du signal, notamment pour la suppression de bruit.

5.4.3.1 Définition de la transformée en ondelettes

La transformée en ondelettes est une variante de la transformée de Fourier. Là où la transformée de Fourier décompose un signal sur un espace de fréquences, la transformée en ondelettes le décompose sur un espace défini par les appels récursifs d'une fonction donnée : l'ondelette.

La transformée en ondelettes discrète, DWT (Discret Wavelet Transform), est une méthode de traitement du signal. Elle est très utilisée dans le traitement d'images, notamment en compression du signal et en nettoyage d'image (notre application). Elle utilise des applications successives de filtres passe-haut et passe-bas pour décomposer le signal donné. La figure 5.20 illustre la DWT multi-échelles d'une image effectuée à partir d'appels successifs de la DWT définit sur un voisinage immédiat des pixels.

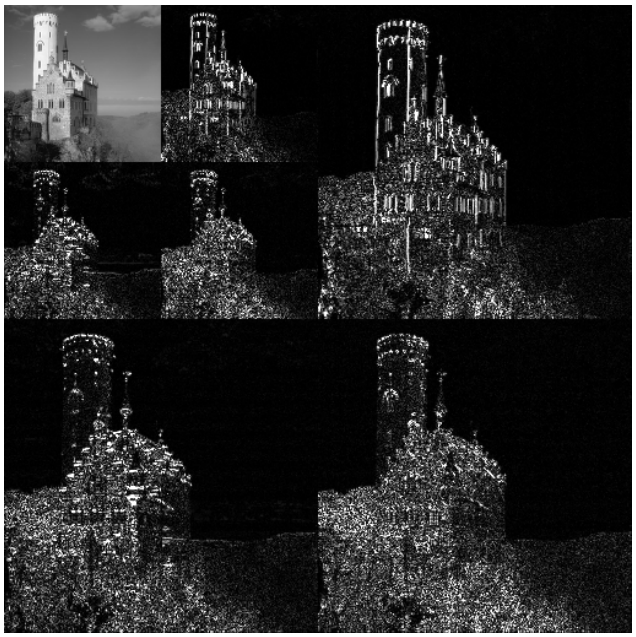


FIGURE 5.20 – Transformée en ondelettes multi-échelle.

Dans ce cas, une description grossière du signal de départ est agrémentée de détails de plus en plus fins ce qui explique sont utilisation en compression et en nettoyage de signal.

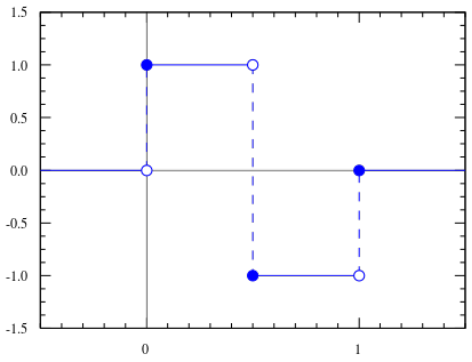
Nous allons tout d'abord introduire la DWT à une dimension qui est suffisante pour définir le mécanisme des ondelettes. Nous introduirons les transformations à deux dimensions dans la section suivante.

La DWT à une dimension décompose un signal donné en deux composantes. Une composante grossière calculée avec un filtre passe-haut et une composante de détail obtenue avec un filtre passe-bas. Ces deux filtres sont appelés récursivement sur les composantes précédemment obtenues ce qui a pour effet de morcelé l'information contenue dans le signal de départ.

Un filtre est dans notre cas une fonction à une dimension appelée ondelette, noté ψ dans la littérature. Cette fonction Ψ doit être de carré sommable ($\Psi \in L^2$) et est généralement oscillante.

Un exemple est l'ondelette de Haar :

$$\Psi(t) = \begin{cases} 1 & \text{pour } 0 \leq t < \frac{1}{2} \\ -1 & \text{pour } \frac{1}{2} \leq t < 1 \\ 0 & \text{sinon} \end{cases} \quad (5.24)$$



La DWT s'interprète comme l'application successive de filtres passe-haut, h, et passe-bas, b, sur la fonction à échantillonner, x. Les coefficients d'approximation, a_k, sont donnés par le filtre passe-haut :

$$a_k = \sum_n x_n \cdot h_{2k-n} \quad \forall k \in [\![0, N[\![\quad (5.25)$$

Et les coefficients de détail, d_k, par le filtre passe-bas :

$$d_k = \sum_n x_n \cdot b_{2k-n} \quad \forall k \in [\![0, N[\![\quad (5.26)$$

Les filtres h et b sont reliés par l'équation :

$$h_{l-1-n} = (-1)^n \cdot b_n \quad \forall n \in [\![0, N[\![\quad (5.27)$$

Où l est la longueur des filtres ou le nombre de points des filtres.

La fonction à reconstruire, x, se calcule à partir des coefficients précédents, a_k et d_k :

$$x_n = \sum_k^N a_k \cdot h_{2k-n} + d_k \cdot b_{2k-n} \quad \forall n \in [\![0, N[\![\quad (5.28)$$

L'application de la transformée en ondelettes discrète produit deux types d'informations :

— Les coefficients d'approximation qui décrivent globalement l'image.
— Les coefficients de détails qui encodent les faibles variations de l'image.

La DWT d'une matrice de $N \times M$ valeurs produits quatre sous matrices de $\frac{N}{2} \times \frac{M}{2}$ valeurs soit une matrice de coefficients d'approximation et trois de coefficients de détails. Une DWT

multi-échelles est constituée d'appels successifs de la DWT de départ sur la matrice des coefficients d'approximation.

Dans la suite nous avons utilisé des transformées en ondelettes uniquement au premier voisinage car les images que nous devons traiter sont peu détaillées. Les images à traiter sont des images calibrées et transformées en matrices par la méthode décrite dans la section 5.2.1.3.

5.4.3.2 Méthode classique

La transformée en ondelettes discrète à deux dimensions, DWT2D, s'obtient en utilisant deux fois la DWT à une dimension. Nous pouvons réécrire les deux filtres à deux dimensions. Dans ce cas, la matrice de signal à décomposée est $x \in M_{NM}(\mathbb{R})$.

Les coefficients d'approximation, a_{kl}, sont déduits de l'équation 5.25 et deviennent :

$$a_{kl} = \sum_i \sum_j x_{ij} \cdot h_{2k-i,2l-j} \quad \forall (k,l) \in \left[\!\left[0, \frac{N}{2}\right[\!\right[\times \left[\!\left[0, \frac{M}{2}\right[\!\right[\tag{5.29}$$

Et les coefficients de détail, d_{kl}, de l'équation 5.26 sont maintenant :

$$d_{kl} = \sum_i \sum_j x_{ij} \cdot b_{2k-i,2l-j} \quad \forall (k,l) \in \left[\!\left[0, \frac{N}{2}\right[\!\right[\times \left[\!\left[0, \frac{M}{2}\right[\!\right[\tag{5.30}$$

Désormais, les filtres $h, b \in M_{NM}(\mathbb{R})$ sont reliés par l'équation :

$$h_{l-1-i,k-1-j} = (-1)^{i+j} \cdot b_{ij} \quad (i,j) \in [\![0, N[\![\times [\![0, M[\![\tag{5.31}$$

La fonction à reconstruire, x, se calcule à partir des coefficients précédents, a_{kl} et b_{kl} et de l'équation 5.28 modifiée :

$$x_{ij} = \sum_k^N \sum_l^M a_{kl} \cdot h_{2k-i,2l-j} + d_{kl} \cdot b_{2k-i,2l-j} \quad (i,j) \in [\![0, N[\![\times [\![0, M[\![\tag{5.32}$$

Dans cette section, les filtres peuvent être représentés par des matrices 2×2. La combinatoire de l'équation 5.31 donne donc une matrice de coefficients d'approximation et trois matrices de coefficients de détail.

La figure 5.21 illustre le processus de DWT sur des matrices dans notre cas d'utilisation.

La figure 5.22 montre le processus de nettoyage d'image classique à la première échelle. Elle consiste à effectuer une DWT sur les images de départ (en haut à gauche) pour obtenir l'image dans l'espace des ondelettes (en haut à droite). Puis, les valeurs de l'espace des ondelettes sont coupées avec un seuil défini préalablement (en bas à gauche), comme pour la méthode historique décrite dans la section 5.4.1. Enfin, la DWT inverse renvoie l'image nettoyée (en bas à droite).

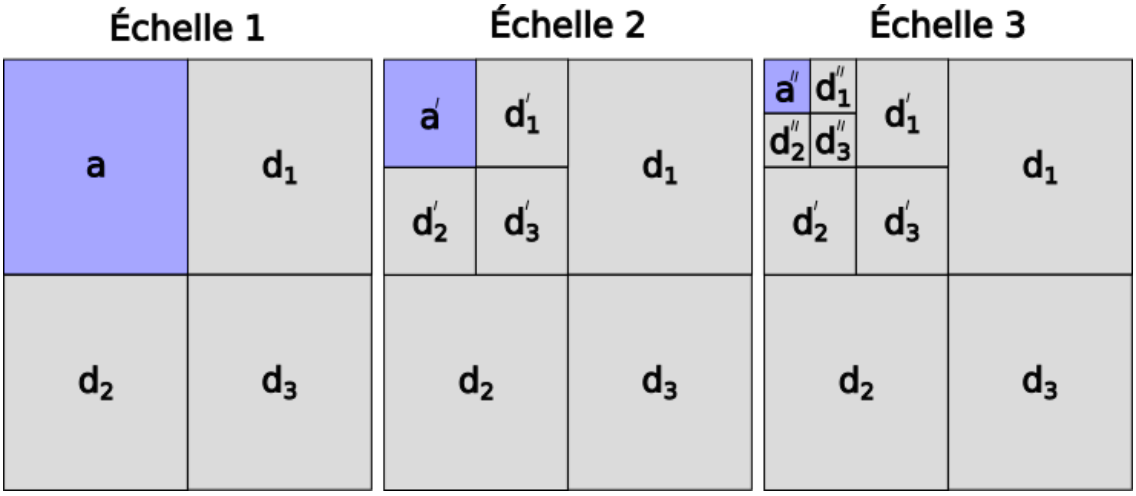


FIGURE 5.21 – Transformée en ondelettes en terme de coefficients. **À gauche**, la première échelle. **Au centre**, la deuxième échelle. **À droite**, la troisième échelle.

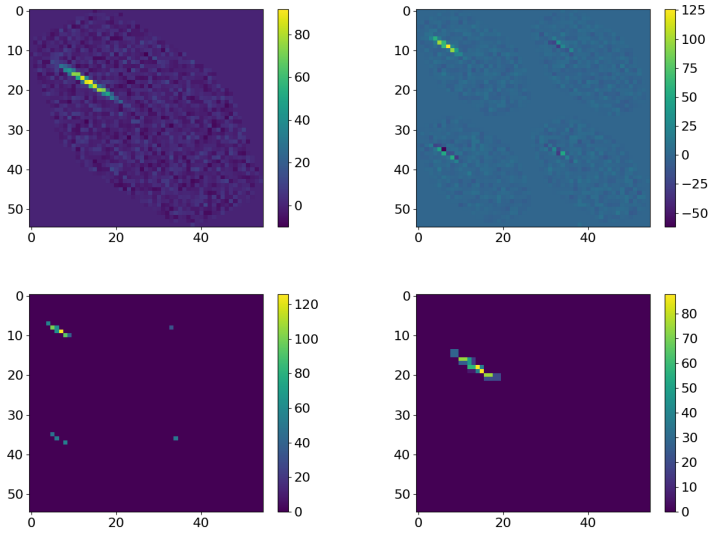


FIGURE 5.22 – Nettoyage de l'image qui utilise la méthode classique des ondelettes. **En haut à gauche**, l'image calibré à nettoyer transformée en matrice (LST-CAM). **En haut à droite**, l'espace des ondelettes à la première échelle associé. **En bas à gauche**, l'espace des ondelettes après application d'un seuil. **En bas à droite**, l'image obtenue par retour dans l'espace de départ.

Cette méthode produit une image nettoyée fortement dégradée (voir figure 5.22 en bas à droite. La morphologie de la gerbe de particules devient chaotique et imprécise comparée à l'image de départ. Malgré le fait que la DWT échantillonne l'image de départ avec un pas de deux pixels, la transformation est lente en temps de calcul.

Après de nombreux tests, nous arrivons à la conclusion qu'appliquer un seuil fixe sur les

coefficients de détail n'est pas efficace.

5.4.3.3 Méthode CTA-LAPP

Au vu des résultats de la section précédente nous pouvons conclure que la première échelle doit être suffisante pour nettoyer les images efficacement. Néanmoins plusieurs choses sont à améliorer :

- Un seuil fixe n'a pas de sens dans notre cas puisque les magnitudes des coefficients changent suivant les images, il serait plus judicieux de l'ajuster au cas par cas.
- Un seuil unique pour tous les coefficients ne peut pas non plus être efficace puisque les coefficients d'approximation et les coefficients de détail n'ont pas la même magnitude. Plusieurs seuils pourraient être utilisés.
- Seuls les coefficients d'approximation semblent contenir les informations que l'on souhaite conserver (voir figure 5.22 en haut à droite). Un masque devrait être appliqué sur ces coefficients puis reporté sur les coefficients de détail, ce qui permettrait de conserver la forme globale de la gerbe de particules dans la caméra.
- La gerbe de particules reconstruite est fortement dégradée (voir figure 5.22 en bas à droite). Une transformée en ondelettes sur une fenêtre glissante serait probablement plus efficace.
- Pour l'instant seuls trois voisins sont pris en compte (voir figure 5.21). Or les pixels des caméras ont six ou quatre voisins immédiats suivant la forme de la caméra. La matrice qui décrit l'ondelette devra prendre cela en compte.
- D'autres ondelettes pourraient être testées voire une ondelette spécifique.

Le calcul automatique du seuil idéal [68], r, dépend de l'intensité, I, de l'image à traiter, sa moyenne μ et son écart type σ :

$$I = \sum_{i}^{N} s_i \tag{5.33}$$

$$r = \sigma\sqrt{\alpha \log_2 I} + \mu \tag{5.34}$$

Où α est une constante qui ajuste l'intensité relative du seuil a appliqué ($\alpha = 2$ dans la version originale de l'article [68]). Cette coupure a été déterminée après plusieurs tests.

L'optimisation des calcul de réductions et de barycentres seront détaillés dans les section 5.5.2 et 5.5.3.

Dans cette version, seuls des coefficients d'approximation sont calculés et aucun sous échantillonnage n'est effectué, de sorte que l'expression 5.29 devient :

$$a_{kl} = \sum_{i}\sum_{j} x_{ij} \cdot h_{k-i,l-j} \quad \forall\,(k,l) \in [\![0, N[\![\times [\![0, M[\![\tag{5.35}$$

Un pixel n'est conservé que si son coefficient d'approximation associé, a_{kl}, est supérieur au seuil r.

Bien que d'avantage de coefficients d'approximation soient nécessaires, le calcul est très rapide. En effet, aucun coefficient de détail n'est calculé et l'image dans l'espace des ondelettes n'est pas conservée. L'empreinte mémoire de cet algorithme est donc moindre.

Une dernière étape consiste à déterminer la meilleure ondelette en se basant sur la précision des paramètres géométriques de l'algorithme Hillas (voir section 5.5). Les 12 paramètres obtenus sur une image non bruitée (vérité terrain) sont comparés à ceux obtenus après nettoyage d'une image bruitée. Les tests sont statistiques et les bruits sont maîtrisés grâce à un outil dans ctapipe [3] qui permet de simuler des gerbes de particules dans les caméras en ajoutant éventuellement un bruit gaussien.
D'autres tests ont été effectué avec les fichiers de simulation Monte-Carlo de CTA pour vérifier la pertinence des résultats de physique.

Les deux meilleures ondelettes construites spécialement pour le nettoyage d'images correspondent au deux types de caméras que nous avons à traiter (hexagonal et carré).

La meilleure ondelette testée sur les caméras hexagonales est :

$$\Psi(x) = \frac{1}{55}\begin{pmatrix} 15 & 15 & 0 \\ 15 & 20 & 15 \\ 0 & 15 & 15 \end{pmatrix} x \tag{5.36}$$

La meilleure ondelette testée sur les caméras carrées est :

$$\Psi(x) = \begin{pmatrix} 0 & 0.4 & 0 \\ 0.4 & 0.4 & 0.4 \\ 0 & 0.4 & 0 \end{pmatrix} x \tag{5.37}$$

La figure 5.23 montre les résultats de cette méthode de nettoyage d'image. L'image obtenue sans bruit (en haut à droite) est comparée à l'image nettoyée par notre méthode de transformée en ondelettes en bas à gauche. On constate que les résultats sont bien meilleurs que ceux de l'algorithme de H.E.S.S.

3. Analyse officielle de CTA

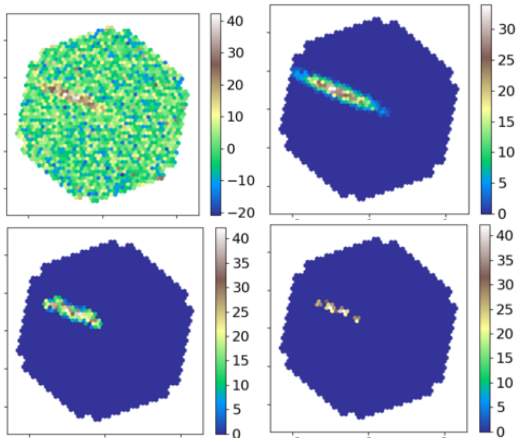


FIGURE 5.23 – Nettoyage de l'image qui utilise la méthode classique des ondelttes. **En haut à gauche**, l'image calibré a nettoyer (LST-CAM). **En haut à droite**, l'image calibré sans bruit (vérité terrain). **En bas à gauche**, image nettoyée obtenue par notre méthode de transformée en ondelettes. **En bas à droite**, image nettoyée obtenue par la méthode de H.E.S.S.

Ce travail a fait l'objet de présentations au sein de la collaboration CTA et une note interne fut rédigée pour l'expliquer en détail [67].

5.5 Calcul des paramètres Hillas

Les images des caméras sont semblables à des ellipses (voir figure 5.25). Une méthode simple et robuste consiste à calculer les paramètres de cette ellipse (position, orientation, asymétrie, etc), appelés paramètres Hillas [69].

Les paramètres Hillas sont au nombre de douze :

— La position de l'ellipse (g_x, g_y).
— L'angle entre cette position à l'axe des abscisses, Φ.
— La longueur de l'ellipse, l.
— La largueur de l'ellipse, w.
— Son orientation, *direction*.
— La quantité de signal qu'elle représente, $\mathcal{A}$.
— Le nombre de pixels sélectionnés, n_p.
— L'asymétrie de l'image, appelée *skewness*.
— L'asymétrie de l'ellipse, appelée $skewness_\Phi$.
— L'excès de Kurtosis évalue à quel point la distribution est piquée sur le grand axe de l'ellipse.
— La séparation de Hofmann évalue à quel point la distribution est piquée sur le petit axe de l'ellipse.

La figure 5.24 illustre les principaux paramètres Hillas.

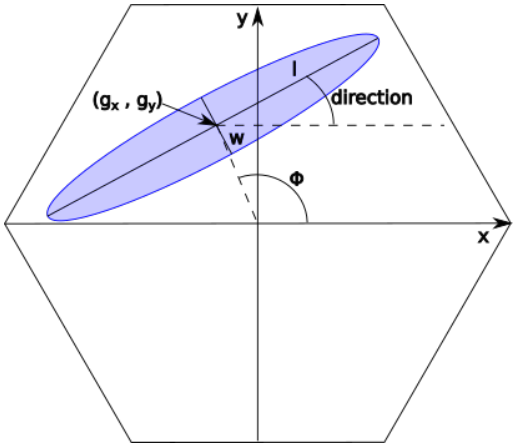


FIGURE 5.24 – Les principaux paramètres Hillas.

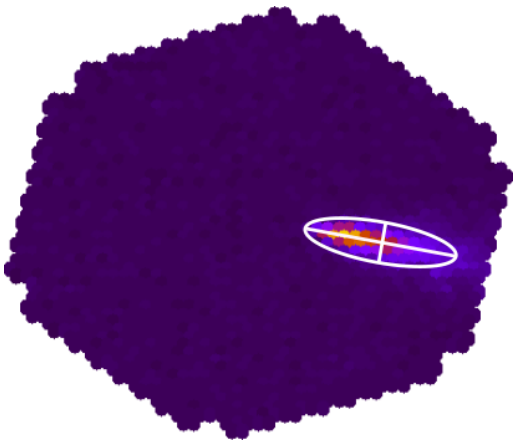


FIGURE 5.25 – Gerbe de particule semblable à une ellipse.

Le calcul des paramètres Hillas, détaillé dans l'annexe A, se décompose en une réduction suivit de trois calculs de moments entrecoupés de deux projections :

- La réduction calcule l'amplitude totale du signal dans la caméra.
- Le premier calcul de moments permet d'obtenir la position de l'ellipse, son orientation, sa longueur et sa largeur dans la caméra.
- Les positions des pixels sont projetées sur le grand axe de l'ellipse.
- Le deuxième calcul de détermine les asymétries de la gerbe et son sens dans la caméra.
- La seconde projection s'effectue sur le sens de la gerbe.
- Le dernier calcul donne l'asymétrie de l'image et un excès appelé Kurtosis.

Tous ces calculs utilisent des tableaux alignés grâce au générateur de format de données présenté au chapitre 3, de sorte qu'aucun peal[4] n'est nécessaire. Cela est d'autant plus important que nous traitons des vecteurs relativement courts.

La vectorisation de ces opérations est importante puisqu'elles représentent 99% de temps de l'analyse non optimisée. Nous avons commencé par vectoriser ces calculs avec les programmes *GCC/G++* [70] mais ils ne traitent pas efficacement la dépendance arrière (voir définition 5.5.1) qui empêche une vectorisation complète du calcul.

Nous avons donc développé une librairie de fonctions intrinsèques pour résoudre ce problème.

4. Ensemble des éléments à traiter dans un tableau avant d'atteindre des éléments alignés sur des frontières de registres vectoriels.

Définition 5.5.1 – **Dépendance arrière**

Comme nous l'avons vu dans la définition 5.3.1, le pipeline CPU permet d'exécuter plusieurs instructions au même instant mais pas à la même étape. En effet, une instruction est exécutée lorsqu'une autre est décodée pendant qu'une troisième est chargée et ainsi de suite. Cela est également valable pour des instructions vectorisées.

Cette technique n'est efficace que lorsque les instructions sont indépendantes entre elles ou lorsqu'une dépendance avant est présente. Cela se produit quand on additionne un élément d'un tableau au suivant avant que celui-ci ne soit modifier par un calcul ($x_i = x_i + x_{i+1}, \forall i \in \mathbb{N}$).

Dans le cas contraire, c'est une dépendance arrière ($x_i = x_{i-1} - x_i, \forall i \in \mathbb{N}$). Dans l'hypothèse où le calcul s'effectue pour un indice, i, croissant, la valeur de l'élément x_{i-1} est nécessaire pour calculer x_i. Or, à l'instant où le calcul de x_i doit s'exécuter, celui de x_{i-1} n'est pas encore fini.

Le CPU a deux méthodes pour résoudre ce problème :

— Insérer une instruction vide ou bulle en attendant que le résultat de l'instruction précédente soit calculée.
— Exécuter une instruction d'un autre programme en attente dans le pipeline.

Dans les deux cas, aucune instruction du programme qui nous intéresse n'est exécutée ce qui ralentit son exécution.

5.5.1 Développement d'une librairie de fonctions intrinsèques

La maîtrise de la vectorisation est essentielle dans ce cas d'optimisation. La ***PLIBS_8*** permet d'aligner automatiquement des tableaux en recherchant l'architecture de la machine hôte (voir section 3.3) qui est décrite par des définitions dans une en-tête C++.

Des librairies existantes comme ***HPX*** [71] ou ***Eigen*** [72] minimisent l'impact des appels aux fonctions intrinsèques en utilisant les *expressions template* du C++. Cela permet de supprimer les temporaires superflus et d'utiliser au mieux une architecture donnée au prix d'une compilation longue [5] et ayant une forte consommation de mémoire [6].

Nous n'avons pas utilisé de librairies existantes afin de maîtriser au mieux le coût des appels aux fonctions intrinsèques et de gérer au mieux l'architecture. Cela évite également d'ajouter une dépendance supplémentaire à l'analyse.

L'architecture de la librairie de fonctions intrinsèques est illustrée par la figure 5.26. Nous avons choisi d'utiliser des fonctions qui inlinent [7] les fonctions intrinsèques de *G++*. Les fonctions qui correspondent à la même commande pour différentes architectures ont le même nom, le choix de l'architecture est fait avec le fichier d'entête généré par la ***PLIBS_8***. De cette manière, les fonctions intrinsèques sont appelées directement sans que la compilation soit ralentie.

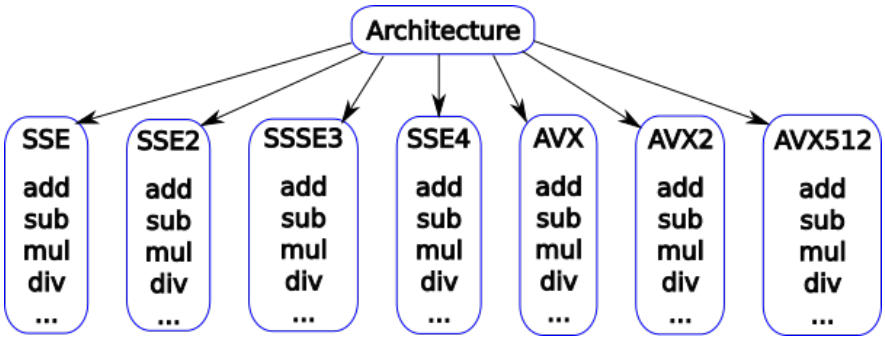


FIGURE 5.26 – Architecture de la librairie de fonctions intrinsèques.

Nous visons uniquement les architectures Intel (de **SSE2** à **AVX512**) pour le moment car elles sont présentes sur tous les centres de calcul que nous utilisons.

5. Plus d'une dixaine de secondes par fichiers.
6. Les programmes compilés avec ***HPX*** peuvent nécessiter plusieurs GO de mémoire pour ne compiler qu'un seul fichier.
7. Une fonction **inline** verra son contenu copié à l'emplacement de son appel. Si cette fonction appelle une autre fonction, seul le deuxième appel coûtera des cycles à l'exécution.

Anecdote 5.5.1 – **Valorisation des fonctions intrinsèques dans CTA**

Différents tests de performance ont été réalisés dans CTA afin d'accélérer un programme écrit en Python. Le premier fut de comparer différentes optimisations de fonctions qui calculent π comme :

$$\pi = \int_0^1 \frac{4}{1+x^2} \mathrm{d}x \tag{5.38}$$

Qui est discrétisé sur 1000 pas d'intégration.

Les différents temps d'exécution des implémentations sont comparées dans la figure 5.27. Les six méthodes passent du Python pur à toutes les techniques utilisables pour accélérer le Python sauf les wrappers C fournis par l'API Python. Le langage C y est également présenté comme référence en terme de vitesse d'exécution.

Afin de convaincre, nous avons écris notre propre fonction à base de fonctions intrinsèques que nous avons appelé *via* un wrapper Python écrit en C. Nous avons gardé l'algorithme de base qui utilise une somme et nous avons cherché à l'optimiser le plus possible ainsi que le calcul :

$$\delta = \frac{1}{1+x^2} \tag{5.39}$$

Nous avons finalement ajouté un septième point au graphique où nous montrons que notre fonction appelée en Python est quatre fois plus rapide que celle écrite en C sur AVX.

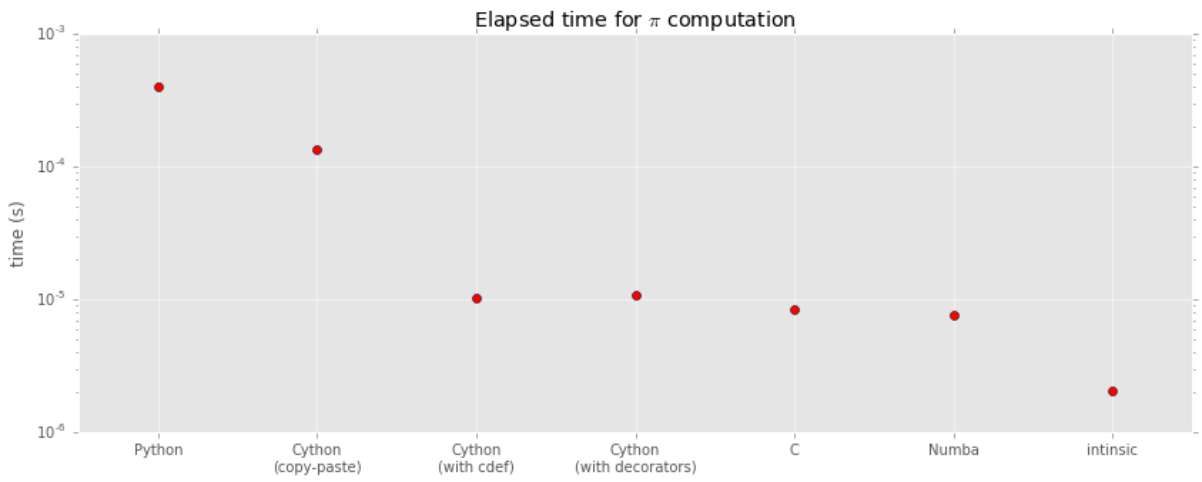


FIGURE 5.27 – Comparaison des temps de calcul de π de différentes libraires.

5.5.2 Optimisation d'une réduction

Une fonction qui réduit un vecteur peut être écrite comme suit en C++ :

```
1 float getSum(const float * tab, size_t size){
2   float sum(0.0f);
3   for(size_t i(0lu); i < size; ++i){
4     sum += tab[i];
5   }
6   return sum;
7 }
```

Cette fonction coûte 2.69842 cy/el à l'exécution. Cette réduction classique sera notre référence dans cette partie.

Il est possible de vectoriser avec *G++* en donnant quelques précisions :

```
1 float getSum(const float * __restrict__ ptab, size_t size){
2   const float* tab =(const float*)__builtin_assume_aligned(ptab,VECTOR_ALIGNEMENT);
3   float sum(0.0f);
4   for(size_t i(0lu); i < size; ++i){
5     sum += tab[i];
6   }
7   return sum;
8 }
```

Le mot clé **__restrict__** assure que la mémoire disponible *via* le pointeur *ptab* n'en recouvre pas d'autres accessibles *via* d'autres pointeurs.

La fonction *__builtin_assume_aligned* indique au compilateur de considérer le pointeur *ptab* comme aligné sur PLIB_VECTOR_SIZE_FLOAT [8] octets. Elle coûte 0.72845 cy/el en utilisant les options de compilation -O3 -ftree-vectorize -march=native -mtune=native pour SSE4. Soit une accélération de 3.839281 par rapport à la réduction classique.

L'extension SSE4 permet d'exécuter un calcul sur 4 float à la fois. Malgré cela, *G++* obtient une accélération inférieure. Lorsque l'on désassemble le binaire créé par *G++*, il apparaît que les calculs sont bien effectués de manière vectorielle. Mais dans le cas d'une réduction, les éléments du registre vectoriel sont sommés à chaque itération car il considère qu'une itération doit retourner un scalaire, ce qui dégrade naturellement la vectorisation puisque cette somme ne devrait être faite qu'une fois la boucle terminée. Ce problème peut être résolu en programmant directement avec des fonctions intrinsèques. Dans ces conditions, seul le problème de dépendance arrière doit être résolu.

8. Cette définition provient du fichier d'entête généré par la ***PLIBS_8***.

Le résultat d'un calcul ne peut être utilisé qu'un laps de temps après que son exécution se soit terminée. Ce temps est incompressible et dépend de l'électronique de l'architecture hôte. Il est, par conséquent, impossible de le supprimer avec une option de compilation.

La solution consiste à utiliser plusieurs accumulateurs [9] indépendants et d'entrelacer leurs calculs respectifs afin d'amortir le temps d'attente incompressible évoqué précédemment.

La figure 5.28 illustre le gain de temps que permet l'entrelacement des accumulateurs. Ici, nous considèrons [10] qu'un calcul coûte 1 cycle, que la mise à disposition de ce résultat coûte également un cycle, et que le temps incompressible écoulé entre ces deux étapes est de 2 cycles.

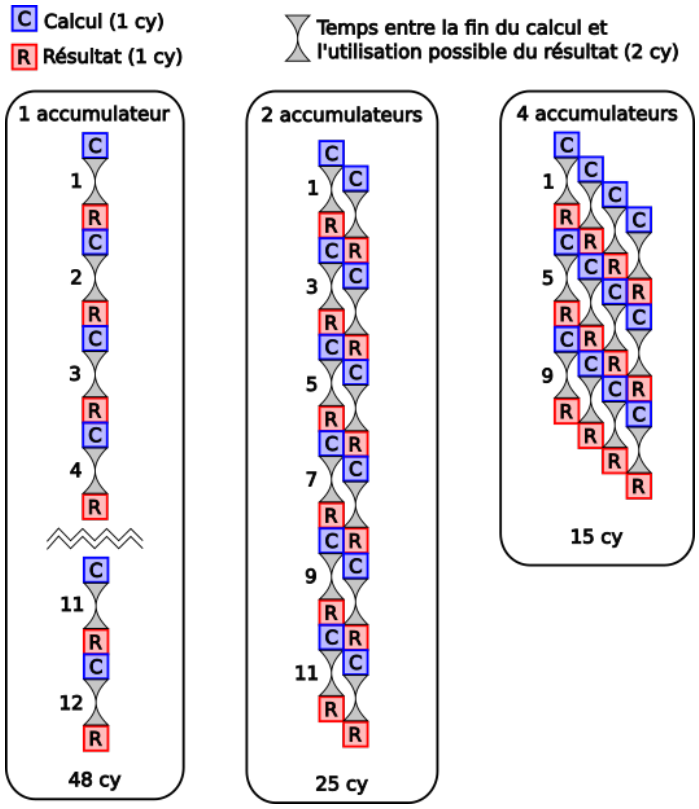


FIGURE 5.28 – Illustration du gain de temps apporté par l'entrelacement des accumulateurs.

Ce pipeline est simplifié au maximum : un calcul vectorisé, un temps d'attente, un résultat et le calcul suivant ne peuvent être exécutés que lorsque l'écriture du résultat précédent est terminée [11]. Une dernière règle interdit deux calculs simultanés car dans notre exemple nous

9. Désigne la variable dans laquelle on somme toutes les autres valeurs.

10. L'exemple des temps est ici purement arbitraire afin d'illustrer la méthode de l'entrelacement tout en simplifiant les calculs d'accélération.

11. Les pipelines modernes découpent leur exécution en 40 ou 80 étapes, contre les 5 étapes de notre exemple (voir figure 5.16). Ils permettent également de réutiliser le résultat d'un calcul avant qu'il soit écrit mais un temps incompressible existe toujours entre la fin du calcul et la disponibilité du résultat.

n'avons qu'une addition [12].

Le traitement complet d'un élément (un registre vectoriel) coûte 4 cycles dans notre exemple donc 12 instructions s'exécutent en 48 cycles (temps de référence).

Si nous utilisons deux accumulateurs, la deuxième instruction calcule un résultat pendant que la première attend de stocker le sien. De cette manière, la fraction de temps où il n'y a aucun calcul diminue. Le temps d'exécution est réduit à 25 cycles, soit un facteur d'accélération de 1.92.

Avec quatre accumulateurs, un calcul est pratiquement effectué à chaque cycle, sauf les trois derniers. Le temps d'exécution atteint 15 cycles, ce qui est 3.2 fois plus rapide que notre temps de référence.

Les temps donnés précédemment ne sont représentatifs que du calcul de 12 instructions, ce qui est peu. Conformément à la figure 5.28, le facteur d'accélération, α, obtenu en fonction du nombre d'instructions exécutés, x, est donné par l'équation :

$$\alpha = \frac{4x}{\frac{4x}{s} + s - 1} \tag{5.40}$$

Où $1 \leq s \leq 4$ est le nombre d'accumulateurs utilisés.

La figure 5.29 illustre les accélérations attendues basée sur l'équation 5.40 pour notre exemple. Les asymptotes des courbes montrent que le facteur d'accélération tend vers le nombre d'accumulateurs que l'on utilise.

Dans notre exemple, nous ne prenons pas en compte le fait que dans la réalité le nombre de registres vectoriels est limité.

12. Les CPUs actuels sont capables d'exécuter plusieurs calcul différents en même temps car dans ce cas ils n'utilisent pas les mêmes composants de l'Unité Arithmétique et Logique (ALU).

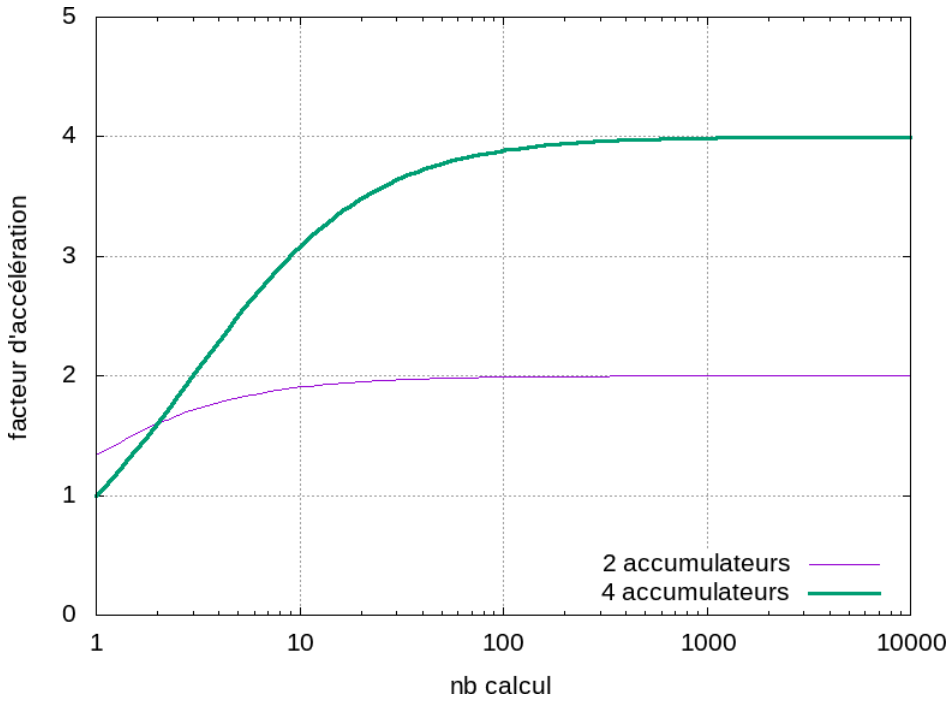


FIGURE 5.29 – Illustration du gain de temps apporté par l'entrelacement des accumulateurs dans le cas d'une réduction.

Le code suivant montre comment nous avons utilisé notre librairie de fonctions intrinsèques pour écrire une réduction de vecteur qui utilise des accumulateurs entrelacés :

```
1  float getSum(const float * tab, size_t size){
2    register size_t step = (PLIB_VECTOR_SIZE_FLOAT*2lu);
3    size_t nbVect = size / step;
4
5    PRegVecf vecRes1 = plib_load_ps(tab);
6    PRegVecf vecRes2 = plib_load_ps(tab + PLIB_VECTOR_SIZE_FLOAT);
7    size_t i;
8    for(i = 1lu; i < nbVect; ++i){
9      register size_t shift = PLIB_VECTOR_SIZE_FLOAT*2lu*i;
10     vecRes1 = plib_add_ps(vecRes1, plib_load_ps(tab + shift));
11     vecRes2 = plib_add_ps(vecRes2, plib_load_ps(tab + shift +
           PLIB_VECTOR_SIZE_FLOAT));
12   }
13   float tmp[PLIB_VECTOR_SIZE_FLOAT];
14   PRegVecf vecRes = plib_add_ps(vecRes1, vecRes2);
15   return reduceRegVecf(vecRes, tmp);
16 }
```

Ici, deux accumulateurs sont utilisés. Lorsque qu'ils sont plus nombreux (4 ou 8), ils sont

sommés deux par deux en arbre afin de limiter le nombre de calcul.

Les tableaux 5.1 et 5.2 montrent les accélérations que nous avons obtenu par l'entrelacement des accumulateurs sur les architectures SSE4 et AVX respectivement. On note que lorsqu'un seul accumulateur est utilisé, l'accélération est supérieure à 4 sur SSE4 à cause du problème de dépendance arrière de *GCC* évoqué au tout début de cette section.

Entrelacement	Vitesse (cy/el)	Accélération
1	0.616841	4.374579
2	0.338947	7.961185
4	0.226990	11.887836
8	0.226675	11.904356

TABLE 5.1 – Accélérations obtenues par entrelacement sur des tableaux float sur une architecture SSE4.

Entrelacement	Vitesse (cy/el)	Accélération
2	0.172954	15.601951
4	0.115568	23.349197
8	0.11379	23.714034

TABLE 5.2 – Accélérations obtenues par entrelacement sur des tableaux float sur une architecture AVX.

Nous observons que plus les calculs sont entrelacés et plus l'utilisation du pipeline CPU est efficace car leur proportion augmente. Cependant, il ne faut pas utiliser trop d'accumulateurs pour qu'ils restent stockés dans les registres vectoriels. En effet, ces registres sont en nombre limité sur les architectures (8 pour SSE4, 16 pour AVX2 et 32 pour AVX-512).

Nous avons finalement comparé les temps de calcul de notre réduction optimisée en Python *via* l'implémentation la plus rapide que nous avons d'un wrapper (voir chapitre 3).

La figure 5.30 montre les vitesses d'exécution de différentes méthodes de réduction en fonction du nombre d'éléments.

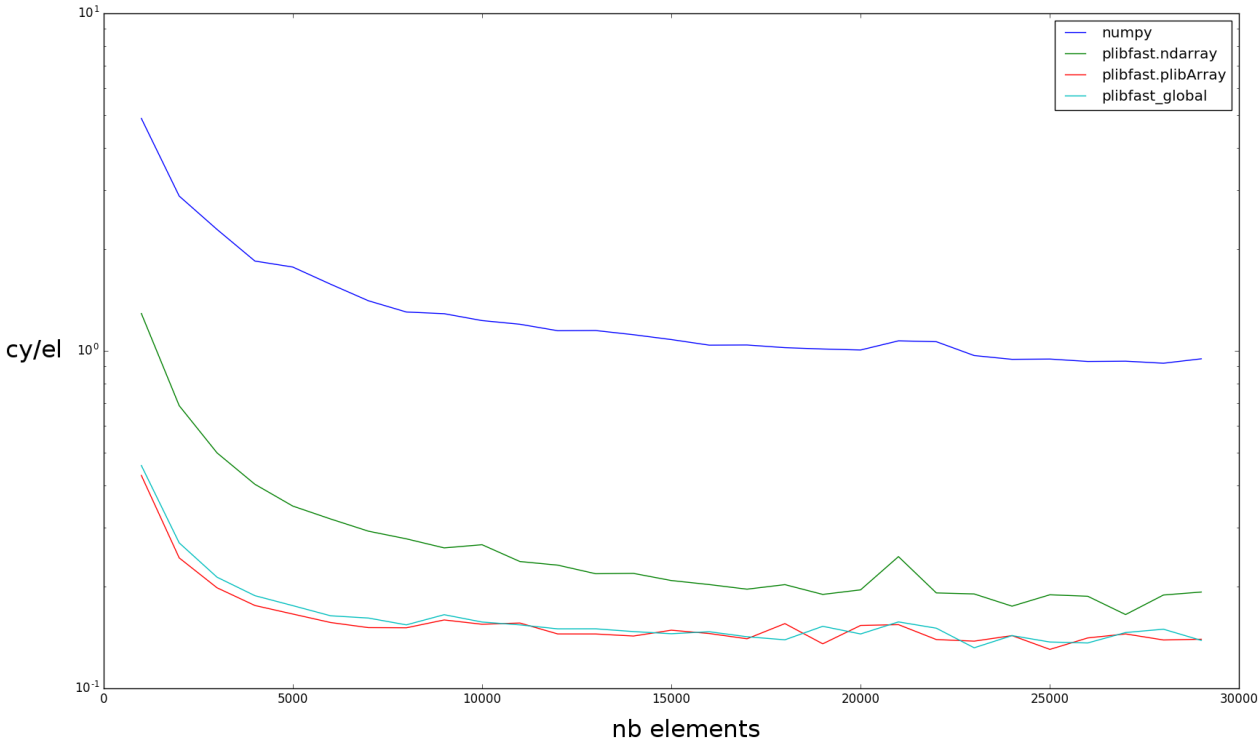


FIGURE 5.30 – Vitesses en cycles par éléments du calcul d'une réduction de différentes implémentations en fonction du nombre d'éléments.

• La première courbe décrit les performances de la librairie ***numpy***, compilée avec la libraire MKL (Math Kernel Library [73]) fournie par Intel, qui est la plus utilisée en Python. Elle est notre référence en terme de performances de calculs.

• La deuxième montre la vitesse de traitement obtenue avec un wrapper Python qui hérite de la classe ***ndarray*** qui décrit des tableaux en Python.

• La troisième utilise notre implémentation du wrapper qui produit des tableaux ***ndarray*** sans hériter de cette classe. Nous observons un écart net par rapport à la précédente, dû à la simplicité de la méthode qui améliore la vitesse d'exécution. Cette méthode est un ordre de grandeur plus rapide que l'implémentation de ***numpy***.

• La dernière courbe est une variante de la précédente mais l'implémentation utilise une variable globale afin de passer des paramètres sans appel Python. Nous voulions être sûr que le temps utilisé pour passer les paramètre à la fonction Python n'était pas important, ce qui est bien le cas puisque cette courbe est compatible avec la précédente.

Nous avons également tester l'implémentation de la réduction de matrice (voir figure 5.31) en adaptant la méthode la plus rapide décrite précédemment. Notre implémentation est quasiment un ordre de grandeur plus rapide que celle de ***numpy***.

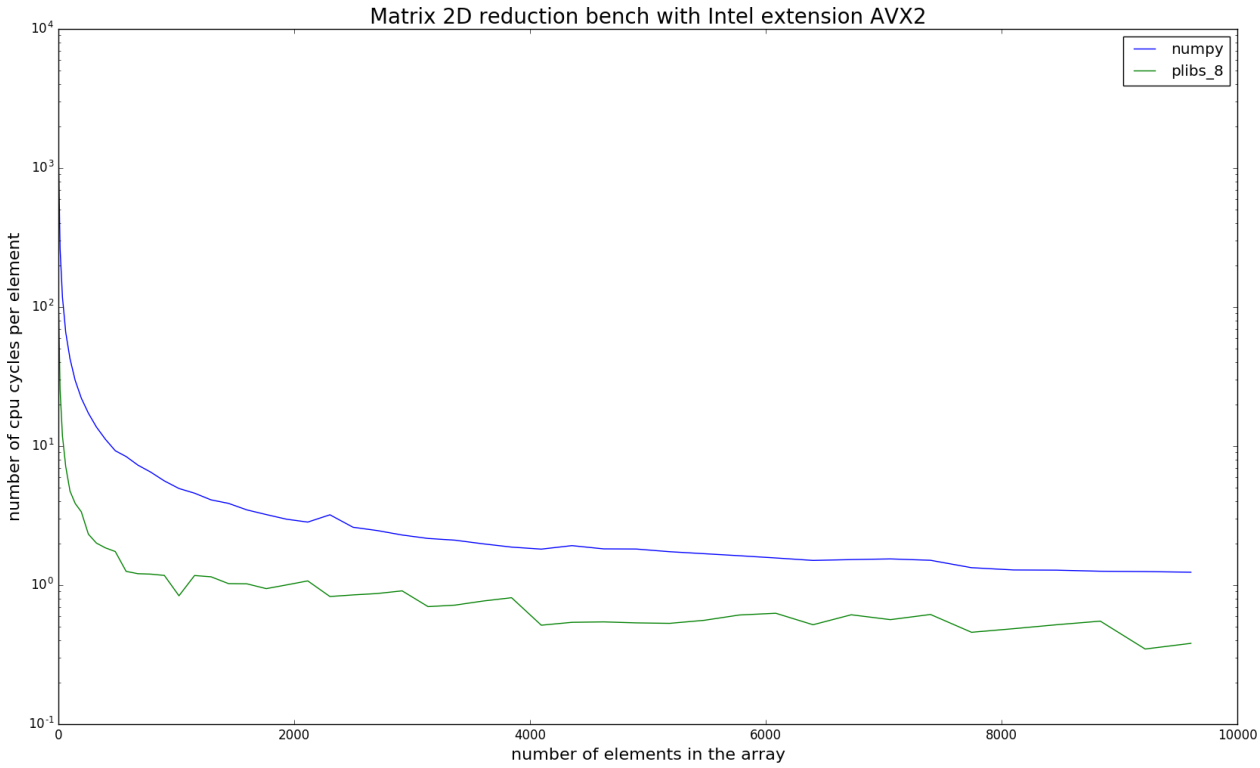


FIGURE 5.31 – Vitesses en cycles par éléments du calcul d'une réduction de matrices entre ***numpy*** (bleu) et notre implémentation (vert) en fonction du nombre d'éléments.

5.5.3 Optimisation d'un calcul de barycentre

Nous pouvons optimiser le calcul de barycentre de la même manière que nous avons optimisé celui de la réduction dans la section précédente.

La figure 5.32 montre l'intérêt de l'entrelacement des calculs dans un cas plus complexe que précédemment. Un calcul de barycentre est une somme de produits, or ces deux calculs n'ont pas le même temps d'exécution : typiquement 4 cycles pour une addition et 6 cycles pour une multiplication. Comme nous l'avons évoqué précédemment, nous ne pouvons pas effectuer deux calculs vectorisés identiques au même instant. Mais il est tout de même possible dans notre cas d'exécuter une addition au même instant qu'une multiplication car ces deux opérations ne concernent pas les mêmes composants de l'ALU [13].

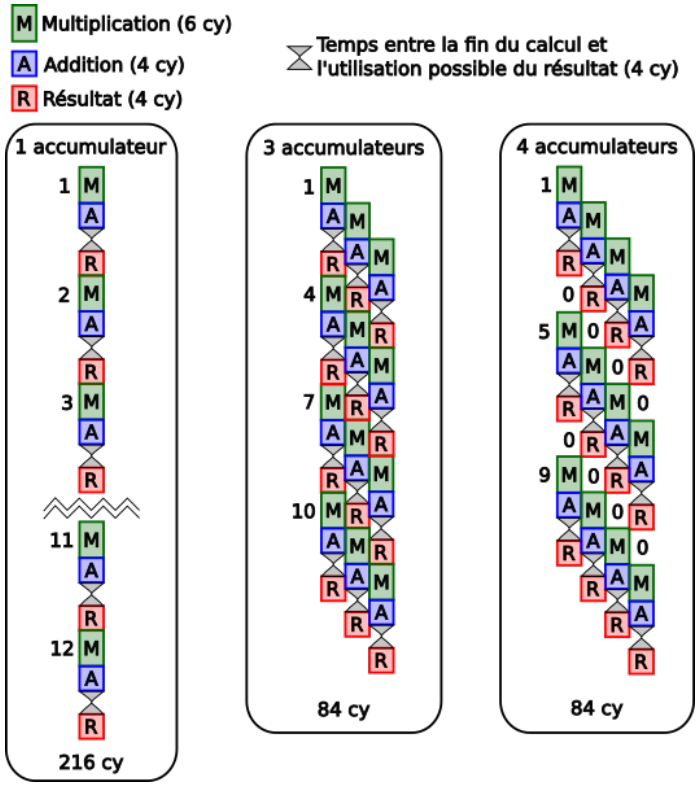


FIGURE 5.32 – Illustration du gain de temps apporté par l'entrelacement des accumulateurs dans le cas d'un calcul de barycentre.

Dans cet exemple, le temps d'attente entre l'addition et la disponibilité du résultat est de 4 cycles et sa mise à disposition prend également 4 cycles. Une itération complète coûte 18 cycles, dont seulement 10 cycles de calcul.

L'exécution de 12 calculs coûte donc 216 cycles (référence). L'utilisation de 2 accumulateurs la ramène à 114 cycles, soit un facteur d'accélération de 1.89. Trois accumulateurs

13. Unité Arithmétique et Logique.

permettent de réduire ce temps à 84 cycles, ce qui est dans notre cas le temps optimal puisque toutes les multiplications sont exécutées sans aucune attente. Nous avons alors un facteur d'accélération de 2.57. On constate qu'un calcul avec 4 accumulateurs ne modifie pas sa vitesse d'exécution (toujours 84 cycles) car le nombre maximum de calculs possible est déjà atteint avec 3. Il n'est donc pas nécessaire d'utiliser d'avantage d'accumulateurs. En revanche, l'empreinte mémoire augmente d'un tiers par rapport à l'utilisation de 3 accumulateurs.

Comme précédemment, ces accélérations ne reflètent pas rigoureusement ce que l'on peut attendre car nous n'avons pris en compte que l'exécution de 12 calculs.

En prenant en compte x calculs à effectuer dans la figure 5.32, le facteur d'accélération, α, du calcul de barycentre est décrit par l'équation :

$$\alpha = \frac{18x}{\frac{18x}{s} + 6 \cdot (s-1)} \tag{5.41}$$

Où $1 \leq s \leq 3$ est le nombre d'accumulateurs.

La figure 5.33 montre l'évolution du facteur d'accélération calculé à partir de l'équation 5.41 pour deux ou trois accumulateurs (meilleur cas).

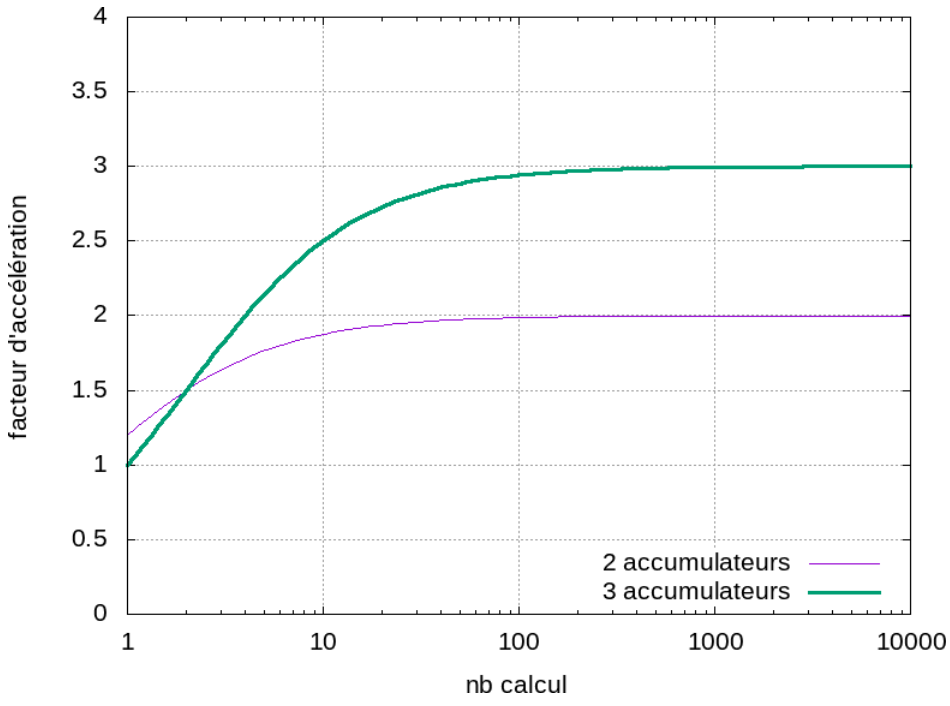


FIGURE 5.33 – Facteurs d'accélération que l'on obtient théoriquement dans l'exemple du calcul de barycentre.

Puisque le calcul est plus complexe qu'une réduction, le nombre d'accumulateurs entrelacés peut être réduit. Le tableau 5.3 montre la vitesse que nous obtenons en vectorisant manuellement calcul grâce aux fonctions intrinsèques.

Moments	Architecture	Vitesse (cy/el)
1^{er} 1D	SSE4	0.457595
$\overline{x}$	AVX	0.179133
1^{er} 2D	SSE4	0.910941
$\overline{x}, \overline{y}$	AVX	0.343091
1^{er} et 2^{nd} 2D	SSE4	1.29088
$\overline{x}, \overline{y}, \overline{x^2}, \overline{y^2}, \overline{xy}$	AVX	0.599513

TABLE 5.3 – Exemple des vitesses de calcul des premier et second moments que nous obtenons sur les architectures SSE4 et AVX.

Dans ces résultats, un élément représente un seul élément d'un tableau lue à une dimension ou un couple de valeur (x, y) dans le cas à deux dimensions.

Le temps de calcul d'une moyenne à deux dimensions est le double de celui à une dimension car deux fois plus de chargements mémoire sont nécessaires et ces calculs sont indépendants. Au contraire, le calcul des moyennes et des variances à deux dimensions coûtent proportionnellement moins de cycles car ces chargements sont mutualisés.

5.5.4 Application à Hillas

L'optimisation du calcul des paramètres Hillas consiste tout d'abord à mutualiser au maximum les chargements des éléments nécessaires à celui-ci. Pour ce faire, les projections décrites dans la section 5.5 sont incorporées dans les calculs de moments suivants.

Ainsi, le premier calcul de moments est indépendant $(\overline{x}, \overline{y}, \overline{x^2}, \overline{y^2}, \overline{xy})$ et son temps correspond donc à 1.29088 cy/el pour SSE4 et 0.599513 cy/el pour AVX (voir tableau 5.3).

Le calcul de moments suivant est fusionné avec la projection sur le paramètre ϕ. Trois moments sont calculés sur x et quatre sur y, il n'est donc pas nécessaire d'entrelacer ces calculs car ceux-ci sont naturellement coûteux en temps. Ce calcul coûte 2.84263 cy/el sur SSE4 et 1.31694 cy/el sur AVX.

La projection sur la direction, d, de la gerbe de particules dans la caméra est incorporée au calcul de moments suivant. Les quatre premiers moments sont calculés sur x, il n'est donc pas entrelacés pour les mêmes raisons que précédemment. Les vitesses d'exécution sont de 1.90996 cy/el sur SSE4 et 0.851525 cy/el sur AVX.

Le tableau 5.4 résume les différentes vitesse d'exécution obtenues pour les différentes étapes du calcul des paramètres Hillas. Le total des cycles des différentes étapes est inférieure à celle du Hillas complet car des calculs complémentaires sont effectués pour déterminer les autres paramètres Hillas (longueur, largeur, direction, etc).

	Vitesse SSE4 (cy/el)	Vitesse AVX (cy/el)
Étape 1	1.29088	0.599513
Étape 2 (projection ϕ)	2.84263	1.31694
Étape 3 (projection d)	1.90996	0.851525
Total	6.04347	2.767978
Notre Hillas optimisé	6.39931	2.98499

TABLE 5.4 – Résumé des vitesses d'exécution des étapes optimisées du calcul des paramètres Hillas.

5.5.5 Facteur d'accélération

Le tableau 5.5 compare les facteurs d'accélération atteints par notre optimisation du calcul des paramètres Hillas à la méthode utilisée dans l'analyse de données de H.E.S.S.

Tout d'abord, on constate que l'optimisation **seule** du format de données que nous avons présenté au chapitre 3 apporte un facteur d'accélération de 40 comparé à l'implémentation qui utilise le format de données de H.E.S.S. Dans ce cas, nous avons adapté l'algorithme pour qu'il utilise des tableaux mais sans utiliser d'optimisation ni d'option de compilation différente.

Nous obtenons finalement un calcul total des paramètres Hillas 332 fois plus rapide que l'original en SSE4 et 712 fois plus rapide en AVX.

	Vitesse (cy/el)	Accélération
Hillas Initial	2125.5	1
Hillas Initial format de données efficace	53.1375	40.0
Notre Hillas optimisé SSE4	6.39931	332.14516
Notre Hillas optimisé AVX	2.98499	712.06268

TABLE 5.5 – Comparaison entre nos différentes optimisations du calcul des paramètres Hillas.

La figure 5.34 montre les vitesses d'exécution des différentes implémentations du calcul des paramètres Hillas sur une architecture AVX2. L'implémentation C++ est 600 fois plus rapide que Python tandis que le facteur deux de vitesse entre l'implémentation C++ et son appel en Python *via* l'API se réduit avec le nombre d'éléments à traiter. On notre également que le ralentissement dû au changement du cache L1 au L2 n'est plus aussi drastique que dans les architectures précédentes. Il est au contraire progressif.

La décroissance que l'on observe au début de chaque courbe n'est pas due à un unique facteur. Lorsque les tableaux sont petits, le préchargement des données est moins amorti, ce qui explique la courbe C++. Le coût de l'appel d'une fonction Python est très élevé (300 cycles pour une fonction appelée *via* un wrapper). Il est donc dominant lorsque le nombre d'éléments à traiter est faible ce qui explique la courbe du wrapper Python. Quand à la courbe en Python pure, le surcoût dû à l'interprétation du Python lui-même est considérable face au calcul à un point tel que le passage entre les caches L1 et L2 est visible. Ceci est causé par la complexité du format des données dans l'analyse officielle de CTA qui dégrade le préchargement des données.

Ce travail a été présenté en réunion de collaboration CTA et a mis en lumière le fait que le traitement des données en temps réel nécessiterait beaucoup moins d'ordinateurs que ce qui avait été prévu au départ. Une centaine d'ordinateurs par télescope contre deux dixaines pour la totalité du site sud (qui produira le plus de données).

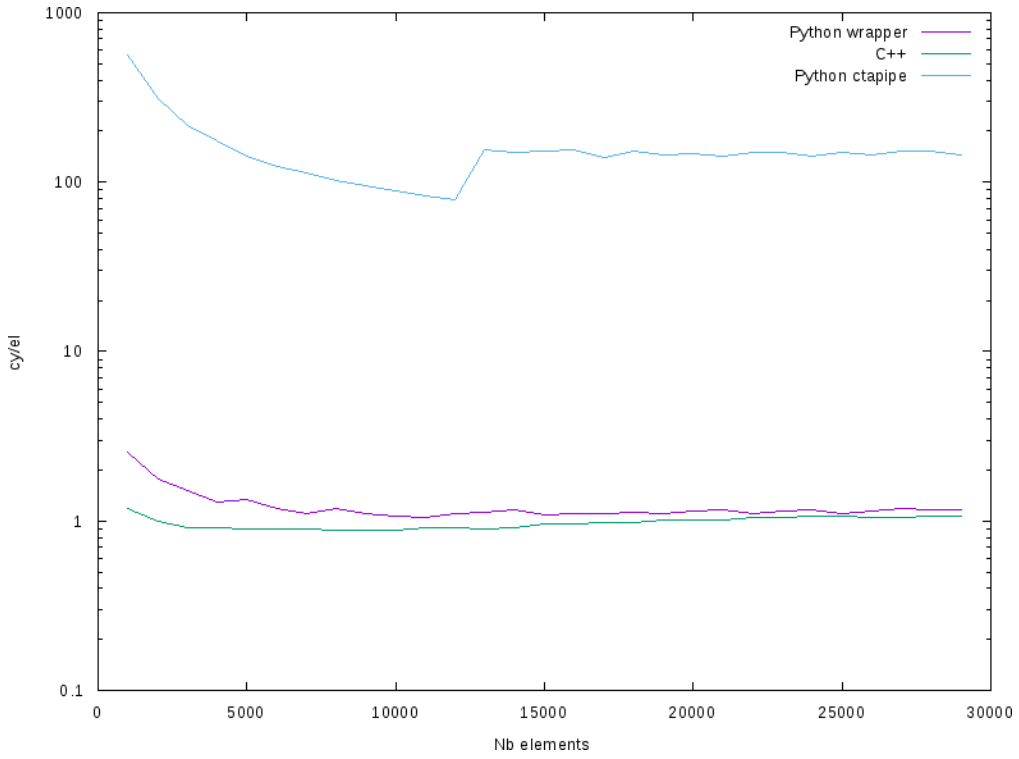


FIGURE 5.34 – Vitesses d'exécution des différentes implémentations du calcul des paramètres Hillas.

Détail technique 5.5.1 – **Double précision ou simple ?**
Le calcul des paramètres Hillas provient de l'analyse de H.E.S.S. Il utilisait à l'origine une double précision. L'optimisation de ce calcul fut de passer en simple précision. En faisant cela des problèmes d'arrondis furent mis en évidence. La simple précision n'était nullement en cause dans ce cas précis. Par soucis de simplicité la position des pixels (à l'origine en mètre) était convertie en radian (chaque pixel voit un certain angle dû au miroir). Cette conversion produisait des nombres bien plus petits que l'unité ce qui était à l'origine d'erreurs de calcul (même en double précision). L'utilisation de position de pixels en mètre a permis de résoudre ce problème, même en simple précision.

5.6 Reconstruction stéréoscopique

La reconstruction stéréoscopique consiste à déterminer les caractéristiques physiques de la particule initiale en combinant les paramètres Hillas qui décrivent les images de tous les télescopes de l'événement. En effet, la prise d'images simultanées de la même gerbe de particules augmente la précision de sa reconstruction.

Dans la section précédente, qui calcule les paramètres Hillas, toutes les images peuvent être traitées simultanément, car ces calculs sont indépendants. c'est pour cela que ces paramètres Hillas doivent être triés pour la reconstruction stéréoscopique.

Tout d'abord, la reconstitution des événements regroupe tous les paramètres Hillas d'un même événement ensemble (section 5.6.1). Ensuite, la reconstruction géométrique détermine la direction et le paramètre d'impact de la gerbe de particules au sol (section 5.6.2). Puis, la reconstruction en énergie évalue l'énergie de la particule initiale (section 5.6.4). Enfin, la discrimination calcule la probabilité que la particule initiale soit un rayon gamma (section 5.6.5).

5.6.1 Reconstitution des événements

L'analyse de données de CTA est constituée de deux parties distinctes :

- Le traitement des images des télescopes (intégration, calibration, nettoyage, paramètres Hillas) qui est massivement parallèle.
- La reconstitution stéréoscopique qui utilise ces paramètres (reconstruction géométrique, en énergie, discrimination) qui est également massivement parallèle mais sur une structure de données différente.

Lors de la prise de données, les images seront stockées par télescope, c'est à dire que les données de chaque télescope ne seront composées que de ces propres images. Cette méthode simplifie l'écriture des fichiers de données brutes et permet une analyse de données massivement parallèle puisque toutes les images de tous les télescopes sont indépendantes entre elles. Des temporaires peuvent également être utilisés tout au long de l'analyse d'un télescopes et ainsi réduire considérablement son empreinte mémoire.

Cependant, l'étape de reconstruction stéréoscopique requière que les différents paramètres décrivant les images soient dans un même événement en terme de structure de données. C'est pour cette raison que nous avons deux formats de données pour stocker les paramètres Hillas (voir section 5.1.5).

Il est donc nécessaire de convertir le format de données "par télescope" en format "par événement". La reconstitution des événements est illustrée par la figure 5.35.

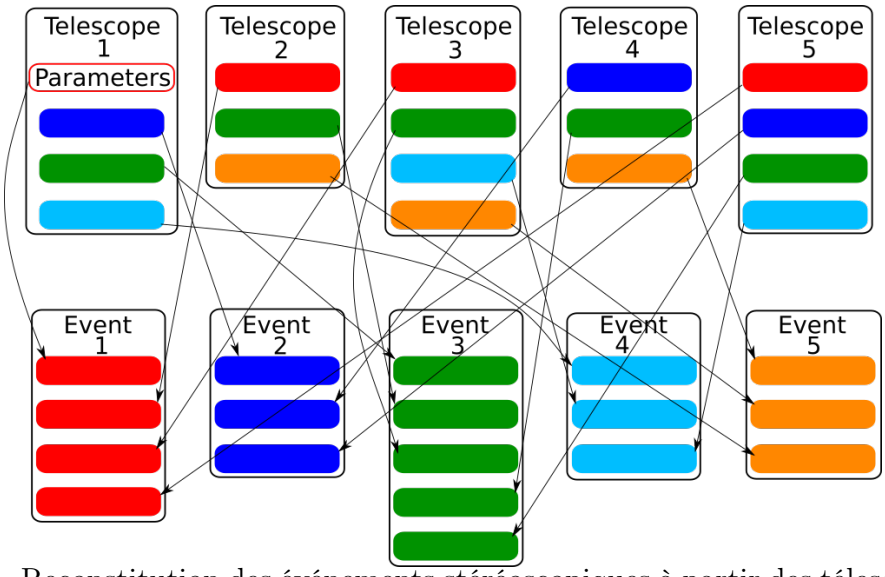


FIGURE 5.35 – Reconstitution des événements stéréoscopiques à partir des télescopes séparés.

Les événements sont reconstitués à l'aide d'un arbre binaire sur les identifiants [14] des images des télescopes. Cela garantit une vitesse d'exécution rapide quelque soit le nombre d'images à trier. Pour le moment, les fichiers sont relativement petits, mais il serait possible d'optimiser d'avantage cette méthode en l'appliquant à des blocs définissants des intervalles sur les identifiants car les images de chaque télescopes sont dans un ordre croissant en identifiants.

Ceci permet une vitesse d'exécution plus rapide qu'un tri rapide couplé [15] à un tri à bulles [16], ce que nous avions implémenté auparavant.

5.6.2 Reconstruction géométrique

La direction de la particule initiale ou le paramètre d'impact de la gerbe de particules au sol sont calculés par le même algorithme. Néanmoins, sa précision influence fortement la suite de la reconstruction stéréoscopique, nous avons donc utilisé et comparé différents algorithmes.

Tous ces algorithmes fournissent différentes manières de calculer l'intersection de plusieurs droites dans un plan.

14. Numéro de l'événement stocké dans un long unsigned int.
15. L'algorithme de tri le plus rapide sur CPU lorsque plus de 15 éléments sont à trier. Il traite N éléments en $N \log N$ opérations.
16. Algorithme de tri plus rapide que le tri rapide si le nombre d'élément est inférieur à 15. Il traite N éléments en $N\,(N-1)$ opérations.

Seul, le calcul des droites diffère suivant l'application. Le calcul de la direction initiale de la particule utilise les coordonnées angulaires données par le champ de vue des télescopes (azimuth [17], altitude [18]). Celui du paramètre d'impact utilise en plus les coordonnées du télescope (voir figure 5.36).

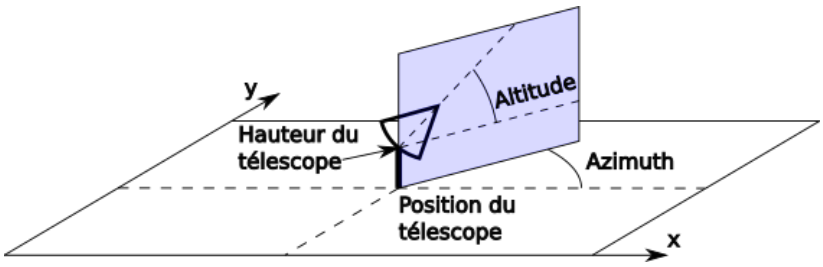


FIGURE 5.36 – Coordonnées utilisées pour décrire un télescope.

5.6.2.1 Projection des directions dans les caméras au sol

La figure 5.37 illustre la projection d'une ellipse dans une caméra au sol. Lorsque la gerbe de particules se déploie dans l'atmosphère, sa lumière (la lumière Cherenkov) est concentrée par un miroir (**1**) dans une caméra (**2**). La position et la direction de l'ellipse dans la caméra peuvent être exprimées en fonction des axes **u** et **v** de celle-ci dans le repère du site. Enfin, la position et la direction de cette ellipse projetée au sol peuvent être déterminées (**3**).

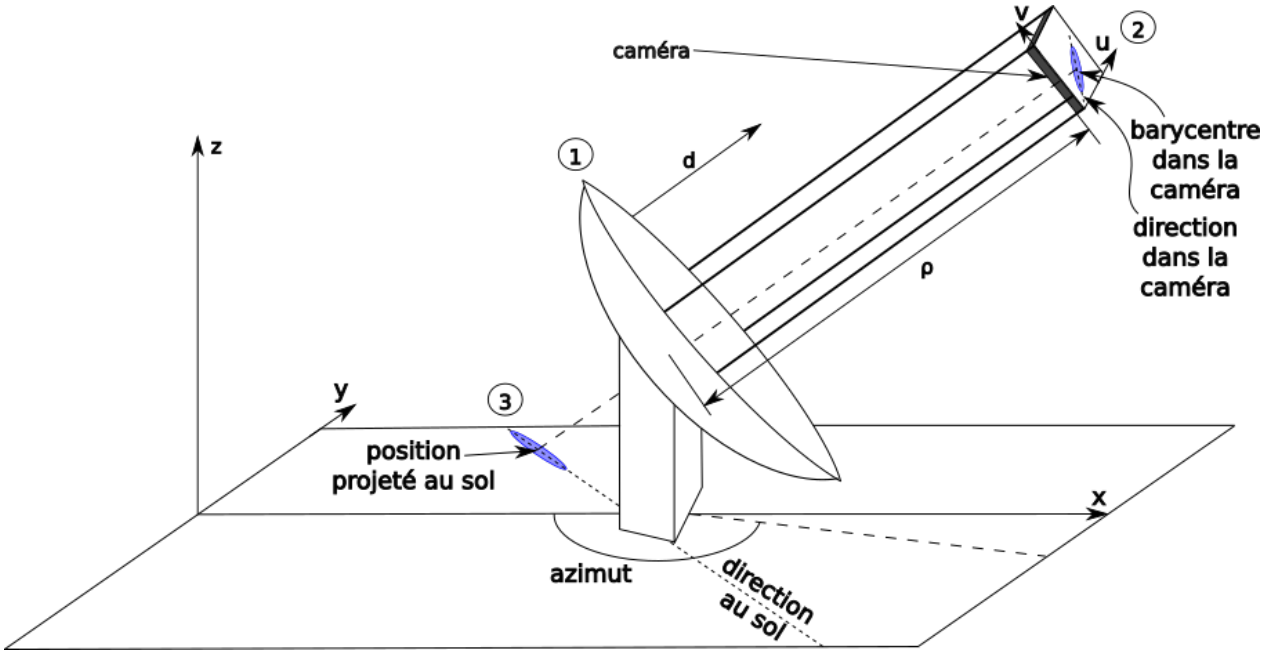


FIGURE 5.37 – Projection d'une ellipse dans la caméra au sol.

17. Azimuth : angle des télescopes avec la direction nord.
18. Altitude : Angle des télescopes avec le sol.

Les origines sont aux centres des sites et sont décrites par des coordonnées au sol ; x en direction du nord et y vers l'ouest. La conversion des positions dans la caméra nécessite l'azimut, az, et l'altitude, alt en radian, et la distance entre le miroir et la caméra, ρ. L'axe de visée du télescope, d, s'exprime dans les coordonnées du site comme :

$$\mathbf{d} = \begin{pmatrix} \rho \cos(az) \cos(alt) \\ \rho \sin(az) \cos(alt) \\ \rho \sin(alt) \end{pmatrix} \tag{5.42}$$

Les vecteurs $\mathbf{u}$ et $\mathbf{u}$ qui portent la caméra sont :

$$\mathbf{u} = \begin{pmatrix} \sin(az) \\ \cos(az) \\ 0 \end{pmatrix} \tag{5.43}$$

Avec :

$$\mathbf{v} = -\mathbf{u} \wedge \mathbf{d} \tag{5.44}$$

D'où :

$$\mathbf{v} = \begin{pmatrix} -\rho \cos(az) \sin(alt) \\ \rho \sin(az) \sin(alt) \\ -\rho \left(\sin(az) \sin(alt) + \cos^2(az) \cos(alt) \right) \end{pmatrix} \tag{5.45}$$

Si (x', y') sont les coordonnées dans la caméra, (x, y) sont les coordonnées au sol, et (t_x, t_y) la position du télescope au sol, alors :

$$x = \rho \left[x' \sin(az) - y' \cos(az) \sin(alt) \right] + t_x \tag{5.46}$$
$$y = \rho \left[x' \cos(az) - y' \sin(az) \sin(alt) \right] + t_x \tag{5.47}$$
$$\tag{5.48}$$

Par similarité, la projection du point $(\alpha' = x' + \cos(direction), \beta' = y' + \sin(direction))$ au sol donne le point (α, β) :

$$\alpha = \rho \left[\alpha' \sin(az) - \beta' \cos(az) \sin(alt) \right] + t_x \tag{5.49}$$
$$\beta = \rho \left[\alpha' \cos(az) - \beta' \sin(az) \sin(alt) \right] + t_x \tag{5.50}$$
$$\tag{5.51}$$

L'angle *direction* (orientation de la gerbe de particules dans la caméra) est donnée par le vecteur $(t_x - \alpha, t_y - \beta)$ au sol :

$$\delta_x = \rho \left[(x' - \alpha') \sin(az) - (y' - \beta') \cos(az) \sin(alt) \right] \tag{5.52}$$
$$\delta_y = \rho \left[(x' - \alpha') \cos(az) - (y' - \beta') \sin(az) \sin(alt) \right] \tag{5.53}$$

5.6.2.2 Méthode historique

La méthode historique est utilisée dans l'analyse de l'expérience H.E.S.S. [74]. Elle consiste à calculer un barycentre entre les points d'intersection des différentes droites portées par les grands axes des ellipses obtenus par le calcul des paramètres Hillas (voir figure 5.38).

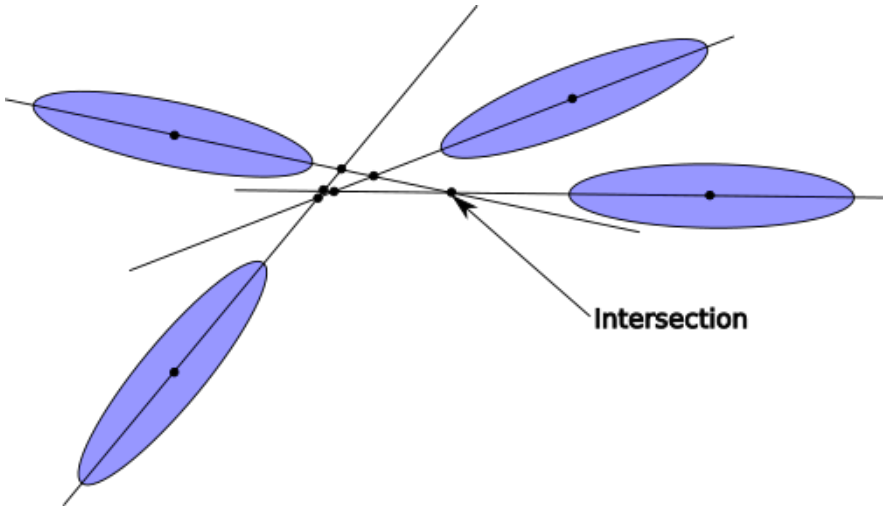


FIGURE 5.38 – Reconstruction stéréoscopique qui utilise une pondération des intersections des directions des ellipses.

Les différents points d'intersections de droites sont pondérés en fonction de l'intensité du signal, du rapport entre la longueur et la largeur des ellipses utilisées. Ce coefficient ρ qui pondère l'intersection (x_i, y_i), est donné par :

$$\begin{aligned} \Delta\beta &= \phi_1 - \phi_2 \\ \rho &= \frac{\|\sin \Delta\beta\|}{\left(\frac{1}{I_1} + \frac{1}{I_2}\right) + \left(\frac{W_1}{L_1} + \frac{W_2}{L_2}\right)} \end{aligned}$$

Où ϕ_1 et ϕ_2 sont les directions des droites, I_1 et I_2 l'intensité du signal, L_1 et L_2 la longueur et W_1, W_2, la largeur des ellipses.

La direction reconstruite, (x, y), est donnée par l'équation :

$$(x, y) = \frac{\sum_{i=0}^{N} \rho_i \, (x_i, y_i)}{\sum_{i=0}^{N} \rho_i}$$

Où N est le nombre d'intersections (x_i, y_i).

L'optimisation de cette méthode est grandement simplifiée par notre format de données (voir chapitre 3). Bien que le nombre d'intersections N augmente grandement selon le nombre de droites à traiter :

$$N = C_2^n \tag{5.54}$$

$$= \frac{n!}{2! \cdot (n-2)!} \tag{5.55}$$

où n est le nombre de droites, le calcul est relativement rapide car il est rare d'avoir plus de 10 télescopes dans un même événement après l'application des coupures précédentes.

Cette méthode nous a surtout servit de référence pour nos futures méthodes de reconstruction stéréoscopique.

Bien que nous ayons eu quelques difficultés lors des premiers tests (voir anecdote 5.6.1), cet algorithme produit des biais lorsque deux directions sont proches (voir figure 5.39). La fluctuation de signal dans les caméras peut décaler le barycentre de la gerbe et modifier sa direction. Les grands axes de deux ellipses peuvent alors devenir quasiment parallèles. L'intersection qui en résulte est alors très éloignée de la position attendue (voir même au-delà du champ de vue des télescopes). Quand un tel cas se produit, le barycentre final est fortement dévié même si la pondération de l'intersection est plus faible puisque les poids sont mutualisés entre les différentes droites. Si deux droites ont un poids fort et que la troisième est plus faible, deux intersections sur trois auront un poids plus faible, ce qui produira un barycentre éloigné de la position voulue.

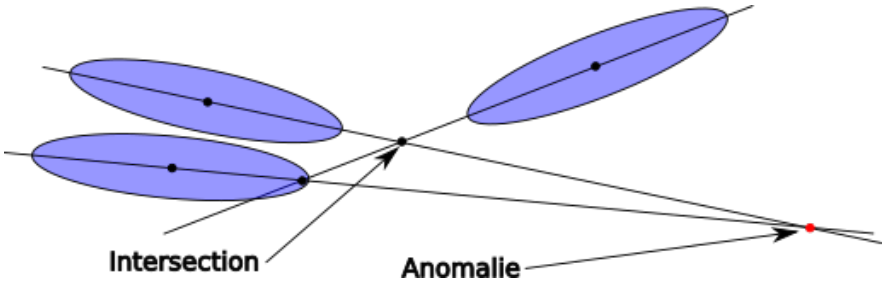


FIGURE 5.39 – Anomalie de reconstruction de la méthode historique.

Anecdote 5.6.1 – **Des télescopes les uns sur les autres**
Le tout premier test de cette méthode ne fut pas concluant. Certains télescopes étaient superposés dans les simulations. Certains points d'intersection se retrouvaient extrêmement loin de l'intersection voulue, puisque deux télescopes superposés ont des images similaires et, par le fait, des directions d'ellipses identiques. L'intersection de deux droites quasiment parallèles se retrouve donc loin de la position attendue.

5.6.2.3 Méthode dite papillon

La méthode dite papillon a été développée pour résoudre le problème posé lorsque deux droites sont quasiment parallèles.

Nous allons expliquer son fonctionnement dans le cadre de la détermination du paramètre d'impact de la gerbe de particules au sol. Celle-ci est plus imagée que la reconstruction angulaire même si cette dernière fonctionne sur le même principe.

La figure 5.40 illustre le fonctionnement de cette méthode. Les fonctions, dites papillons, sont superposées au sol. Chacune d'elles décrit l'ensemble des directions possibles de la gerbe de particule au sol reconstruite à l'aide des paramètres Hillas de l'image correspondante. Dans ce cas, les différents angles d'ouverture de ces fonctions permettent de modéliser les incertitudes que nous avons sur chaque direction reconstruite.

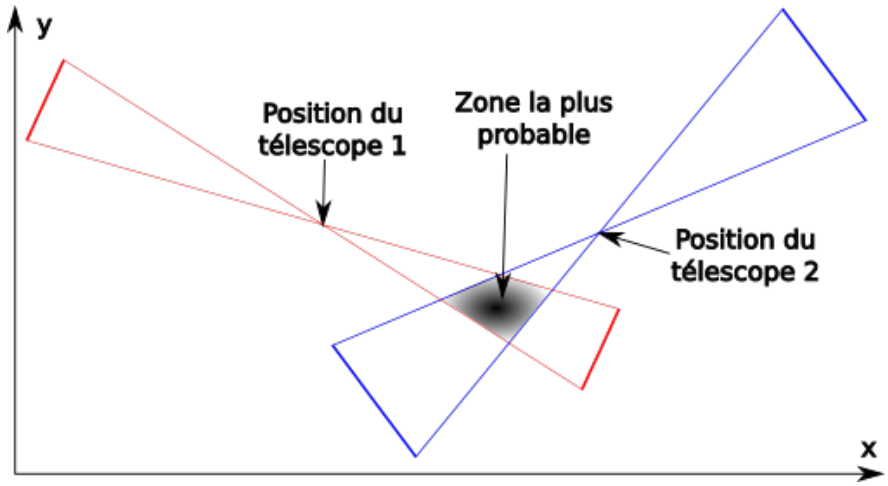


FIGURE 5.40 – Illustration de la reconstruction dite papillon.

La différence majeure avec la méthode historique est que l'erreur sur le champ de vus du télescope est prise en compte autrement. Les papillons s'étalent en gaussiennes à deux dimensions. Ces gaussiennes sont préalablement définies comme :

$$f \;:\; \mathbb{R}^4 \longrightarrow \mathbb{R}$$

$$(x, y, s, \sigma) \longmapsto \begin{cases} \begin{cases} A \exp\left(\dfrac{-y^2}{2 \cdot (s \cdot \sigma \cdot x)^2}\right) & \text{si } \|x\| > 1 \\ A \exp\left(\dfrac{-y^2}{2 \cdot (s \cdot \sigma)^2}\right) & \text{sinon} \end{cases} & \text{si } \|x\| > r \\ 0 \quad \text{sinon} \end{cases} \tag{5.56}$$

Où :

- **r** est la distance maximale où le télescope peut détecter une gerbe de particules. Cette distance a été évaluée à 700 mètres grâce au simulations Monte-Carlo des gerbes de particules (voir figure 5.41) et ajustée après plusieurs essais.
- σ décrit l'ouverture du papillon lorsque $\|x\| \leq 1$. Cela évite à la fonction décrite ci-dessus d'être trop piquée sur le télescope.
- **A** le poids global de la gaussienne est déterminé avec l'intensité de la gerbe dans la caméra de l'image courante.
- **s** le coefficient d'ouverture de la gaussienne calculé avec les paramètres Hillas tel que :

$$s \;=\; \tan\left(\frac{\pi \cdot w}{2 \cdot l}\right) \tag{5.57}$$

Où l est la longueur de l'ellipse dans la caméra et w sa largeur sont calculés (voir annexe A) de sorte que l'angle d'ouverture de la gaussienne soit petit lorsque l'axe de l'ellipse est bien défini et égal à $\frac{\pi}{2}$ quand c'est un cercle (dans ce cas le champs de vue est circulaire).

L'implémentation de cette méthode utilise une matrice qui décrit le sol. Chaque colonne décrit une position en x (direction nord) et les lignes décrivent les différentes positions en y (direction ouest). Les matrices que nous avons testées représentent $4\,\mathrm{km}^2$ de terrain (légèrement plus grand que le site nord).

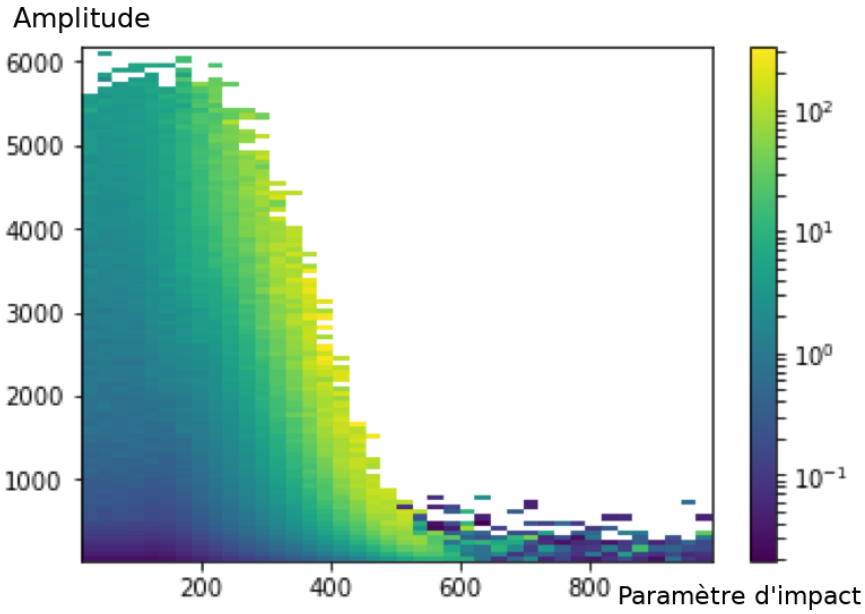


FIGURE 5.41 – Distribution de l'amplitude dans les images des caméras LST-CAM en fonction du paramètre d'impact correspondant.

Un élément, m_{ij}, de la matrice $M \in M_{nm}(\mathbb{R})$ est calculé comme :

$$x' = j \cdot d_x + o_x \tag{5.58}$$
$$y' = i \cdot d_y + o_y \tag{5.59}$$
$$x = x' \cdot \cos\theta - y' \cdot \sin\theta \tag{5.60}$$
$$y = x' \cdot \sin\theta + y' \cdot \cos\theta \tag{5.61}$$
$$s_k = \tan\left(\frac{\pi \cdot w_k}{2 \cdot l_k}\right) \tag{5.62}$$
$$m_{ij} = \sum_{k=1}^{N_t} \sum_{p=1}^{n} \sum_{q=1}^{m} f\left(x, y, s_k, \sigma_k\right) \tag{5.63}$$

— $(\mathbf{o_x}, \mathbf{o_y})$: origine située au sud-ouest du site.
— $(\mathbf{d_x}, \mathbf{d_y})$: taille d'un élément (ici $d_x = d_y = 2\,\mathrm{m}$).
— θ : direction présumée de la gerbe de particules.
— $\mathbf{N_t}$: nombre de télescopes dans l'événement à reconstruire.
— $\mathbf{f}$: décrit par l'équation 5.56.
— $\sigma_\mathbf{k}$: ouverture minimale de la gaussienne.

Nous utilisons une somme sur les contributions des télescopes et non pas un produit pour évité que cette méthode revienne à une intersection de droite ce qui nous placerait exactement dans le cas décrit par la section suivante.

Bien que la complexité de cette méthode soit linéaire, elle reste plus lente que la précédente car de nombreux appels mémoire sont nécessaires. Néanmoins, elle peut être accélérée en diminuant le nombre de mailles utilisé.

La figure 5.42 montre deux cas de reconstruction du paramètre d'impact avec des pas de deux mètres. Le paramètre d'impact est positionné sur le score le plus élevé. Nous avions tout d'abord utilisé un calcul de maximum par fenêtre glissante, mais les résultats ne sont meilleurs que lorsque les cellules sont extrêmement petites (moins d'un mètre). Nous avons donc augmenté la taille des mailles et cherché un maximum classiquement. On constate que l'on obtient de bien meilleurs résultats que la méthode historique.

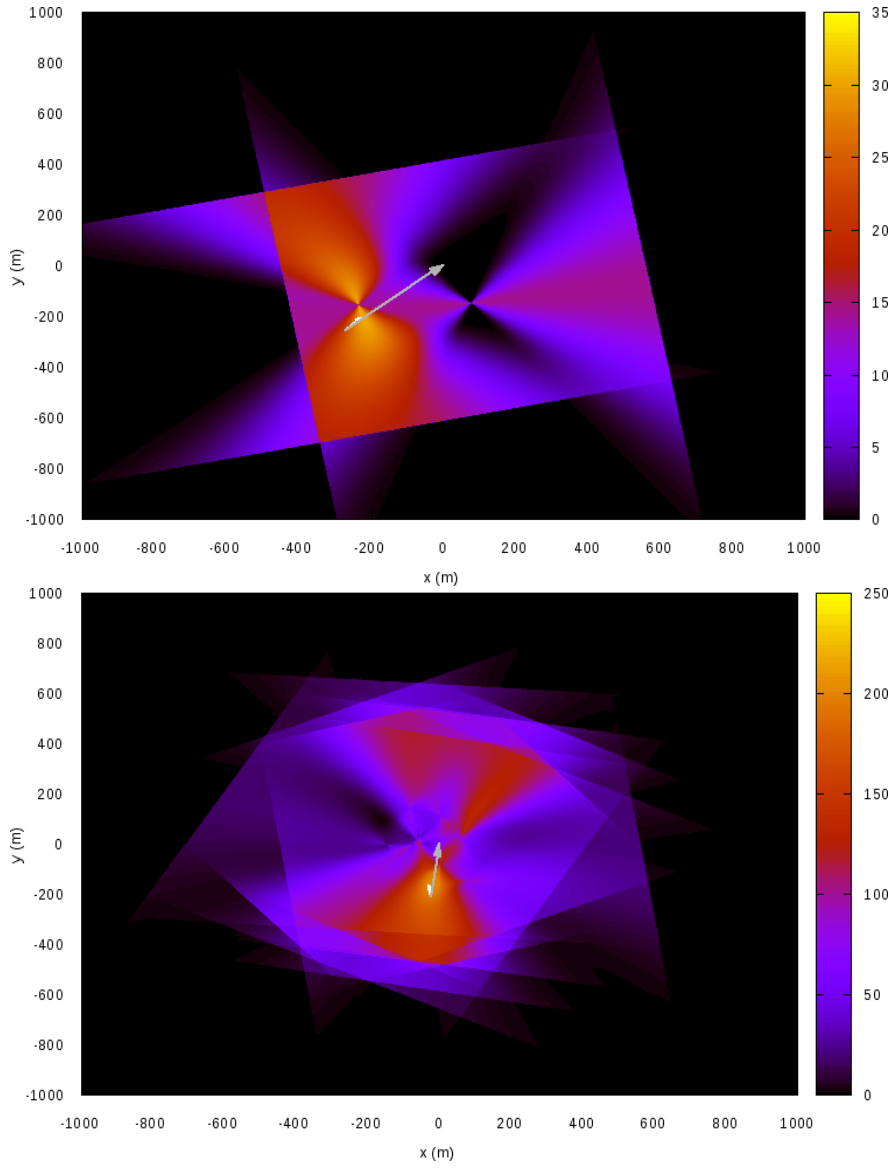


FIGURE 5.42 – Reconstruction du paramètre d'impact avec deux télescopes (en haut) et avec huit télescopes (en bas). La flèche grise donne le paramètre d'impact reconstruit par la méthode historique. La flèche blanche indique le paramètre d'impact reconstruit par la méthode papillon. Les flèches partent du paramètre d'impact simulé.

Notre stagiaire à ensuite implémenté une méthode utilisant des boîtes qui réduit le nombre de calcul par télescope.

La figure 5.43 montre un résultat de ce travail. L'utilisation de boîtes englobantes réduit d'une manière significative le nombre de calcul à effectuer (environ quatre fois moins dans ce cas).

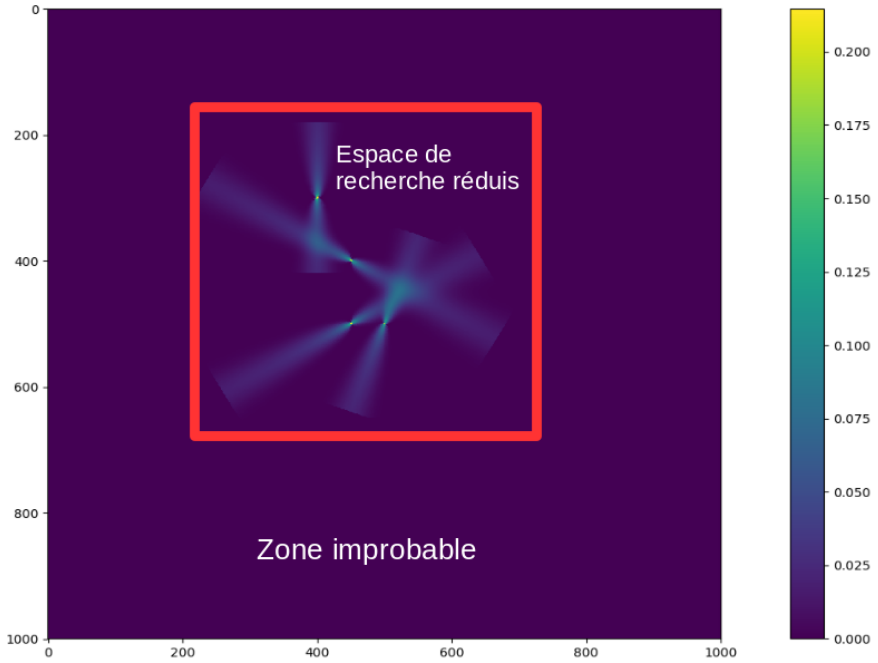


FIGURE 5.43 – Reconstruction du paramètre d'impact avec quatre télescopes. L'espace de calcul est réduit grâce à des boîtes englobantes.

Bien que cette méthode soit plus lente que la méthode historique, elle obtient de bien meilleurs résultats.

Anecdote 5.6.2 – **Rotation des caméras**

Nous avons remarqué lors de la première utilisation de cette méthode que certaines caméras pointaient dans des directions aberrantes. Nous avions tout d'abord soupçonné le calcul des paramètres Hillas, mais il s'est avéré que l'erreur provenait d'une rotation délibérée des caméras dans les simulations. Cette nouvelle a suscité un vif débat entre les responsables de la simulations de CTA et les autres physiciens puisque aucune reconstruction ne parvenait à obtenir les objectifs de sensibilité et de résolution angulaire annoncés par CTA. De plus, les angles de rotations annoncés dans la simulation n'étaient pas non plus pertinents. L'équipe en charge de la simulation a finalement annoncé qu'une documentation aurait due être écrite deux ans auparavant.

5.6.2.4 Méthode d'intersection multiple de droites

La méthode d'intersection multiple de droites (multi-line intersection) est inspirée de l'article [75], qui décrit le cas à N dimensions d'une manière complexe.

L'idée est de déterminer le point qui minimise la distance entre toutes les droites à la fois. Ces droites peuvent également avoir une pondération (comme dans les algorithmes précédents).

Pour toute droite associée à un point $M = (a_x, a_y)$ et un vecteur porteur $\mathbf{n}$. La distance entre un point, P et une droite est donnée par :

$$D_i = (\mathbf{PM}.\mathbf{u}) = (a_x - p_x)u_x + (a_y - p_y)u_y \tag{5.64}$$

Avec $\mathbf{u} = -\mathbf{n} \wedge \mathbf{k}$, $\mathbf{k} = (0, 0, 1)$. Où $\wedge$ représente le produit vectoriel. Le point d'intersection est celui qui minimise la somme des distances D_i :

$$P = \underset{(p_x,p_y)}{argmin}(D(P)) \tag{5.65}$$

$$D(P) = \sum_i D_i^2(P) \tag{5.66}$$

Pour résoudre ce problème, il faut trouver un minimum. Pour cela il faut que la dérivée de l'expression 5.66 s'annule :

$$\partial D_x = \sum_i \frac{\partial D_i}{\partial p_x} = p_x A + p_y Q - C_1 \tag{5.67}$$

$$\partial D_y = \sum_i \frac{\partial D_i}{\partial p_y} = p_x Q + p_y B - C_2 \tag{5.68}$$

Avec :

$$A = \sum u_x^2 \tag{5.69}$$

$$B = \sum u_y^2 \tag{5.70}$$

$$Q = -\sum u_x u_y \tag{5.71}$$

$$C_1 = \sum u_y \left(a_y u_x - a_x u_y\right) \tag{5.72}$$

$$C_2 = \sum u_x \left(a_x u_y - a_y u_x\right) \tag{5.73}$$

Donc le point recherché satisfait le système d'équation suivante :

$$\begin{pmatrix} A & Q \\ Q & B \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \tag{5.74}$$

Ceci n'est valable uniquement si $AB - Q^2 \neq 0$

La solution est donc donnée par :

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \frac{1}{AB - Q^2} \begin{pmatrix} B & -Q \\ -Q & A \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \tag{5.75}$$

La figure 5.44 compare les erreurs sur la reconstruction du paramètre d'impact entre la méthode papillon et la méthode d'intersection multiple de droite en fonction du nombre de télescopes dans l'événement. Les erreurs sont comparables mais la méthode d'intersection multiple de droite a une dispersion d'erreurs plus faible ce qui en fait un meilleur candidat, d'autant que son temps d'exécution est bien plus court.

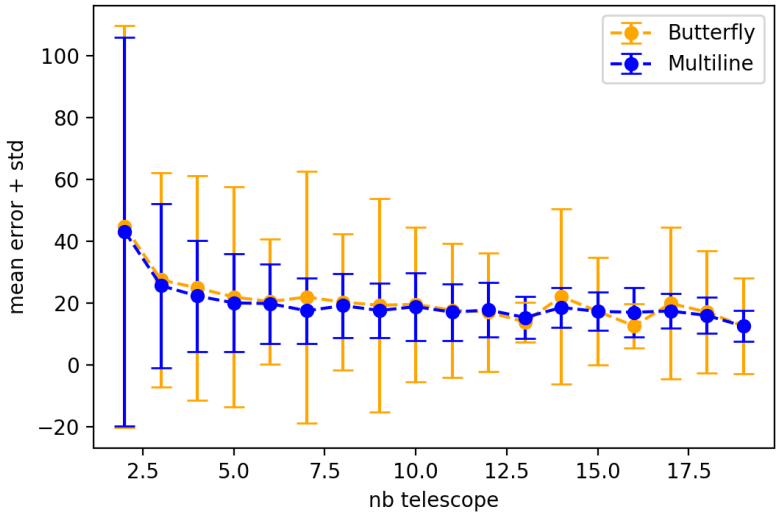


FIGURE 5.44 – Erreur de la reconstruction du paramètre d'impact avec la méthode papillon (Butterfly) en orange et la méthode multi-line intersection en bleue.

La figure 5.45 montre la résolution angulaire de notre analyse de données HPC. Les résultats sont en accord avec les objectifs de CTA.

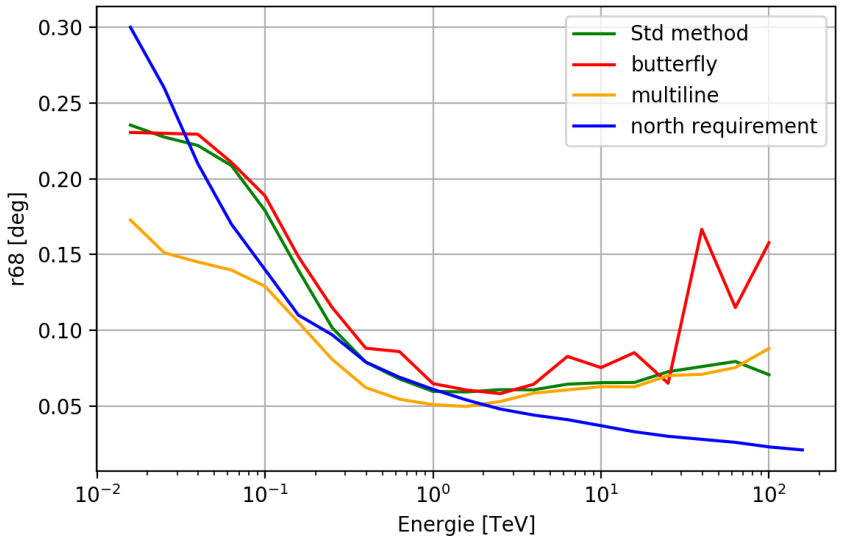


FIGURE 5.45 – Résolution angulaire de notre version HPC de l'analyse de données de CTA.

Bien que les trois méthodes de reconstruction donnent des résultats similaires, la méthode d'intersection multiple de droites améliore grandement la résolution angulaire à basse énergie ($E < 200\,\mathrm{GeV}$) ce qui sera un atout majeur pour CTA.

5.6.3 Surface effective

La surface effective de la reconstruction désigne la surface couverte des télescopes au sol sensible aux rayons gamma. Elle est proportionnelle au nombre d'événements reconstruits par rapport au nombre d'événements simulés en fonction de leurs énergies. Plus la sensibilité est grande et plus la surface effective est importante.

La figure 5.46 montre la surface effective de notre analyse HPC. On constate que notre analyse est globalement en accord avec les prérequis de CTA, sauf à basse énergie : $E < 80\,\text{GeV}$ pour une multiplicité de 6 télescopes, $E < 50\,\text{GeV}$ pour une multiplicité de 4, $E < 40\,\text{GeV}$ pour une multiplicité de 2 télescopes.

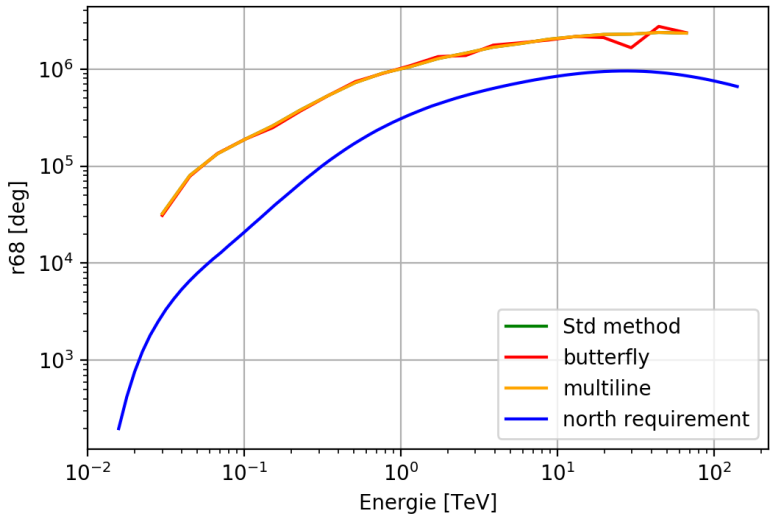


FIGURE 5.46 – Surface effective de notre version HPC de l'analyse de données de CTA comparée aux performances de la reconstruction CTA.

Nous remarquons que les surfaces effectives des différentes méthodes de reconstruction sont quasiment identiques.

5.6.4 Reconstruction en énergie

Comme son nom l'indique, la reconstruction en énergie détermine, à partir des paramètres Hillas, l'énergie de la particule initiale qui a créé la gerbe.

Deux méthodes sont couramment utilisées :

— L'utilisation d'une table qui renvoie l'énergie de la particule en fonction de différents paramètres Hillas comme l'amplitude de l'image, le nombre de pixels sélectionnés, la longueur de l'ellipse, etc.
— Un arbre de décision peut également déterminer l'énergie si il est entraîné avec les paramètres Hillas de tous les télescopes de l'événement.

5.6.4.1 Utilisation de tables

La méthode la plus simple qui détermine l'énergie de la particule incidente consiste à créer un tableau. Celui-ci prend en entrée les paramètres Hillas et renvoie l'énergie moyenne associée à ces paramètres. Sa complexité varie en fonction du nombre de paramètres utilisés.

Nous avons choisi une approche minimaliste, qui nous permet de tester préalablement la reconstruction avec une méthode simple, donc facile à vérifier. Celle-ci n'utilise que trois paramètres (les deux premiers sont les plus importants) :

— La distance entre le télescope et le point d'impact. Celle-ci varie entre 0 et 1 000 mètres et est subdivisée en 100 intervalles.
— L'amplitude du signal à l'intérieur d'une caméra qui varie entre 0 et 6 400 photons et est également divisée en 100 intervalles.
— La distance entre le pixel qui contient le maximum de signal et le centre de la caméra. Elle varie entre 0 et le rayon de la caméra (1.2 m pour les LST-CAM) et est divisée en 12 intervalles.

Nous avons effectué nos tests avec des simulations de sources de rayons gamma ponctuelles et également avec des rayons gamma diffus.

Les simulations ponctuelles décrivent des sources de rayons gamma ponctuelles. Elle sont utilisées pour évaluer l'acceptance de l'expérience, c'est à dire la surface effective de la reconstruction associée à chaque position dans le champ de vue des télescopes. Les simulations de rayons gamma diffus sont quant à elles utilisées afin de tester la reconstruction sans les biais inhérents aux sources ponctuelles comme leurs positions ou leurs tailles supposées nulles.

La figure 5.47 montre les résultats obtenus avec des sources ponctuelles et des gamma diffus.

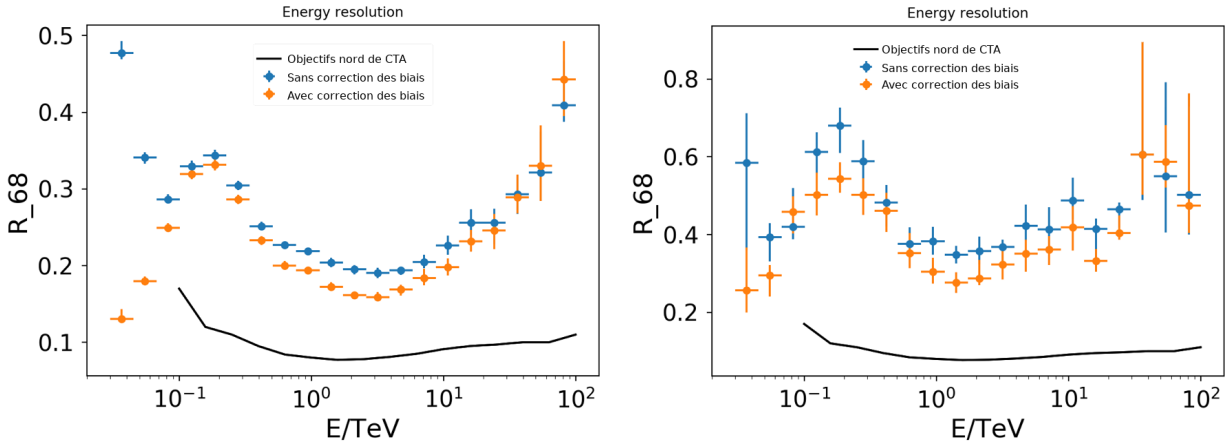


FIGURE 5.47 – Résolution en énergie de notre reconstruction avec des tables. **À gauche**, le résultat avec des simulations de sources ponctuelles. **À droite**, le résultat avec les simulations de rayons gamma diffus.

La méthode que nous avons utilisée est très simple, ce qui explique les différences de résultats comparés aux objectifs de CTA. Cependant, les objectifs de CTA sont très probablement biaisés du fait que la source se situe au centre du champ de vue des caméras (voir section 5.6.4.3). En second lieu, les objectifs de CTA sont déterminés avec des sources ponctuelles, les objectifs utilisant des rayons gamma diffus ne sont pas déterminés et la courbe des objectifs de CTA est donc montrée à titre de comparaison.

L'optimisation physique d'une telle reconstruction implique de prendre en considération les variations de quasiment tous les paramètres disponibles pour chaque télescope. Cette tâche complexe est généralement effectuée par des algorithmes spécialisés comme les arbres de décision.

5.6.4.2 Utilisation d'arbres de décision

L'utilisation d'arbres de décision est de plus en plus courante en physique des particules. En effet, les expériences sont de plus en plus complexes et l'espace des paramètres à traiter augmente en conséquence.

Le projet CTA n'échappe pas à la règle. La multiplication des télescopes augmente le nombre de paramètres à utiliser afin de minimiser l'erreur sur la reconstruction en énergie dans ce cas.

Cette reconstruction en énergie utilise un arbre de décision par type de caméra (figure 5.48). Nous en avons donc deux pour décrire le site nord (La Palma).

Ces arbres sont entraînés en même temps avec les paramètres géométriques reconstruits des différents télescopes. L'apprentissage avec des paramètres reconstruits permet aux arbres de corriger les biais dûs à la reconstruction géométrique.

Une fois la phase d'apprentissage terminée, les paramètres des événements sont utilisés dans les arbres en fonction du type de télescope utilisé et les résultats de chaque arbres sont finalement moyennés afin d'obtenir l'énergie reconstruite finale de l'événement.

Nous avons utilisé les arbres de décision fournis par le module Python ***scikit-learn*** [76].

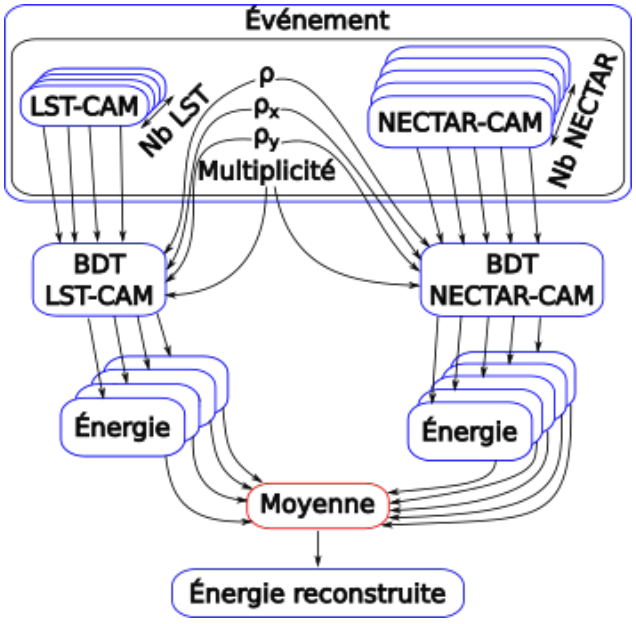


FIGURE 5.48 – Utilisation des arbres de décision (BDT) dans notre reconstruction en énergie.

Notre reconstruction en énergie a été entraînée avec 2 192 fichiers de simulations de sources de rayons gamma ponctuelles et testée avec 272 fichiers de simulation également ponctuelles. La figure 5.49 montre les résultats obtenus avec des arbres de décision.

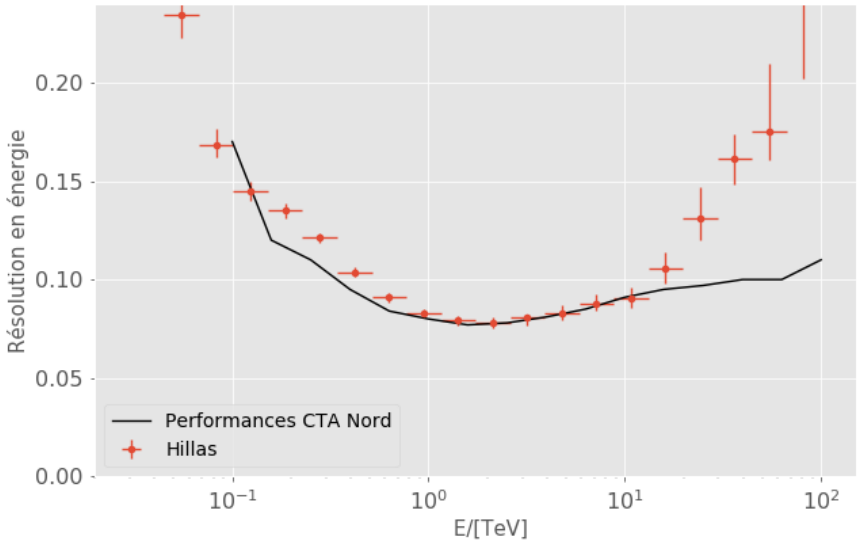


FIGURE 5.49 – Résolution en énergie de notre reconstruction qui utilise des arbres de décision sur des sources de rayons gamma ponctuelles.

Les performances de CTA nord sont plus précises que l'objectif de CTA. Elles sont déterminées par la reconstruction de l'expérience MAGIC, MARS [77] basée sur la reconstruction Hillas. Celles de notre analyse sont en accord jusqu'à 10 TeV. L'écart au delà de 10 TeV est présent car nous avons optimisé la reconstruction physique à plus basse énergie. De ce fait, nous conservons trop d'événements à plus haute énergie. Cette différence pourrait être simplement réglée par l'ajout d'une coupure sur les paramètres des événements de haute énergie. Cependant, la statistique disponible est insuffisante pour conclure au delà de 10 TeV, nous n'avons donc pas ajouté une telle coupure.

La figure 5.50 montre l'utilisation des paramètres donnés aux arbres de décision pour les grands télescopes, qui utilisent des LST-CAM, et les moyens télescopes équipés de NECTAR-CAM.

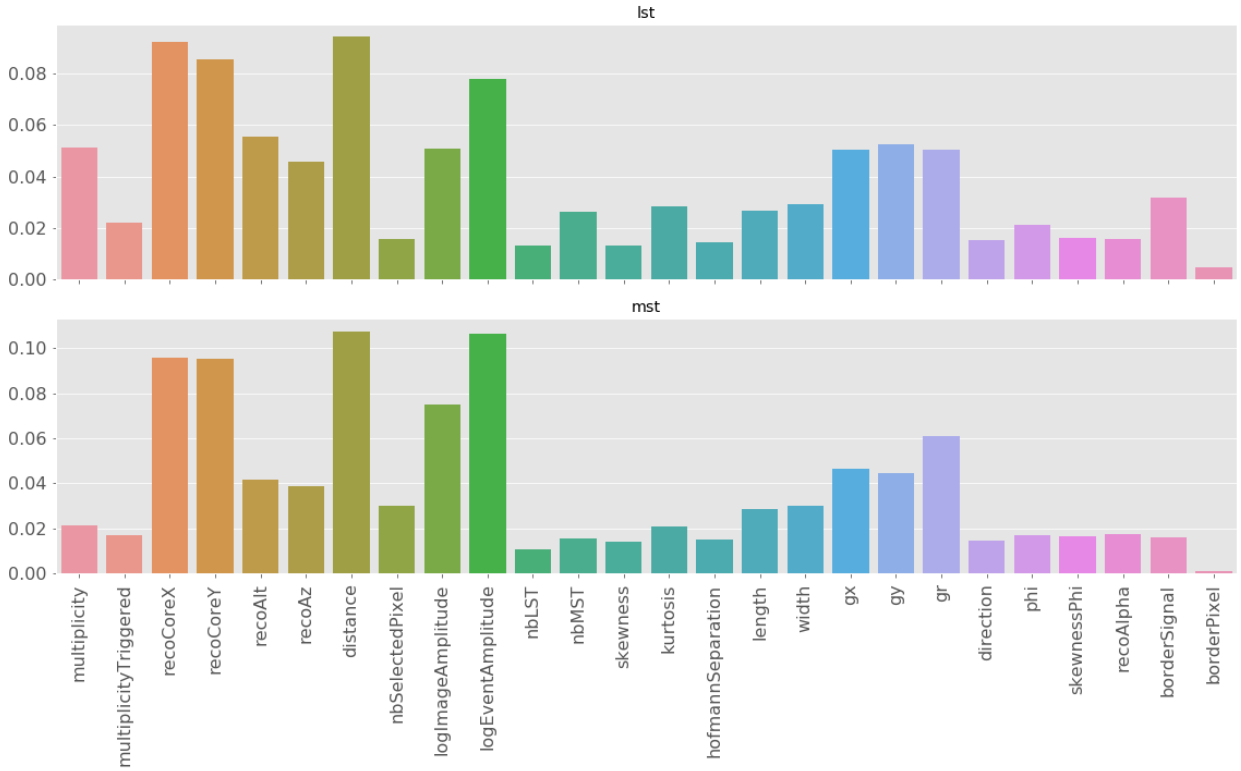


FIGURE 5.50 – Pertinence des différents paramètres passés aux arbres de décision utilisés pour la reconstruction en énergie.

Dans les deux cas la distance entre le point d'impact reconstruit et les différents télescopes est très importante, au même titre que la position absolue de ce point d'impact, ou le logarithme de la somme de toutes les amplitudes des images de l'événement (*logEventAmplitude*).

La multiplicité est d'avantage utile pour les LST car ils sont moins sensibles aux hautes énergies. Ce paramètre permet donc de rehausser l'énergie perçue par ces télescopes.

Les paramètres qui décrivent le barycentre de la gerbes dans les caméra sont également très utilisés (g_x, g_y et $g_r = \sqrt{g_x^2 + g_y^2}$).

Les autres paramètres Hillas apportent des informations plus anecdotiques sur l'énergie.

Nous avions ajouté un paramètre *borderPixel* qui décrit la quantité de signal au bord des caméras. Cela permet d'estimer à quel point la gerbe est coupée dans la caméra. Il semble ne pas être important dans le cas d'une source ponctuelle placée au centre du champs de vue des caméras mais pourrait s'avérer utile si cette dernière est décentrée.

Optimisation cachée 5.6.1 – **Détermination des coupures**
Les résultats de l'analyse de données dépendent fortement des coupures appliquées dans celle-ci. Une analyse extrêmement rapide permet de tester un plus grand nombre de combinaisons de coupures et ainsi d'obtenir plus facilement de meilleurs résultats.

5.6.4.3 Biais des résultats

L'utilisation des arbres de décision est devenue courante dans le domaine de la physique des particules. Néanmoins, l'utilisation et l'optimisation des analyses de physiques sur des sources ponctuelles met en lumière des résultats pour le moins étonnants.

La collaboration de CTA travaille avec des sources ponctuelles simulées au centre du champ de vue des caméras. Cette configuration produit des biais sur des paramètres Hillas utilisés, comme le nombre de pixels sélectionnés ou la position du barycentre, dont la norme est corrélée avec l'énergie de la particule initiale (voir figure 5.51).

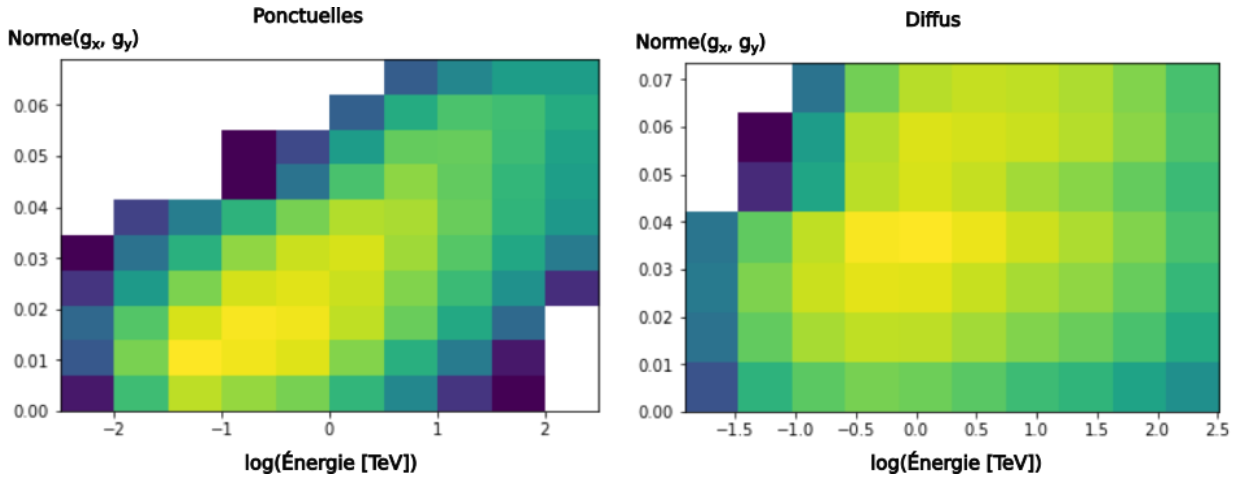


FIGURE 5.51 – **À gauche**, la corrélation entre la norme du barycentre (g_x, g_y) et l'énergie de la particule incidente pour des simulations de sources ponctuelles. **À droite**, la non corrélation de la norme de (g_x, g_y) en fonction de l'énergie de la particule incidente avec les simulations de rayons gamma diffus.

Cette corrélation implique que le seul calcul du barycentre de la gerbe de particule dans la caméra renseigne sur l'énergie de la particule incidente. En effet, il peut arriver qu'une gerbe soit dans l'axe de vue de la caméra, ce qui produit un barycentre qui est également au centre de la caméra. Dans ce cas, la norme de ce barycentre n'est pas corrélée à l'énergie. Cette relation est néanmoins statistiquement juste du fait de l'acceptance [19] dans la caméra :

19. L'acceptance est la sensibilité du télescope. Elle donne directement le rapport de déclenchement entre les gerbes de basses et de hautes énergies.

— Les gerbes de particules de toutes énergies atteignent le centre de la caméra. Mais les basses énergies sont détectées vers le centre de la caméra, là où l'acceptance est maximale. Il y a donc statistiquement plus de basses énergies du fait du spectre de la source (ou par rapport au spectre des rayons cosmiques que nous avons présenté au chapitre 1).
— Ce de fait, plus le barycentre est éloigné de l'axe de la source (dans notre cas, le centre du champ de vue des caméras) et plus les gerbes sont statistiquement de plus haute énergie.

Lors d'un vrai cas d'utilisation, la source n'est pas placée au centre du champs de vue, pour des raisons que nous aborderons dans la section 5.7. Les gerbes de particules peuvent donc être coupées, ce qui implique que leurs barycentres ne sont pas déterminés à l'endroit où ils devraient être. C'est pour cela que la corrélation est très peu évidente dans le cas de rayons gamma diffus puisque leurs barycentres sont répartis aléatoirement dans les caméras. D'où le fait que la norme $g_r = \sqrt{g_x^2 + g_y^2}$ soit un mauvais paramètre.

5.6.5 Discrimination

La discrimination évalue la probabilité d'un événement d'avoir été créé par un rayon gamma plutôt que par un rayon cosmique comme un proton. Cette évaluation était, à l'origine, réalisée empiriquement. Pour cela deux lots de simulations sont nécessaires :

— Des simulations de rayons gamma, les vrais positifs.
— Des simulations de protons (grande majorité du bruit), les faux positifs.

Les deux lots de simulations sont analysées avec des algorithmes et des coupures identiques. Ensuite, les distributions de tous les paramètres Hillas sont comparées entres celles des protons (bruit) et celles des rayons gamma (signal).

Le jeu consiste à exploiter au mieux toutes les différences apparentes entre ces deux lots afin de les séparer. Cela peut s'avérer fastidieux lorsque le nombre de paramètres à traiter augmente.

De la même façon que pour la reconstruction en énergie dans la section précédente, nous avons choisi une approche par arbres de décision. Deux arbres sont utilisés, un pour les LST-CAM (grand télescopes), l'autre pour les NECTAR-CAM (moyens télescopes). Les paramètres Hillas des images sont passés à l'un ou à l'autre en fonction du type de caméra. Finalement, un estimateur renvoie la probabilité que l'événement ai été produit par un rayon gamma (1) ou non (0).

L'utilisateur doit alors définir une ultime coupure qui détermine au dessus de quelle probabilité renvoyée il considère que l'événement est effectivement produit par un rayon gamma. Cette coupure va déterminer la proportion de vrais et de faux positifs dans les données finales. D'ordinaire, les listes de rayons gamma produites contiennent cette probabilité ou une qualité. Il revient alors à l'astrophysicien de choisir quelle qualité il désir utiliser. Plus la qualité est élevée et moins il aura d'événements.

Nous avons entraîner nos arbres de décision avec 2 192 fichiers de simulations de sources de rayons gamma ponctuelles et 6 810 fichiers de protons [20]. Nous les avons testés avec 790 fichier de rayons gamma et 272 fichiers de protons.

Le module Python ***scikit-learn*** fournit un moyen d'évaluer statistiquement le nombre de vrais positifs en fonction du nombre de faux positifs et ainsi aide à déterminer les coupures pour différentes qualités.

Les courbes produites, appelées ROC (Receiver Operating Characteristic), définissent ces ratios. La figure 5.52 montre les différentes courbes ROC que nous obtenons avec un nombre variable de paramètres d'entrée.

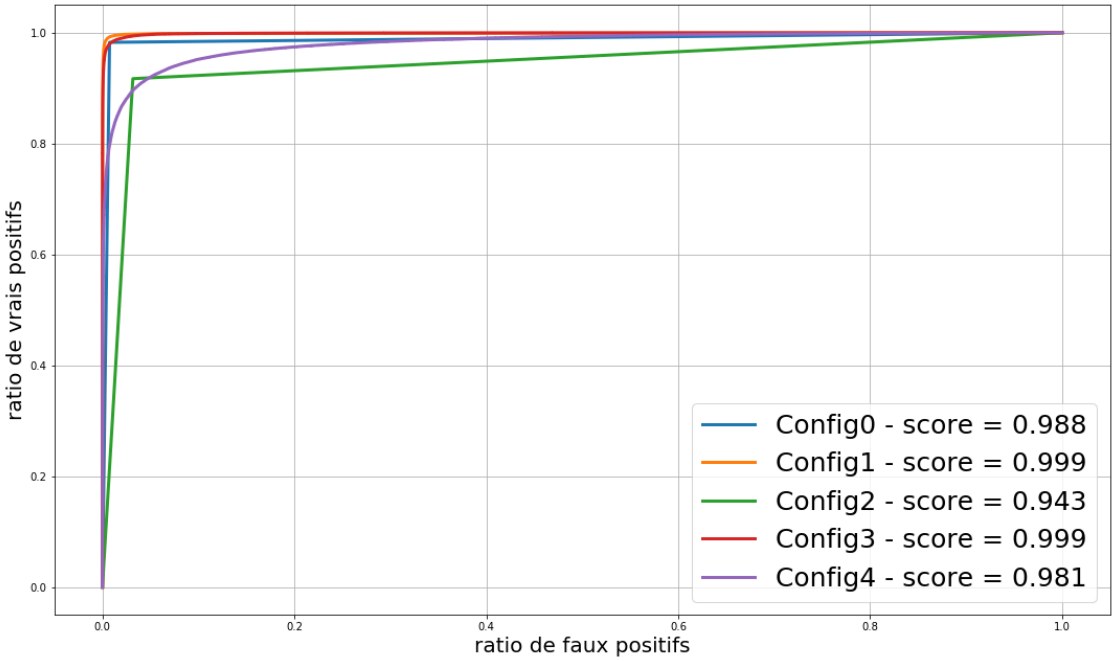


FIGURE 5.52 – Coubres ROC pour différentes configurations. Le score correspond à l'aire sous la courbe.

Les différentes configuration sont les suivantes :

— **Config0** : Arbres de décision, paramètres Hillas, paramètre d'impact, multiplicité, altitude et azimut reconstruits.
— **Config1** : Forêt aléatoire, paramètres Hillas, paramètre d'impact, multiplicité, altitude et azimut reconstruits.
— **Config2** : Arbres de décision, paramètres Hillas, paramètre d'impact, multiplicité.
— **Config3** : classificateur par gradient, paramètres Hillas, paramètre d'impact, multiplicité, altitude et azimut reconstruits.
— **Config4** : classificateur par gradient, paramètres Hillas, paramètre d'impact, multiplicité.

20. Les simulations de protons décrivent nécessairement des protons diffus car ils sont déviés par les champs magnétiques dans cette gamme d'énergie. Leur distribution est donc uniforme (voir chapitre 1).

Le tableau 5.6 résume les différentes configurations.

Configuration	Méthode	Paramètre Hillas + paramètre d'impact + multiplicité	altitude et azimut reconstruits
Config0	Arbres de décision	oui	oui
Config1	Forêt aléatoire	oui	oui
Config2	Arbres de décision	oui	non
Config3	classificateur par gradient	oui	oui
Config4	classificateur par gradient	oui	non

TABLE 5.6 – Résumé des différentes configuration utilisées pour la discrimination Hillas.

Le score est l'aire sous la courbe, c'est un paramètre simplifié qui permet de hiérarchiser simplement les différentes configurations. Nous remarquons que l'utilisation des angles reconstruits de l'altitude et de l'azimut influence fortement la qualité de la discrimination. Ces résultats suspects sont abordés dans l'anecdote 5.6.3.

Le travail d'optimisation de l'analyse dite Hillas s'arrête ici. Nous allons néanmoins expliquer succinctement l'utilisation des listes de photons que nous fournissons aux astrophysiciens dans la prochaine section.

Anecdote 5.6.3 – **Biais de la discrimination**

Les taux de rejet annoncés lors des conférences étaient particulièrement bons (supérieurs à 80%). Nous avions beaucoup de difficultés à atteindre de tels résultats, jusqu'au jour où nous avons appris comment les calculs étaient effectués. Les taux de rejet annoncés étaient calculés entre des sources ponctuelles de rayons gamma et des protons diffus (dispersés sur tout le ciel). Une telle discrimination se résume au calcul du rayon de la source ponctuelle, (voir figure 5.53). Les événements reconstruits sous le rayon sont considérés comme des rayons gamma et les autres sont rejetés comme étant des protons. Le jeu consiste à augmenter le rayon pour que la surface effective soit suffisamment grande sans trop l'augmenter pour que le taux de rejet soit également important. Comme nous l'avons mentionné, ce travail est effectué par un arbre de décision. Celui-ci n'a aucune difficulté à faire ce que nous venons de décrire puisqu'il utilise le barycentre calculé de la gerbe de particules dans la caméra sachant que la source se situe en son centre. Il suffit donc de couper sur la norme du barycentre. Bien entendu, cette méthode est extrêmement biaisée. L'excuse avancée dans ce cas était que les simulations de rayons gamma diffus n'étaient pas assez nombreuses.

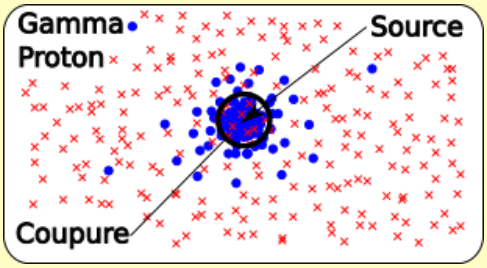


FIGURE 5.53 – Reconstruction d'une source ponctuelle de rayons gamma entourée de protons diffus.

5.7 Création de cartes du ciel et analyse de physique

Une fois la reconstruction stéréoscopique terminée, les listes d'événements produites ne contiennent pas uniquement des rayons gamma. Un grand nombre a été supprimé mais certains subsistent malgré l'étape de discrimination.

Pour cela, une carte du ciel est construite à partir de tous les événements reconstruits. Une étape supplémentaire de discrimination peut alors être effectuée en utilisant le fait que l'astronomie gamma étudie des sources qui ont par définitions des taille finies dans le ciel.

En effet, nous avons vu au début de ce manuscrit que les rayons cosmiques chargés sont considérés comme du bruits en astronomie gamma car leur distribution dans ce ciel ne donne pas de renseignement sur la position de leurs sources. De ce fait, leur distribution est diffuse dans le ciel contrairement aux rayons gamma qui ne sont pas déviés par les champs magnétiques galactiques et dont la distribution décrit celle des sources.

Une méthode d'élimination de ce bruit résiduel, appelée ON-OFF, est utilisée. Deux régions sont alors définies :

— La région dite ON, qui contient la source que l'on désire étudier.
— La région dite OFF, qui ne contient aucune source et permet d'évaluer le bruit résiduel.

Trois méthodes existent pour définir la région OFF :

— **On-Off** : la région ON est observée, ensuite, la région OFF. Cette dernière permet d'évaluer le ratio de bruit dans la région ON (voir figure 5.54).
— **Ring-background** : La région ON n'est qu'une partie du champ de vue. C'est un disque décentré du champ de vue. La région OFF est un anneau. Son épaisseur est le diamètre de la région ON et son rayon moyen est donné par la position de la région ON [21] (voir figure 5.55). Cette technique permet un temps d'observation supérieur à la méthode **On-Off** car le signal et le bruit sont évalués lors de la même observation.
— **Multiple-Off** : est très similaire à la méthode **Ring-background** mais la région OFF est composée de plusieurs disques distribués autour de l'anneau du **Ring-background** (voir figure 5.56).

Ensuite les événements des régions ON et OFF sont comptés. Après une normalisation selon l'aire des régions, le nombre d'événements bruités est soustrait à la région ON.

21. Le décalage est très important car il défini le rayon de la région OFF. Dans ce cas, la région OFF a une symétrie de rotation qui garanti que les événements sont reconstruits avec la même acceptance (qui se dégrade selon la position radiale dans la caméra).

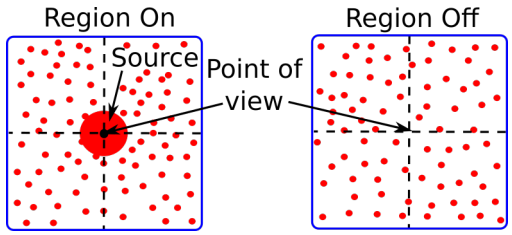


FIGURE 5.54 – Méthode **On-Off** classique utilisée pour réduire le bruit causé par les rayons cosmiques chargés. Les points rouges représentent les événements reconstruits.

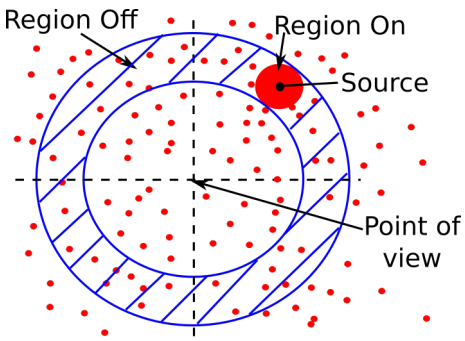


FIGURE 5.55 – Méthode **Ring-background**. Les points rouges représentent les événements reconstruits.

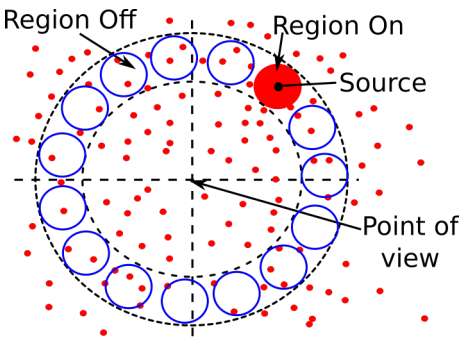


FIGURE 5.56 – Méthode **Multiple-Off**. Les points rouges représentent les événements reconstruits.

Les événements conservés seront utilisés pour créer des spectres, des courbes de lumière ou des cartes du ciel.

— Un spectre donne des informations sur l'activité de la source en terme de flux de rayons gamma.
— Une courbe de lumière montre l'activité de la source dans le temps (jour, semaine, mois).
— Une carte du ciel décrit le ciel visible en rayons gamma.

Tous ces outils permettent d'améliorer notre connaissance des sources de rayons gamma.

L'analyse physique est rapide comparée à la reconstruction, seulement 15 minutes pour finaliser une observation de 28 minutes dans l'expérience H.E.S.S.

5.8 Conclusion

Nous avons présenté dans ce chapitre la méthode classique de reconstruction des événements ainsi que son optimisation.

Nous avons obtenu d'importants facteurs d'accélération, supérieurs à 700 comparés à la version de l'analyse de l'expérience H.E.S.S., et proches de 600 pour l'analyse officielle de CTA en Python.

Ce travail a influencé la conception des centres dédiés à l'analyse des données sur site en diminuant drastiquement le nombre de CPU nécessaires pour les calculs ; typiquement d'une centaine de CPUs par télescope à deux dixaines pour la totalité du site sud.

Nous obtenons des résultats de physiques en adéquation avec les objectifs de CTA. Nous avons proposé de nouveaux algorithmes de reconstruction stéréoscopique tels que la méthode papillon ou la méthode d'intersection multiple de droites.

Tout ce travail a montré à la collaboration que le HPC est incontournable pour répondre aux objectifs de traitement des données de CTA.

Dans ce chapitre, nous avons présenté et optimisé la méthode de reconstruction la plus simple, Hillas. Celle-ci est robuste car elle est indépendante des conditions d'observation ou des caractéristiques physiques des caméras.

Cependant, sa discrimination est limitée car il sera toujours possible d'ajuster une ellipse sur une gerbe de particules produite par un proton ce qui n'est pas souhaitable.

Une méthode plus complexe existe dans le but d'améliorer le rejet des événements de bruit tels que ceux produits pas des protons. Elle se base sur la comparaison des images des événements à analyser aux simulations Monte-Carlo de gerbes de particules et obtient ainsi

une meilleure discrimination. C'est cette méthode que nous allons décrire dans le prochain chapitre.

L'analyse extrêmement rapide que nous avons développée sera testée et installée par nos soins sur le premier prototype de LST qui sera inauguré en octobre 2018 à La Palma (Espagne). Elle servira d'analyse temps réelle monoscopique et sera branchée directement sur le flux de sortie en ZFits du prototype.

L'essentiel du chapitre 5 –

L'analyse des données est un des nombreux défis de CTA. Ce chapitre a décrit la méthode d'analyse classique des expériences d'astronomie gamma au sol, ainsi que la manière de l'optimiser. Les facteurs d'accélération obtenus ont montré à la collaboration que le HPC est indispensable pour CTA. Ils ont également réduis le nombre de CPUs prévu pour l'analyse sur site, d'une centaine par télescope à deux dixaines pour la totalité du site sud (le site le plus grand). Nous avons également introduis deux nouvelles méthodes de reconstruction stéréoscopique afin d'améliorer la résolution angulaire et la surface effective de l'analyse.

Chapitre 6

Analyse de données par comparaison d'images

Sujet du chapitre 6 –
Ce chapitre montre comment optimiser l'analyse de données avancée de CTA. Elle consiste à reconstruire les événements en comparant leurs images à des images simulées où l'on connaît à l'avance les paramètres de la particule initiale. Le jeux d'images qui décrit le mieux celles que l'on doit analyser donne les caractéristiques physiques de la particule incidente.

Sommaire

6.1 Méthode *Model++* de H.E.S.S.

L'analyse basée sur la comparaison d'images a pour but de reconstruire les événements en comparant leurs images à des images pré-calculées à l'aide de simulations Monte-Carlo (dans lesquels les paramètres initiaux des gerbes de particules sont connus). Ces images sont comparées entre elles et la configuration simulée la plus proche donne les paramètres physique de la particule initiale. La qualité de la comparaison, appelée vraisemblance, permet également de déterminer efficacement si la particule initiale est un rayon gamma ou non.

La figure 6.1 rappelle les paramètres géométriques qui seront utilisés tout au long ce de chapitre.

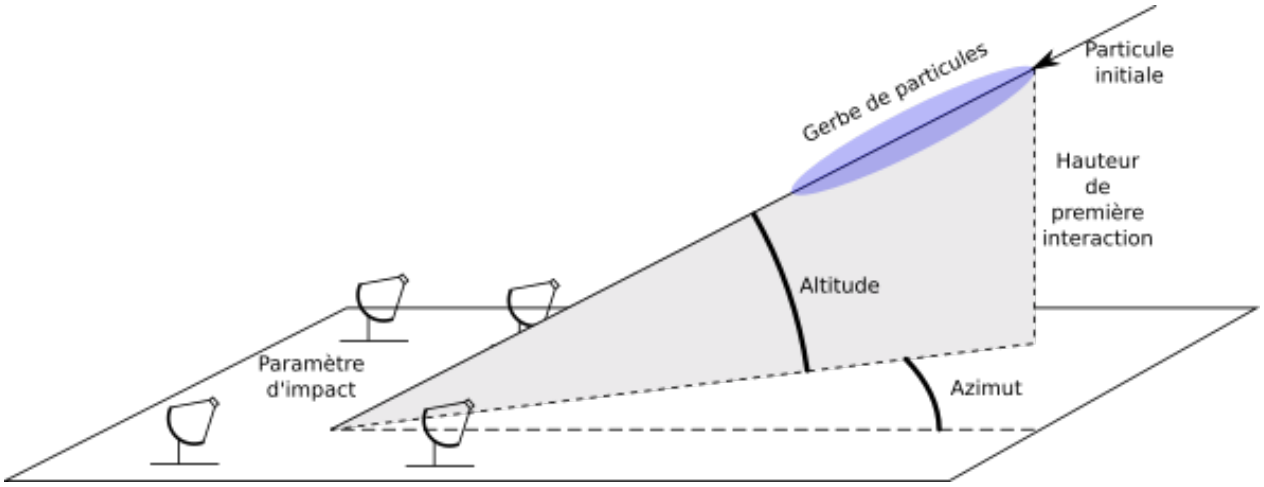


FIGURE 6.1 – Illustration des différents paramètres géométriques qui seront utilisés dans ce chapitre.

- **La hauteur de première interaction** décrit la hauteur à laquelle la particule initiale interagit avec l'atmosphère.
- **L'azimuth** est l'angle entre la direction nord et la visée des télescopes.
- **L'altitude** est l'angle entre le sol et la visée des télescopes. L'angle zénithal est égale à $\frac{\pi}{2} - altitude$.
- **Le paramètre d'impact** est l'intersection de l'axe de la gerbe de particules avec le sol.

La méthode *Model++*, utilisée dans l'analyse de données de l'expérience H.E.S.S. obtient des résultats de physique intéressants en améliorant principalement le rejet du bruit de fond [78]. Cette méthode est inspirée de l'analyse de données de l'expérience CAT [79].

La méthode *Model++* compare les images des événements aux réponses moyennées des télescopes, appelés modèles, avec un calcul de vraisemblance. Elle se décompose en plusieurs étapes :

— la création des modèles (voir section 6.1.1).
— L'algorithme de reconstruction (voir section 6.1.2).

La figure 6.2 montre le schéma de l'analyse *Model++* de H.E.S.S.

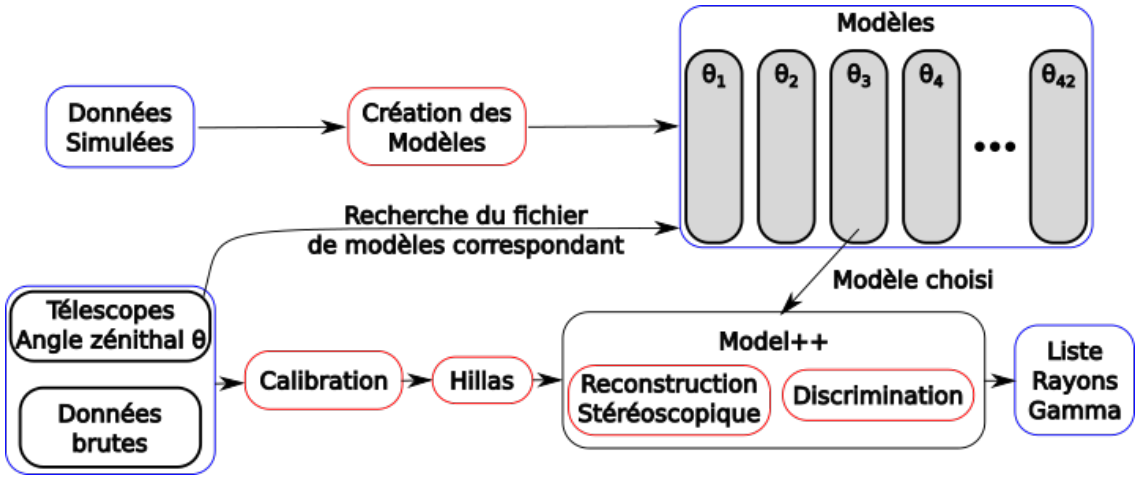


FIGURE 6.2 – Schéma de l'analyse de données *Model++* de H.E.S.S.

• Tout d'abord, les modèles sont créés à partir de simulations de gerbes de particules et sont regroupés en fichiers. Chaque fichier correspond à un angle zénithal, θ_i, particulier. Il sont au nombre de 42 dans H.E.S.S., entre 0 et 60 degrés.

• Ensuite, un fichier de modèles est choisi selon l'angle zénithal de l'observation à analyser. Les étapes classiques de calibration et du calcul des paramètres Hillas sont réalisées avant l'analyse *Model++*. Cette dernière compare les images des télescopes à analyser aux images pré-calculées des modèles afin de trouver le plus proche. Ensuite, les paramètres physiques du modèle le plus proche sont conservé et une vraisemblance entre ce modèle et les données quantifie la probabilité de l'événement d'avoir été produit par un rayon gamma. Finalement, une liste de rayons gamma est produite.

6.1.1 Création de l'espace des modèles

La création de l'espace des modèles est cruciale. Ils sont créés à partir des simulations Monte-Carlo de gerbes atmosphériques (KASKADE [80]), et décrivent un espace de paramètres à trois dimensions, pour un angle zénithal donné :

- E : l'énergie de première interaction (énergie de la particule initiale).
- ρ_{abs} : le paramètre d'impact (distance entre le centre du site et l'intersection du grand axe de la gerbe au sol).
- h : la hauteur de première interaction (hauteur à laquelle la particule initiale a interagi avec l'atmosphère).

La création des modèles requière des simulations de gerbes de particules produites par des rayons gamma dans la haute atmosphère. Elles sont moyennées par énergies, paramètres d'impact et hauteur de première interaction. Ensuite, ces gerbes de particules moyennées sont projetées sur un plan qui pointe dans la même direction que les télescopes (angle zénithal). Dans le cas de H.E.S.S., chaque angle zénithal est décrit par un fichier de modèles. Finalement, ce plan est subdivisé avec un pas très fin et est utilisé comme un histogramme à deux dimensions pour créer le modèle (voir figure 6.3). Les modèles sont des distributions très simples (voir figure 6.4). Chaque modèle correspond à une configuration en énergie (E), paramètre d'impact (ρ) et hauteur de première interaction (h). L'espace des modèles est fini, lorsque toutes les configurations physiques ont été simulées. La création de l'ensemble des modèles de H.E.S.S. représente de 50 à 100 jours de temps CPU [78].

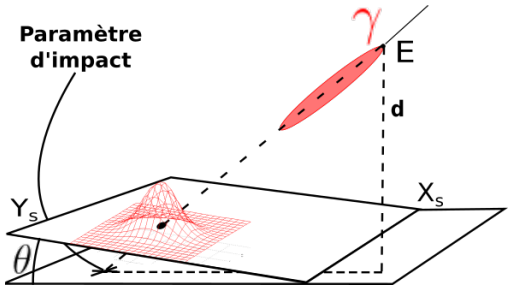


FIGURE 6.3 – Création d'un modèle. Des simulations de gerbes de particules sont projetées dans un plan qui est orienté selon l'angle zénithal des télescopes(θ). Ensuite, ce plan est subdivisé comme un histogramme pour créer les modèles pour chaque énergie, paramètre d'impact et hauteur de première interaction.

La résolution des paramètres physiques influence directement la résolution finale de l'analyse. Généralement, 75 énergies différentes sont utilisées avec une échelle logarithmique [1], 40 paramètres d'impact et 12 hauteurs de première interaction décrites avec une échelle linéaire.

1. L'échelle logarithmique est nécessaire car la plus petite énergie simulée est 0.005 TeV et la plus grande est 200 TeV. Cette échelle permet une erreur systématique de 5% dans chaque intervalle d'énergie.

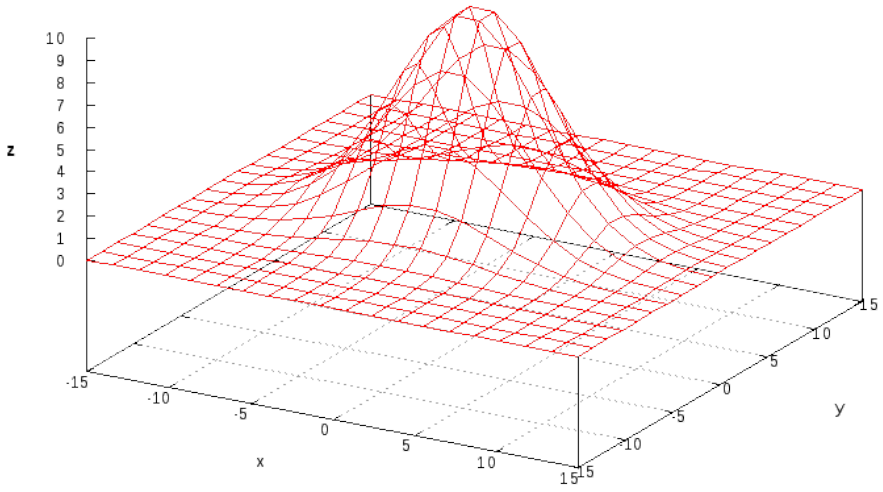


FIGURE 6.4 – Modèle utilisé dans la reconstruction *Model++*. Chaque modèle est un maillage simple qui décrit le signal reçu par un plan qui est orienté selon l'angle zénithal des télescopes. Les modèles sont plus pour moins détaillés en fonction de l'énergie de la particule initiale et de son étalement sur le plan considéré.

6.1.2 Algorithme de reconstruction

L'algorithme de reconstruction de la méthode *Model++* est illustré par la figure 6.5.

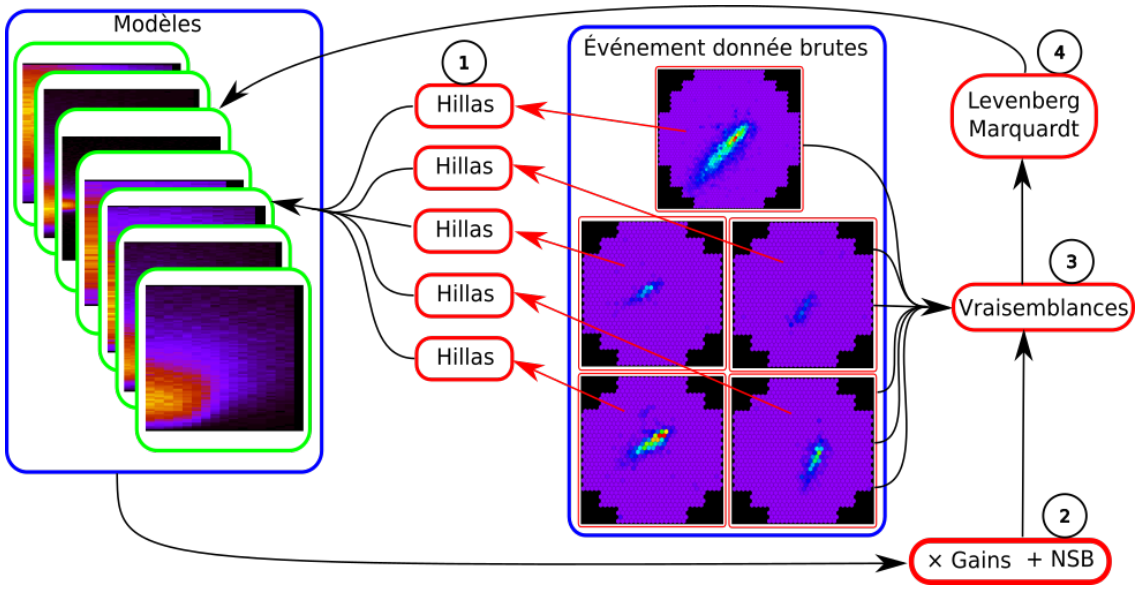


FIGURE 6.5 – Étapes qui déterminent le maximum de vraisemblance des modèles par rapport aux données brutes à analyser.

Cet algorithme se compose des étapes suivantes :

1. Le premier modèle est choisi avec les paramètres Hillas (voir chapitre 5).
2. Ce modèle est ajusté dans les caméras par translation et rotation. Il est ensuite dégradé avec les valeurs réelles des gain, piédestaux et de sensibilité des pixels.
3. Les données réelles sont alors comparées au modèle dégradé grâce à plusieurs vraisemblances.
4. Ensuite, le prochain modèle est choisi en utilisant l'algorithme de minimisation de Levenberg-Marquartd [81] [82] (voir annexe B.4).
5. Lorsque le maximum de vraisemblance est atteint, l'algorithme s'arrête et les paramètres du modèle de plus semblable à l'événement réel servent à décrire celui-ci.
6. L'étape de discrimination est similaire à celle de l'analyse Hillas : elle utilise des tables où des seuils ont été déterminés sur les paramètres Hillas ainsi que sur les vraisemblances calculées par *Model++* afin de séparer les protons des rayons gamma.

6.1.2.1 Interpolation des modèles

L'interpolation des modèles de H.E.S.S. convertie un modèle en caméra hexagonale. Elle calcule la contribution du modèle donné dans chaque pixel de la caméra en estimant la superposition du modèle sur celle-ci. Pour ce faire, une interpolation linéaire suréchantillonne

le modèle de sorte à avoir une centaine de mailles par pixel. Ensuite le modèle est placé dans la caméra à la position désirée puis tourné. Enfin, la superposition du modèle sur la caméra est calculée. Cette image peut maintenant être comparée aux données observées.

6.1.2.2 Déplacement dans l'espace des modèles

La recherche des modèles ajuste les paramètres (E, ρ, h) mais aussi la position de la gerbe dans les télescopes $(g_x, g_y)_i$ et son orientation, ϕ_i, où $2 \leq i \leq N_{tel}$, où N_{tel} est le nombre de télescopes dans l'événement à traiter.

La méthode de Levenberg-Marquardt, basée sur celle de Newton-Raphson, est utilisée pour explorer cet espace à six dimensions. Son but est de minimiser la fonction de vraisemblance entre le modèle courant et les images à analyser.

Soit un vecteur de paramètres $\beta = (d, \rho, Y, g_x, g_y, \phi)$, où :

— d : hauteur de première interaction.
— ρ : distance entre le paramètre d'impact et le télescope courant.
— $Y = \ln\left(\frac{E}{E_c}\right)$: description linéaire de l'énergie de la particule initiale, où E est l'énergie du rayon gamma incident et E_c est une énergie critque [2].
— (g_x, g_y) : position du barycentre de la gerbe de particules dans la caméra.
— ϕ : orientation de la gerbe de particules dans la caméra en radian.

L'équation de Levenberg-Marquardt donne le vecteur de modification, $\delta\beta$, de β :

$$\delta\beta \quad = \quad \left(J^T J + \lambda D\right)^{-1} J^T r \tag{6.1}$$

Où r est le vecteur de résidus entre l'image de la caméra à analyser et le modèle courant (voir section 6.1.2.3), J est la matrice Jacobienne du système, calculée dans la section 6.1.2.4, J^T la matrice transposée de J, D et la matrice identité et λ et le paramètre de Marquardt (voir annexe B.4).

6.1.2.3 Vraissemblance entre les modèles et les données réelles

La vraisemblance entre le modèle et les données est une manière de quantifier leur similarité. Elle est calculée par la somme des erreurs entre le signal attendu dans chaque pixel et celui prédit par le modèle.

Le vecteur de résidus donne l'erreur entre le nombre de photons attendus dans un pixel (donné par le modèle) et le nombre réel de photons dans les données, $\mathbf{s}$.

La méthode la plus simple est de trouver le maximum de vraisemblance. Soit un modèle $\mathcal{M}$ converti en caméra, et un signal calibré , $\mathbf{s}$, dans la caméra, et $\mathbf{b}$ un vecteur de booléens

2. Ce changement de variable permet uniquement de simplifier les calculs.

qui indique quels sont les pixels que l'on doit prendre en compte pour la comparaison (1), ou non (0).

Le vecteur de résidus est calculé comme suit :

$$\mathbf{v}_i = (\mathcal{M}_i - \mathbf{s}_i)\,\mathbf{b}_i \quad \text{si } 0 \leq i < N_{pixel} \tag{6.2}$$

Où N_{pixel} est le nombre de pixels dans la caméra.

Cette méthode est utilisée pour vérifier l'algorithme mais ne tient pas compte de la physique du détecteur. Un photomultiplicateur a une sensibilité : une probabilité de détecter un photon (40% dans le cas de H.E.S.S. I). Cela implique que toutes les erreurs n'ont pas la même importance.

La reconstruction actuelle de *Model++* utilise un calcul de vraisemblance plus complet qui est décrit dans ce qui suit.

Ce calcul de vraisemblance évalue la différence de signal entre chaque pixel et le signal attendu par le modèle en prenant en compte les fluctuations physiques de ce dernier. Pour ce faire, il calcule la probabilité de chacun des pixels d'obtenir un nombre arbitraire de photons.

Un pixel à la position p, a une largeur de piédestal, σ^p_{ped}, et une largeur de pic de photons [3], σ^p_{am}, et une incertitude sur la calibration σ^p_{calib}. Dans ce pixel, le signal mesuré est x_p et le signal attendu, déterminé par le modèle le plus proche, est μ_p. Ainsi, l'erreur entre le modèle et le signal mesuré peut être calculée comme :

$$\sigma_i = i \cdot \left(\sigma^p_{am}\right)^2 + \left(\sigma^p_{ped}\right)^2 + \left(\sigma^p_{calib}\right)^2 \quad 0 \leq i < N \tag{6.3}$$

$$t_i = \frac{i \cdot \ln \mu_p - \ln(i!) - \frac{1}{2}\left(x_p - i\right)^2}{\sigma_i} \tag{6.4}$$

$$t_{max} = \max\left(t_i\right), \tag{6.5}$$

$$\varepsilon_p = \left(\sum_{i=0}^{N} \frac{e^{t_i}}{\sqrt{\sigma_i}}\right) e^{-t_{max}} \tag{6.6}$$

$$l_p = 2 \cdot \mu \ln(2\pi) - 2 \cdot \left[\ln\left(\varepsilon_p\right) + t_{max}\right] \tag{6.7}$$

Le paramètre l_p décrit la vraisemblance entre les signaux attendus et détectés dans un pixel donné en fonction du modèle considéré. Le maximum, N, doit être choisi judicieusement afin de minimiser le nombre de calcul tout en prenant en compte tous les termes non négligeables ; actuellement, $N = 5\,000$.

3. Ils sont appelés photo-électrons car ils s'agit d'un signal électrique qui représente un nombre de photons tombés sur le télescope.

La vraisemblance l entre toute la caméra et le modèle est donnée par la somme sur tous les pixels :

$$l = \sum_{p=0}^{N_{pixels}} l_p \tag{6.8}$$

Ce calcul de vraisemblance produit de meilleurs résultats que la méthode précédente décrite au début de cette section.

6.1.2.4 Calcul de la matrice Jacobienne

Une matrice Jacobienne est une matrice de dérivés d'une fonction à N dimensions (voir annexe B.1).

Le calcul de la matrice Jacobienne J est critique car elle détermine le chemin de convergence dans l'espace à six dimensions pour trouver le modèle le plus probable en fonction des données mesurées.

Dans notre cas, seules les dérivées premières sont utilisées car cela simplifie le calcul. Cette matrice a N lignes (ce qui correspond au nombre de pixels), et 6 colonnes (les six paramètres à ajuster).

Soit un modèle $\mathcal{M}$ et un modèle $\mathcal{M}_p$ qui diffère de δ_p du modèle précédent sur le paramètre p. La dérivée pour tous les pixels est données par :

$$\left(\frac{\partial F}{\partial p}\right)_i = \frac{(\mathcal{M}_p)_i - \mathcal{M}_i}{\delta_p} \quad \text{pour tous les pixels } 0 < i \leq N \tag{6.9}$$

Où p est un paramètre de l'espace des modèles. La matrice Jacobienne J est :

$$J = \begin{pmatrix} \frac{(\mathcal{M}_d)_1 - \mathcal{M}_1}{\delta_d} & \frac{(\mathcal{M}_\rho)_1 - \mathcal{M}_1}{\delta_\rho} & \frac{(\mathcal{M}_Y)_1 - \mathcal{M}_1}{\delta_Y} & \frac{(\mathcal{M}_{g_x})_1 - \mathcal{M}_1}{\delta_{g_x}} & \frac{(\mathcal{M}_{g_y})_1 - \mathcal{M}_1}{\delta_{g_y}} & \frac{(\mathcal{M}_\phi)_1 - \mathcal{M}_1}{\delta_\phi} \\ \frac{(\mathcal{M}_d)_2 - \mathcal{M}_2}{\delta_d} & \frac{(\mathcal{M}_\rho)_2 - \mathcal{M}_2}{\delta_\rho} & \frac{(\mathcal{M}_Y)_2 - \mathcal{M}_2}{\delta_Y} & \frac{(\mathcal{M}_{g_x})_2 - \mathcal{M}_2}{\delta_{g_x}} & \frac{(\mathcal{M}_{g_y})_2 - \mathcal{M}_2}{\delta_{g_y}} & \frac{(\mathcal{M}_\phi)_2 - \mathcal{M}_2}{\delta_\phi} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{(\mathcal{M}_d)_N - \mathcal{M}_N}{\delta_d} & \frac{(\mathcal{M}_\rho)_N - \mathcal{M}_N}{\delta_\rho} & \frac{(\mathcal{M}_Y)_N - \mathcal{M}_N}{\delta_Y} & \frac{(\mathcal{M}_{g_x})_N - \mathcal{M}_N}{\delta_{g_x}} & \frac{(\mathcal{M}_{g_y})_N - \mathcal{M}_N}{\delta_{g_y}} & \frac{(\mathcal{M}_\phi)_N - \mathcal{M}_N}{\delta_\phi} \end{pmatrix} \tag{6.10}$$

La matrice $J^T J$ est une matrice carrée 6×6. Pour être exacte, les données qui concernent les pixels rejetés n'ont pas été prises en compte dans ce calcul car elles sont préalablement remplacées par 0. Donc, les lignes de la matrice J sont multipliées par le terme $\mathbf{b}_i$ donné par l'équation 6.2.

Les pas δ_d, δ_ρ et δ_Y sont fixés par le fichier de modèles considéré car ils représentent les intervalles sur les axes d (hauteur de première interaction), ρ (paramètre d'impact) et Y (énergie de la particule initiale) respectivement.

Cependant, les paramètres δ_{g_x}, δ_{g_y} et δ_ϕ doivent être ajustés car il déterminent la vitesse de convergence de l'algorithme de Levenberg-Marquardt.

Si le calcul est effectué sur la matrice défini par l'équation 6.10, il en résulte une matrice mal conditionnée[4]. La matrice $J^T J$ contient des valeurs importantes (supérieures à 10^{11}), cela implique que la matrice inverse $\left(J^T J + \lambda D\right)^{-1}$ aura des valeurs proches de 0.

Une matrice d'échelle est utilisée pour résoudre ce problème. La matrice d'échelle $\left(J^T J + \lambda D\right)^{-1}$ produit une matrice $J^T J$ dont les coefficients sont entre 0 et 10, cela garanti un bon conditionnement des matrices J, J^T, $J^T J$ et $\left(J^T J + \lambda D\right)^{-1}$.

6.1.2.5 Stereoscopie et discrimination avec *Model++*

Comme nous l'avons mentionné précédemment, la première étape de *Model++* utilise directement les images calibrées des télescopes d'un événement en même temps. Dans ce cas, l'altitude, l'azimut et l'énergie sont déjà déterminés par l'analyse Hillas. Comme pour l'analyse Hillas, la dernière étape est la discrimination. Pour cela, les vraisemblances calculées sont utilisées pour évaluer la qualité de l'ajustement entre les données réelles et le modèle. Puisque les modèles sont créés à partir de simulations de rayons gamma, une mauvaise vraisemblance indique probablement que l'événement n'est pas un rayon gamma.

La résolution angulaire et la résolution en énergie sont toutes deux globalement améliorées par *Model++* comparé à une méthode à base de moments comme Hillas grâce à un meilleur rejet du bruit de fond. Ce constat est exact à basse énergie (80 GeV for H.E.S.S.).

6.1.3 Temps de calcul

Model++ est une analyse complexe. Elle nécessite typiquement 12 heures[5] de calcul pour analyser un seul fichier d'observation de données H.E.S.S. (une observation dans H.E.S.S. dure 28 minutes) soit 5.5 GO de données.

6.2 Problématique de *Model++* pour CTA

Comme nous l'avons vu dans la section précédente, la méthode *Model++* effectue un ajustement stéréoscopique des images des télescopes d'un événement. Or, la structure de données de CTA ne sera pas ordonnée par événement mais par télescope car cela permet une écriture plus rapide des fichiers au vu des flux de données à traiter et facilite la gestion des temporaires utilisés lors des calculs.

L'utilisation de type Grille implique d'effectuer un tri des données au fur et à mesure de leur lecture. Comme nous l'avons détaillé dans le chapitre 3, le format de données devra

4. Matrice où le rapport entre la plus grande valeur et la plus petite est trop grand par rapport à la précision de la machine ce qui produit de lourdes erreurs de calcul.

5. Le temps de calcul est donné sans optimisations car le format de données est si complexe qu'aucune optimisation du compilateur n'améliore le temps de l'analyse.

permettre un chargement partiel du fichier en mémoire RAM. Or, si nous nous limitons à une utilisation classique de la Grille (2 GO de mémoire RAM par thread), le temps de lecture des fichiers sera extrêmement coûteux à cause de la mauvaise utilisation du bus de lecture du disque dur. Sans compter le fait que tous les fichiers à traiter devront être transférés sur le même disque dur, ou au moins lus à distance. Donc, les temps de lecture des fichiers seront dégradés au point que l'optimisation de l'analyse de données n'apportera aucune accélération.

La solution permettant de réduire le problème du transfert des données est de les charger complètement en mémoire RAM. Dans ce cas, les 100 fichiers du site nord ou les 19 fichiers du site sud représentent 200 GO ou 38 GO dans le meilleur des cas (2 GO par fichier), et 500 GO ou 95 GO dans le pire des cas (5 GO par fichier) respectivement. Et cela, pour ne traiter qu'un vingtième ou un cinquantième d'une observation moyenne (30 minutes de données).

Une telle quantité de mémoire serait disponible sur une architecture distribuée, où les différents threads et noeuds de calcul communiqueraient leurs résultats lors de l'utilisation de l'algorithme de Levenberg-Marquardt. Ce genre d'utilisation ne sera pas possible si la Grille n'évolue pas, mais pourrait être envisageable lors d'une analyse sur site, bien qu'elle nécessiterait d'importantes optimisations afin d'être utilisable dans un centre de calcul sur site. Dans ce dernier cas, le problème de la réanalyse des données reste entier.

La problématique du volume de données à traiter sera d'autant plus grande que les modèles devront également être chargés en mémoire RAM. L'analyse de données de H.E.S.S. utilise des fichiers de modèles de 5.5 GO afin d'analyser un fichier de 6.5 GO de données brute. Ces 12 GO de données sont bien au delà d'une utilisation normale de la mémoire pour la Grille ou un centre de calcul comme le CCALI (voir section 1.5). Or, les modèles utilisés pour CTA seront *a priori* plus gros que ceux de H.E.S.S. [64] ainsi que les fichiers de données. Les programmes d'analyse de données devront évolués afin de mutualiser l'utilisation de leur mémoire.

En dernier lieu, la comparaison des pixels entre eux par le calcul de vraisemblance sera d'autant plus coûteux que le nombre de pixels est plus important sur les caméras de CTA.

Le travail qui suit propose une méthode pour résoudre ce problème.

6.3 La décomposition en valeurs singulières

Nous avons vu dans la section précédente que la comparaison d'image pixel par pixel est lente dans H.E.S.S. et le sera d'avantage dans CTA. Néanmoins, le concept de comparaison d'image est efficace du point de vue des résultas de physiques et doit donc être conservé dans ce qui suit.

Nous introduisons une méthode de comparaison d'images qui nécessite moins de données. Une telle comparaison peut être effectuée après une projection des images dans un sous-espace. La taille de ce sous-espace détermine le nombre de valeurs à comparer ainsi que le niveau de détail que l'on souhaite conserver.

La transformée de Fourier (TF) est une méthode qui permet une projection de l'espace du temps dans celui des fréquences (figure 6.6). Dans cette méthode le signal est décomposé sur un espace de fréquences. Dans l'espace projeté, chaque valeur correspond à la proportion de sa fréquence associée dans le signal temporel de départ. Dans ce cas, l'ensemble des valeurs dans l'espace fréquentiel est appelé spectre.

La comparaison des spectres de deux images suffit à décrire leur similarité en terme de fréquences. Il est même possible de sélectionner un niveau de détail en n'autorisant que les fréquences qui représentent une certaine proportion de l'espace d'arrivée.

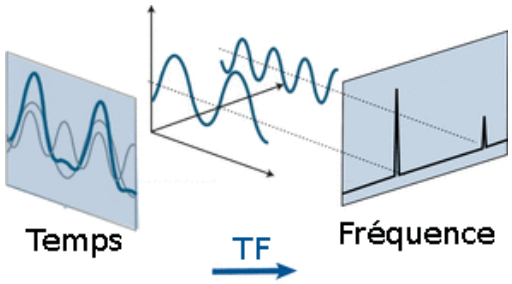


FIGURE 6.6 – Projection d'un espace en temps à un espace en fréquences à l'aide d'une TF.

La description de l'image en fréquences permet de chercher des zones de différentes tailles mais *a priori* pas leurs positions. Cette propriété sera détaillée plus loin dans ce manuscrit.

Nous avons effectué des tests au sein de notre analyse de données afin de vérifier la faisabilité de la méthode. Nous avons conclu que les spectres obtenus étaient pertinents mais que le temps de calcul, à une ou deux dimensions, était bien trop long comparé à nos exigences et aux objectifs de CTA. Nous avons donc cherché une méthode plus rapide.

Nous nous sommes alors tourné vers le calcul de valeurs propres de matrices qui est la généralisation de la TF sur des matrices carrées. Et plus particulièrement à la décomposition en valeur singulière (SVD) qui est la généralisation du calcul de valeurs propres pour des matrices rectangulaires.

Si l'on fait agir une matrice à deux dimensions sur le cercle unité on obtient une ellipse (figure 6.7). L'orientation du grand axe de cette ellipse est le premier vecteur propre, et sa longueur associée est la première valeur propre de cette matrice. De la même manière, l'orientation de son petit axe est le second vecteur propre, et sa norme associée est la seconde valeur propre.

Il est donc possible d'écrire cette matrice comme produit d'une matrice de rotation puis d'une matrice d'échelle qui sont respectivement les vecteurs propres et les valeurs propres.

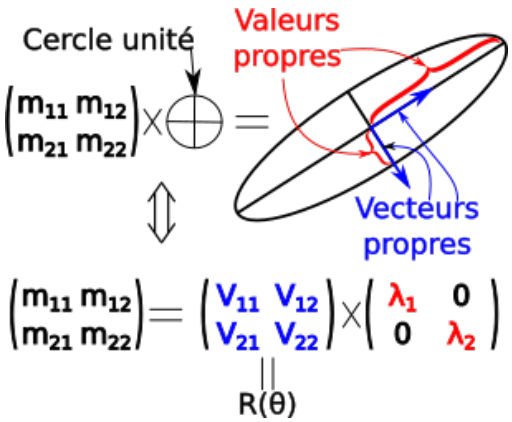


FIGURE 6.7 – Interprétation géométriques des valeurs et des vecteurs propres d'une matrice M.

Dans le cas à N dimensions, les valeurs et les vecteurs propres d'une matrice permettent de décrire celle-ci comme une hyper-ellipse.

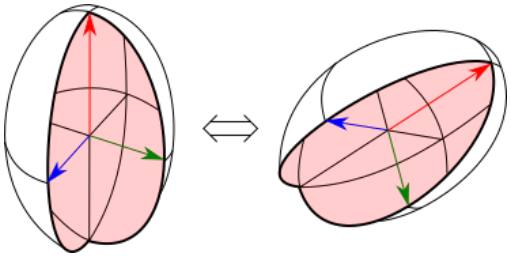


FIGURE 6.8 – La comparaison des valeurs singulières seules (normes des vecteurs colorés) est insensible à la rotation des matrices mais sensible à la forme des ellipses qu'elles produisent.

Ainsi, la comparaison des spectres de valeurs singulières permet d'évaluer les formes des hyper-ellipses et non leur orientation (voir figure 6.8), à la manière de la TF qui compare également la forme d'un signal et non sa position spatiale. La seule différence avec la TF est que les vecteurs propres ne sont *a priori* pas égaux entre deux matrices. Cela implique que l'espace de décomposition n'est pas exactement le même. Cependant l'espace des vecteurs propres est orthogonal et leurs valeurs propres associées sont ordonnées par magnitude (généralement par ordre décroissant). Nous avons donc effectuer les premiers tests en comparant uniquement les valeurs propres.

Le calcul des valeurs et des vecteurs propres peut être effectué formellement pour des matrices $n \times n$ si $n \leq 5$. Dans le cas contraire, le théorème d'Abel impose que la méthode qui les détermine soit itérative. Ainsi, on détermine en priorité les extrémités du spectre (voir figure 6.9). Le plus grand axe de l'hyper-ellipse est déterminé en priorité, ensuite le plus petit (orthogonal au plus grand), après, le deuxième plus grand (orthogonal aux deux précédents et ainsi de suite jusqu'à ce que tous les axes soient déterminés.

La comparaison des spectres de valeurs propres peut s'avérer difficile car elles sont potentiellement complexes. Il serait plus aisé de comparer des nombres réels entre eux.

La SVD calcule les valeurs propres d'une matrice symétrique obtenue par la multiplication de la matrice de départ par sa transposée (ou inversement). Par définition, les valeurs propres d'une matrice symétriques sont réelles positives, elles deviennent les valeurs singulières de la matrice de départ. Cela permet de décomposer une matrice quelconque sur un spectre de

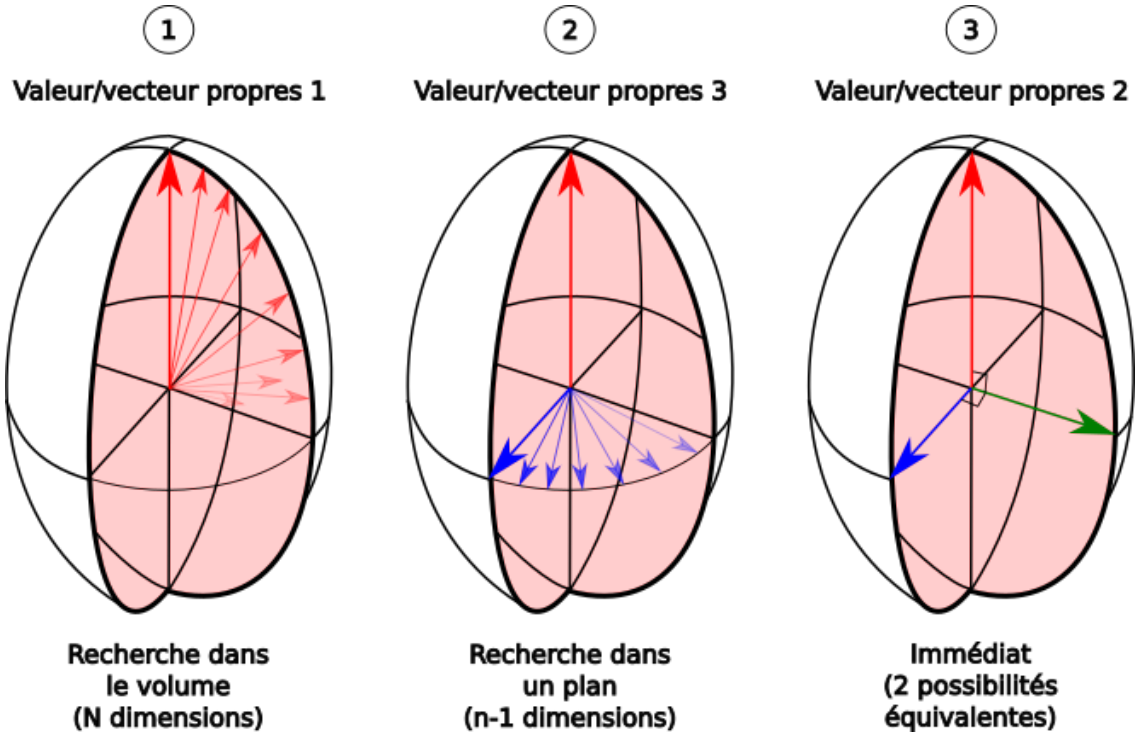


FIGURE 6.9 – Illustration de la détermination iterative des valeurs et vecteurs propres dans un cas à trois dimensions.

valeurs réelles positives.

La SVD est la généralisation du calcul des valeurs propres pour des matrices rectangulaires. Soient des matrices $A \in M_{n,m}(\mathbb{R})$, $U \in O_{n,n}(\mathbb{R})$ (matrice orthogonal $n \times n$), $V \in O_{m,m}(\mathbb{R})$, $D \in D_{n,m}(\mathbb{R})$ (matrice diagonale), la SVD de la matrice A est donnée par :

$$A = UDV^T \tag{6.11}$$

L'algorithme, plus détaillé, calcule d'abord les valeurs propres de la matrice $M = A^T A$ ou $M = AA^T$ de sorte que la matrice M soit carrée de taille $\min(n, m)$. Nous nous plaçons dans le cas $n > m$, alors $M = A^T A$. Par définition, la matrice $M \in M_{m,m}(\mathbb{R})$ est symétrique, ses valeurs propres sont donc des réels positifs.

La méthode QR [83] permet de calculer les valeurs propres de M. L'algorithme QR est basé sur la méthode QR qui décompose une matrice carrée $M \in M_{m,m}(\mathbb{R})$ comme :

$$M = QR \tag{6.12}$$

Où $Q \in O_{m,m}(\mathbb{R})$ et $R \in R_{m,m}(\mathbb{R})$(matrice triangulaire supérieure).

L'algorithme QR est itératif sur n, et est défini comme :

$$M_n = Q_n R_n \quad (6.13)$$
$$M_{n+1} = R_n Q_n \quad (6.14)$$

Avec $M_1 = M$. Quand $n \to \infty$ la matrice M_n tend vers une matrice diagonale. Puisque les transformations sont orthogonales par définition, car les matrices Q_n sont orthogonales, les valeurs propres de M sont les mêmes que M_∞. Or, les valeurs propres de M_∞ sont sur sa diagonale car M_∞ est une matrice diagonale.

Généralement, l'algorithme QR s'arrête quand les valeurs absolues des termes de M_n en dehors de sa diagonale sont plus petits qu'un ϵ qui définie la précision du calcul souhaité.

Dès lors, les valeurs propres de M sont les valeurs singulières de A, et les vecteurs propres de M composent la matrice V. La dernière étape consiste à calculer la matrice U avec :

$$U = AVD^{-1} \quad (6.15)$$

Puisque la matrice D est diagonale, D^{-1} se calcule facilement.

6.3.1 L'optimisation de la comparaison des images

Les images que nous devons analyser n'ont pas beaucoup de détails, hormis le bruit qui est plutôt gênant. Nous pouvons donc les comparer en utilisant les valeurs singulières. Ainsi, le spectre SVD, relié à une configuration physique donnée, est comparé aux spectres SVD de l'événement à analyser.

L'étude préliminaire de la section 6.3.2 montre que seulement quelques valeurs singulières peuvent être utilisées. Les différents niveaux de parallélisme de l'algorithme sont présentés dans la section 6.4. La section 6.5 expose une méthode de reconstruction basée sur des spectres de valeurs singulières pré-calculées. La section 6.6 présente une autre méthode qui utilise un dictionnaire de spectres de valeurs singulières où chacun des spectres correspond à un jeu de paramètres physiques. Finalement, la section 6.9 conclue sur ces deux approches.

6.3.2 Étude préliminaire

Nous avons réalisé plusieurs études afin de déterminer le lien entre les spectres SVD et les paramètres physiques des gerbes de particules ou ceux des images des télescopes. La section 6.3.2.1 expose l'étude réalisée à partir des simulations fonctionnelles utilisées dans l'expérience H.E.S.S. L'étude sur les simulations de CTA est présenté dans la section 6.3.2.2.

6.3.2.1 Étude à partir d'images de gerbes moyennées H.E.S.S.

Nous voulions déterminer comment réagit la SVD aux images de gerbes de particules. Nous avons commencé avec le modèle de l'expérience H.E.S.S. [78] pour plusieurs raisons :

— Le modèle est fonctionnel et est actuellement utilisé en production. La physique qui y est décrite est donc pertinente dans le cadre de H.E.S.S. Or, CTA étend l'intervalle de sensibilité en énergie de H.E.S.S., ce qui permet une évaluation raisonnable des performances de la SVD.
— L'intervalle d'énergie décrit est certes plus petit, mais mieux défini en terme de statistiques (les simulations de CTA ont été longues à produire, il nous fallait donc un moyen de substitution en attendant).
— Le modèle a été produit avec une statistique suffisante pour toutes les configurations physiques que nous voulons explorer.

Les études de l'influence géométrique de la gerbe, position et rotation dans la caméra, ont été réalisées en interpolant les modèles (section 6.1.1) dans les caméras LST-CAM de CTA. Cette interpolation permet de translater et de tourner le modèle de manière arbitraire et rapide.

Nous avons commencé par évaluer l'effet de la pixelisation sur le comportement de la SVD lorsque l'on tourne une gerbe de particules dans la caméra de sorte qu'elle soit entièrement contenue dans la caméra de sorte à supprimer les effets de bord qui pourraient affecter les valeurs singulières (figure 6.10).

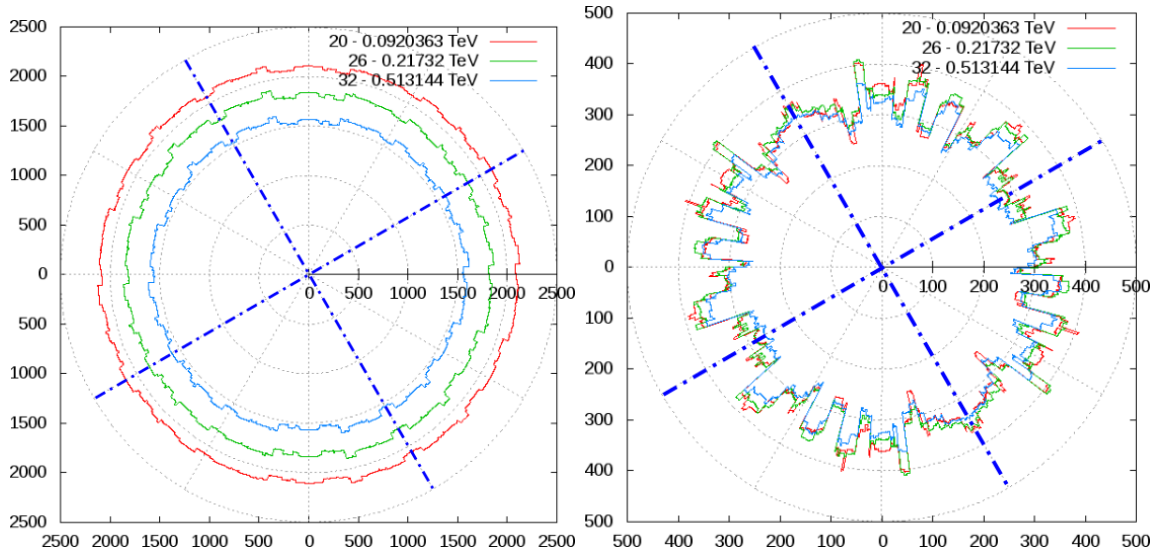


FIGURE 6.10 – Variation des valeurs singulières en fonction de la rotation de la gerbe de particules dans la caméra. **À gauche :** variation de la première valeur singulière. **À droite :** variation de la deuxième valeur singulière. Les axes de symétrie sont indiqués en pointillés bleu.

Ceci montre que la première valeur singulière est en moyenne très peu influencée par la rotation de l'image dans la caméra, tandis que la deuxième l'est d'avantage. Cette symétrie est intéressante du point de vue de la création de modèles *via* des simulations spécifiques car une symétrie implique qu'il est possible d'en utiliser une quantité plus faible pour atteindre un résultat équivalent sans symétrie.

Nous avons ensuite étudié plusieurs translations de gerbes dans les caméras. La figure 6.11 montre le résultat obtenu par la translation sur l'axe des abscisses.

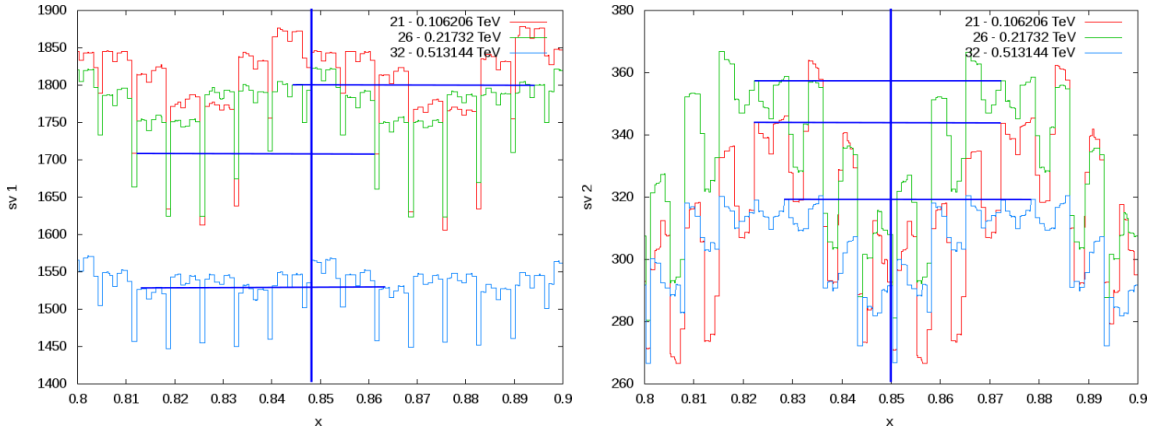


FIGURE 6.11 – Zoom de la variation des quatre premières valeurs singulières en fonction de la translation de la gerbe de particules sur l'axe des abscisses de la caméra. **À gauche :** variation de la première valeur singulière. **À droite :** variation de la deuxième valeur singulière. La symétrie de translation est indiquée par les axes bleus.

Bien que la première valeur singulière ne permette pas de distinguer une gerbe à 106 GeV d'une autre à 217 GeV (deux modèles adjacents en énergies), les graphiques zoomés semblent faire apparaître des motifs qui se répètent. Ces derniers sont d'avantage visibles pour la deuxième valeur singulière. Ceci tendrait à montrer qu'une symétrie supplémentaire existe. Cependant, la surface de détection d'un télescopes, ou acceptance, varie en fonction du rayon de la caméra, ce qui explique bien la symétrie de rotation mais annule la possible symétrie de translation.

Nous avons ensuite testé la séparation des énergies avec les spectres SVD.

Le résultat de la première étude, figure 6.12, montre la séparation des valeurs singulières sur sept énergies différentes, entre 51 GeV et 1.05 TeV.

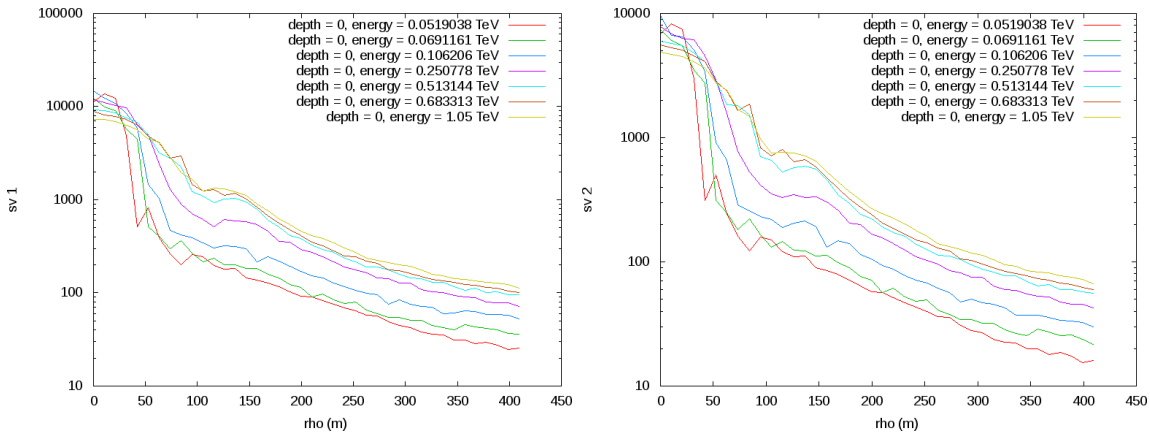


FIGURE 6.12 – Variation des premières (en haut à gauche), deuxièmes (en haut à droite), valeurs singulières en fonction de différentes énergies et différents paramètres d'impact.

Nous remarquons que les premières valeurs singulières semblent avoir la meilleure sensibilité à l'énergie de la particule initiale. En première approximation, les premières valeurs singulières varient avec le paramètre d'impact (*rho*) comme l'amplitude du signal dans la caméra. Cela implique que la magnitude de ces valeurs singulières décroit lorsque le paramètre d'impact augmente.

Ensuite, ces valeurs singulières séparent des énergies croissantes par ordre croissant quand le paramètres d'impact est supérieur à 50 mètres et quasiment par ordre décroissant si le paramètre d'impact est inférieur à 25 mètres.

La séparation des énergies sera probablement dégradée quand le paramètre d'impact est compris entre 25 et 50 mètres car les valeurs singulières se croisent. Or, la séparation des énergies implique que les spectres ne doivent pas se confondre.

Nous avons répété cette étude pour différentes hauteurs de première interaction et différents paramètres d'impact et il apparaît que les spectres SVD se croisent lorsque le paramètre d'impact est inférieur à 50 mètres.

Cela implique que des coupures devront probablement être appliquées pour rejeter les petites distances d'impact, ou pour les traiter différemment, afin d'améliorer la qualité de la reconstruction.

La reconstruction de la hauteur de première interaction est elle aussi importante puisqu'elle permet de lever la dégénérescence sur l'énergie au même titre que le paramètre d'impact. À l'origine, ce paramètre permet de vérifier l'angle zénithal reconstruit de la gerbe de particules. Il est généralement découpé en moins d'intervalles (typiquement 12) que l'énergie (typiquement 75) ou le paramètre d'impact (typiquement 40) dans le *Model++* classique (voir section 6.1).

La figure 6.13 montre comment la première valeur singulière peu séparer différentes énergies lorsque la hauteur de première interaction change. Chaque graphique de cette figure décrit d'autres paramètres d'impact simulés avec la même énergie.

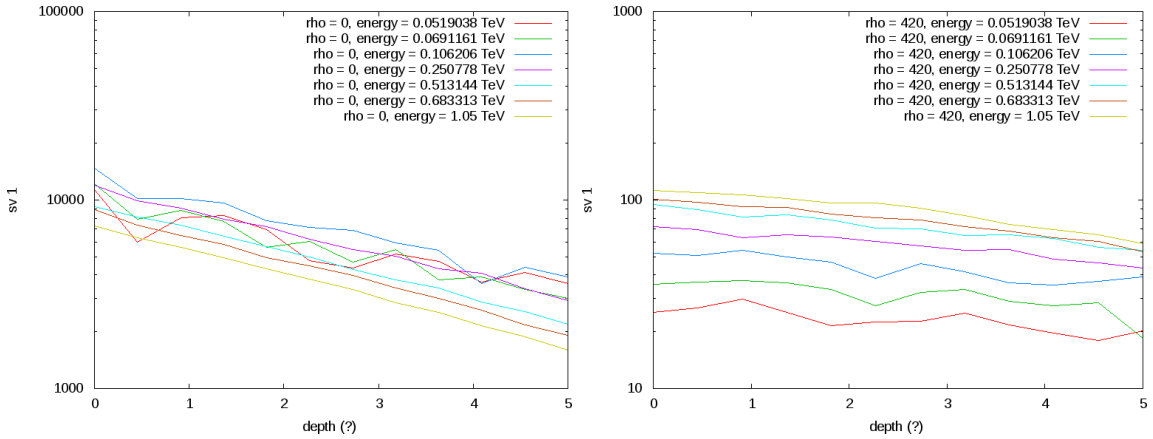


FIGURE 6.13 – Variation de la première valeur singulière pour différents paramètres d'impact en fonction de la hauteur de première intéraction.

Le graphique à gauche montre les variations de la première valeur singulière correspondant à un paramètre d'impact à 0 mètre. Il montre également qu'à de faibles énergies (52 GeV ou 69 GeV), les oscillations de la première valeur singulière sont importantes donc la reconstruction en énergie ne sera pas efficace. La reconstruction à plus haute énergie est de meilleure qualité.

Nous avons vu précédemment qu'une tendance similaire se dégage lorsque le paramètre d'impact est inférieur à 50 ou 60 mètres. Le graphique à droite montre que la séparation des énergies est clairement améliorée lorsque le paramètre d'impact augmente (420 m).

Analyser les gerbes de particules complètement contenues dans les caméra n'est pas suffisant car beaucoup d'entre elles ne sont que partiellement détectées à cause de la limitation du champs de vue des caméras [6].

Nous avions commencé par rejeter ces événements puisqu'ils étaient mal reconstruits par l'analyse de données classique (voir chapitre 5) mais puisque l'objectif de CTA est d'améliorer d'un ordre de grandeur la sensibilité aux rayons gamma, tous ces événements devaient être conservés. En conséquence, l'analyse de donnée devait être améliorée afin de les prendre en compte. Cela implique que notre analyse SVD doit également prendre en compte ces événements.

Dans la méthode classique de *Model++* (voir section 6.1), la position des gerbes de particules est ajustée dans la caméra pour chaque image. Cette méthode n'a donc pas besoin de discrétiser grossièrement les paramètres x, y et ϕ, ce qui lui permet d'ajuster des gerbes coupées.

Au contraire, notre modèle SVD utilise des spectres de valeurs singulières à la place des pixels des images, donc les positions et rotations des modèles doivent être prises en compte en amont dans le model.

Cela implique que le modèle SVD conserve moins de données que la méthode *Model++* classique, mais en contre partie, plus de configurations doivent être simulées pour tenir compte des rotations et des translations dans les caméras.

Les gerbes coupées de plus haute énergie ne sont pas un problème car leur direction est généralement bien défini dans la caméra, seule la détermination de leur énergie est biaisée car l'amplitude du signal est moindre comparer à ce qu'il devrait être. Les gerbes de plus faible énergie sont statistiquement moins coupées car elles sont plus petites, mais lorsqu'elles le sont, la proportion de signal masqué est très importante.

Nous avons utilisé notre interpolation de modèles de sorte à ce qu'ils soient coupés dans les six encoches des caméras LST-CAM (figure 6.14) lors d'une rotation.

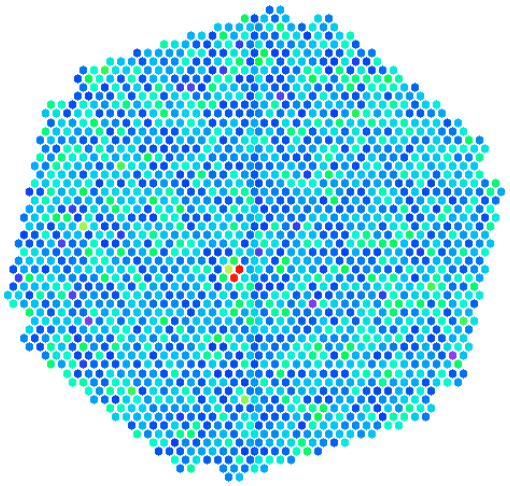


FIGURE 6.14 – Caméra LST-CAM.

6. Le rapport entre le nombre de gerbes de particules détectées en bord de caméra est directement proportionnel à rapport de surface entre le bord d'une caméra et sont intérieur.

La figure 6.15 montre comment les deux premières valeurs singulières sont modifiées lorsque la gerbe de particules est tournée dans la caméra.

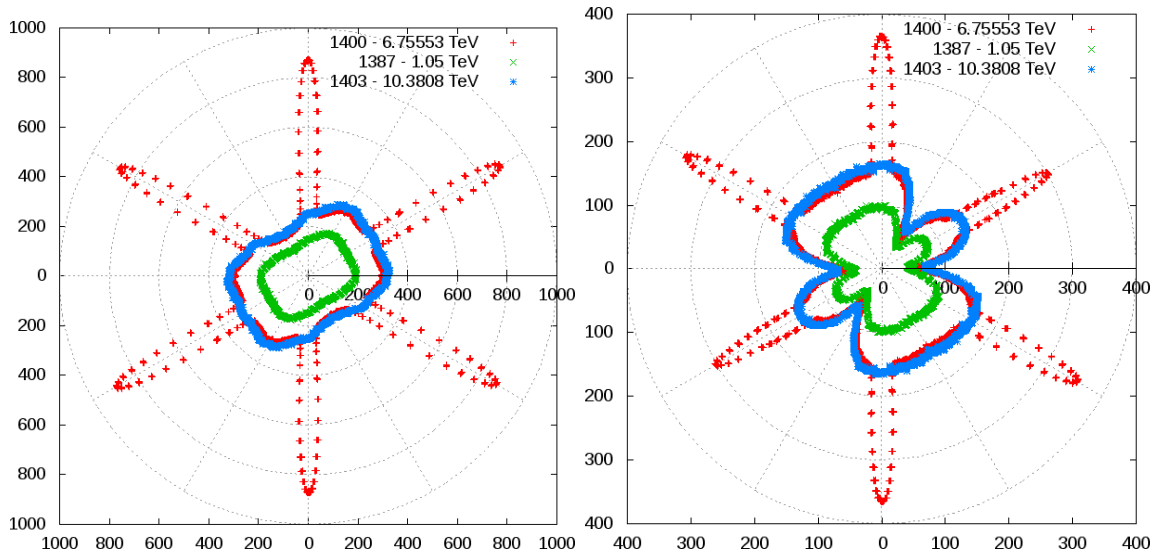


FIGURE 6.15 – Évolution des première (à gauche) et deuxième (à droite), valeurs singulières lorsque la gerbe de particules est coupée par rotation dans une LST-CAM.

Nous remarquons tout d'abord que les coupures annoncées des gerbes apparaissent nettement au niveau des encoches de la caméra.

Ensuite, ces modifications sont différentes selon l'énergie de la gerbe. Les valeurs singulières d'une gerbe à 6.75 TeV sont plus affectées par la coupure que des gerbes à 1.05 ou 10.38 TeV car sa longueur est supérieure à celle des deux autres. Donc, la longueur de la gerbe influence plus les valeurs singulières que son énergie dans ce cas.

Une gerbe de particules peut être coupée de différentes façons, en fonction de son axe de translation. Nous avons également étudié la coupure des gerbes selon une translation grâce à notre fonction d'interpolation. Plusieurs tests ont été effectués sur des axes de translation passant par le centre de la caméra. Les résultats sont similaires, nous n'en présenterons donc qu'un seul.

La figure 6.16 montre une translation du l'axe des abscisses.

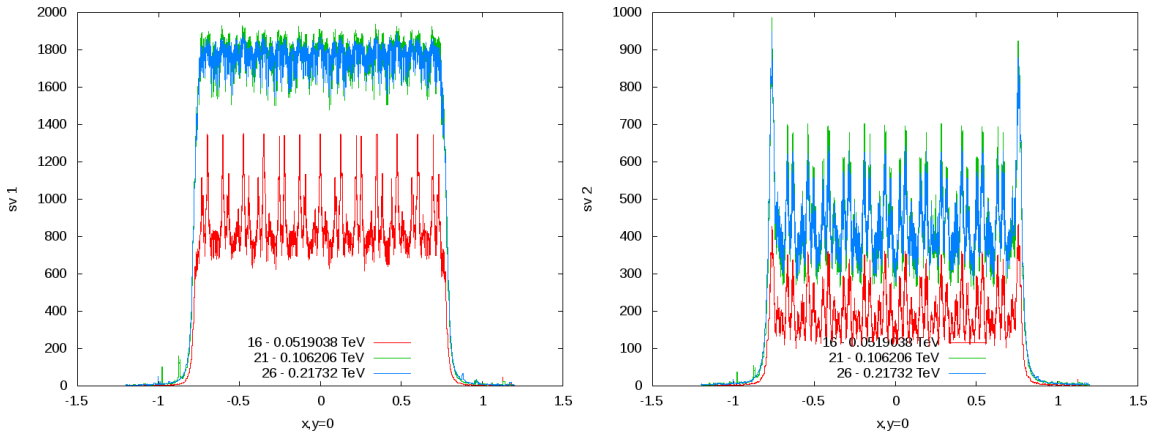


FIGURE 6.16 – Évolution des première (à gauche), deuxième (à droite) valeurs singulières lorsque la gerbe de particules est coupée par translation sur l'axe des abscisses dans une LST-CAM.

Dans ce cas, le barycentre de la gerbe est dans la caméra quand $x \in [-1, 1]$, les valeurs singulières diminuent donc sur cet intervalle et sont également influencées par la longueur de la gerbe dans la caméra. Nous attendions ce comportement car les valeurs singulières sont reliées à l'amplitude totale de l'image. Donc, plus la gerbe de particules est coupée et plus les valeurs singulières diminuent.

L'étude du comportement de la SVD sur les modèles de l'expérience H.E.S.S. a montré que la création d'un modèle SVD nécessiterait moins de comparaisons due au faible nombre de valeurs singulières impliquées dans celle-ci. En contrepartie, la sensibilité de la SVD aux translations et aux rotations des gerbes de particules dans les caméras impose un plus grand nombre de simulations qui pourra néanmoins être diminué par les symétries que nous avons mis en évidence tant sur la translation que sur la rotation des images.

Un modèle test pourrait se basé sur les acquis de l'expérience H.E.S.S. pour la définition des intervalles des paramètres physiques :

— 75 énergies
— 4 paramètres d'impact
— 12 hauteurs de première interaction

Les modèles qui décrivent les translations des gerbes dans les caméras devront exploiter la symétrie de translation qui se répète sur des intervalles de quelques pixels. Dans un deuxième temps, la prise en compte de l'acceptance de la caméra (qui dépend de la position radiale de la gerbe de particules dans la caméra) diminuera la symétrie de translation puisque les intervalles devront être définis pour que l'acceptance puisse être considérée constante.

Enfin, la rotation des gerbes de particules dans les caméras sera divisé par quatre du fait des deux symétries axiales que nous avons montré précédemment.

6.3.2.2 Étude à partir des simulations de CTA

L'étude qui utilise les modèles de H.E.S.S. nous a permis de mieux comprendre l'influence des paramètres physiques sur les valeurs SVD. Les résultats étant encourageant, nous avons poursuivi notre étude avec les simulations de CTA.

Nous avons effectué ces tests avec 1 092 fichiers de la production de simulations Monte-Carlo CTA (Prod 3b). Une reconstruction classique Hillas est appliquée ainsi que le calcul des valeurs SVD par la méthode SVD de la librairie ***Eigen***. En effet, cet algorithme n'a pas besoin d'être optimisé pour le moment car nous l'utilisons sur de petites matrices (typiquement 55×55).

Le premier test consiste à étudier la corrélation entre la première valeur singulière et l'amplitude de l'image (voir figure 6.17).

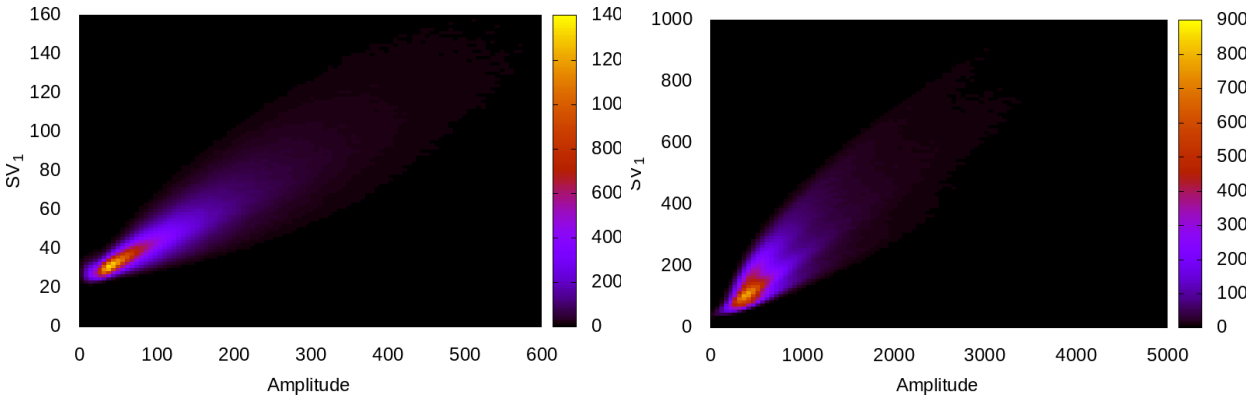


FIGURE 6.17 – Corrélation entre la première valeur SVD en fonction de l'amplitude du signal dans les caméras LST-CAM, à gauche, et NECTAR-CAM à droite.

Ces deux paramètres sont fortement corrélés malgré un élargissement notable pour une amplitude de 300 photons et une valeur de première valeur singulière à 80. On remarque également que les intervalles de variations des valeurs singulières sont moindre que dans le cas des simulations de l'expérience H.E.S.S. (voir figure 6.10).

Nous avons étudié le nombre de valeurs SVD nécessaires pour décrire un certain pourcentage de l'information décrite dans la matrice de départ (entre 10 et 95%), et sa corrélation avec l'amplitude de l'image dans la caméra ainsi que l'énergie de la particule incidente.

La proportion d'information, r_i, décrite par une valeur singulière, v_i est donnée par :

$$r_i = \frac{v_i}{\sum_{j=0}^{N} v_j} \tag{6.16}$$

Où N est le nombre total de valeurs SVD.

La figure 6.18 montre la corrélation entre le nombre de valeurs SVD nécessaires pour conserver 95% de l'information (meilleur résultat) en fonction de l'amplitude du signal dans les caméras LST-CAM et NECTAR-CAM.

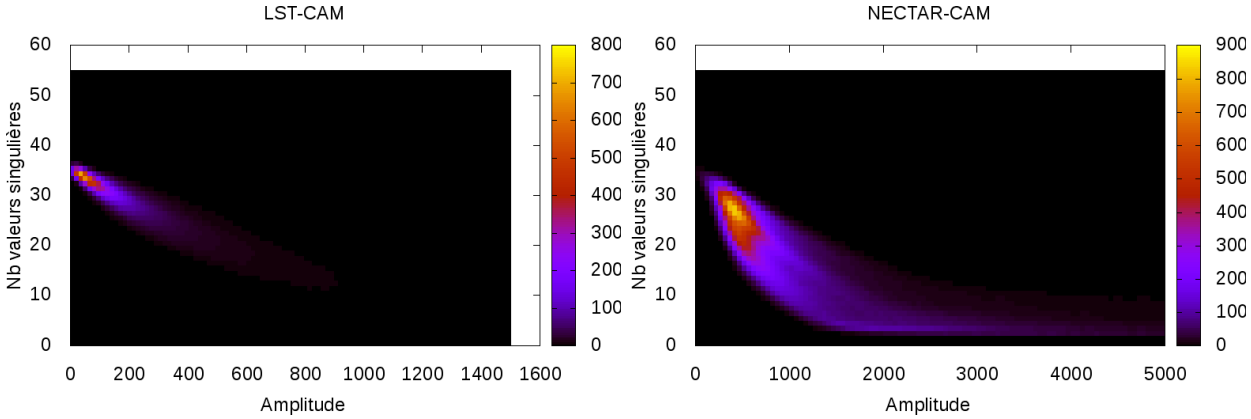


FIGURE 6.18 – Corrélation entre le nombre de valeurs SVD nécessaires pour conserver 95% de l'information en fonction de l'amplitude du signal dans les caméras LST-CAM, à gauche, et NECTAR-CAM à droite.

Plus l'amplitude de l'image est importante et moins de valeurs SVD sont nécessaires pour conserver 95% de l'information. Cette propriété est assez naturelle car la magnitude de la première valeur singulière est également corrélée à l'amplitude du signal.

La figure 6.19 montre cette corrélation avec l'énergie de la particule initiale dans les caméras LST-CAM et NECTAR-CAM.

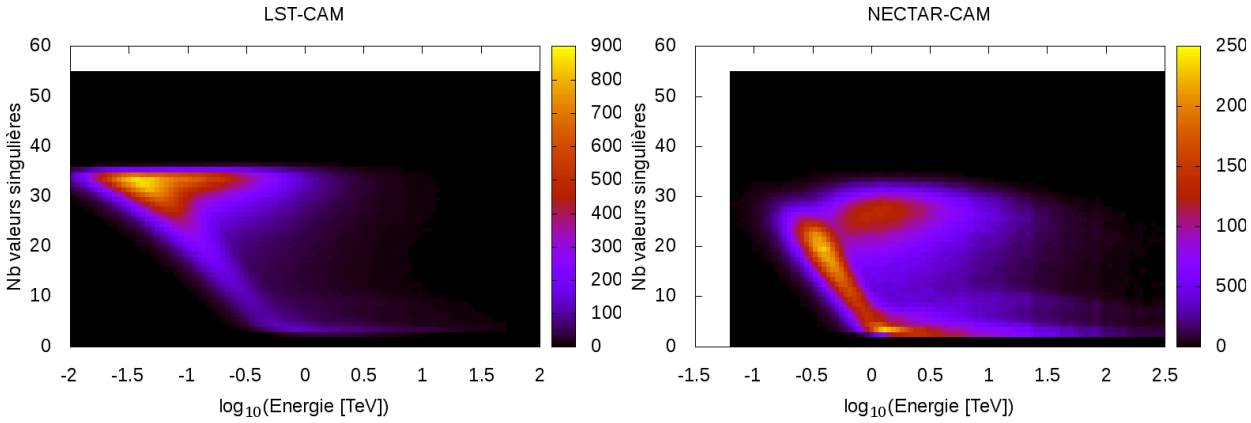


FIGURE 6.19 – Corrélation entre le nombre de valeurs SVD nécessaires pour conserver 95% de l'information en fonction de l'énergie de la particule initiale dans les caméras LST-CAM, à gauche, et NECTAR-CAM à droite.

Nous allons détailler dans ce qui suit, l'explication des différentes populations que l'on peut clairement identifier dans la figure précédente, notamment pour les NECTAR-CAM (cela reste néanmoins valable pour les LST-CAM).

La figure 6.20 définit clairement trois populations dans les événements que nous avons. Nous allons également définir une quatrième population qui est tout ce qu'il reste.

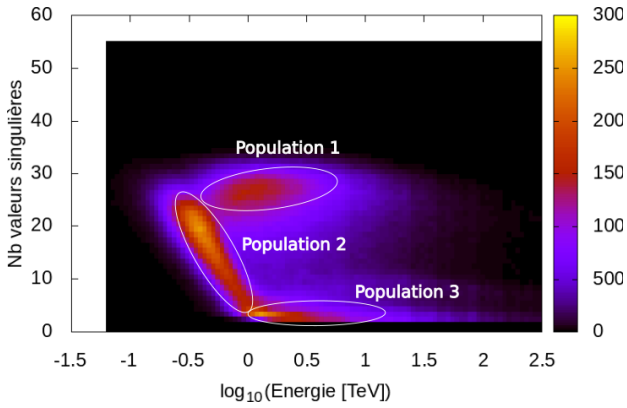


FIGURE 6.20 – Corrélation entre le nombre de valeurs SVD nécessaires pour conserver 95% de l'information en fonction de l'énergie de la particule initiale dans les caméras NECTAR-CAM à droite. Définittion des populations.

La figure 6.21 montre la séparation de ces différentes populations sur le paramètre d'impact relatif et l'amplitude de l'image dans la caméra.

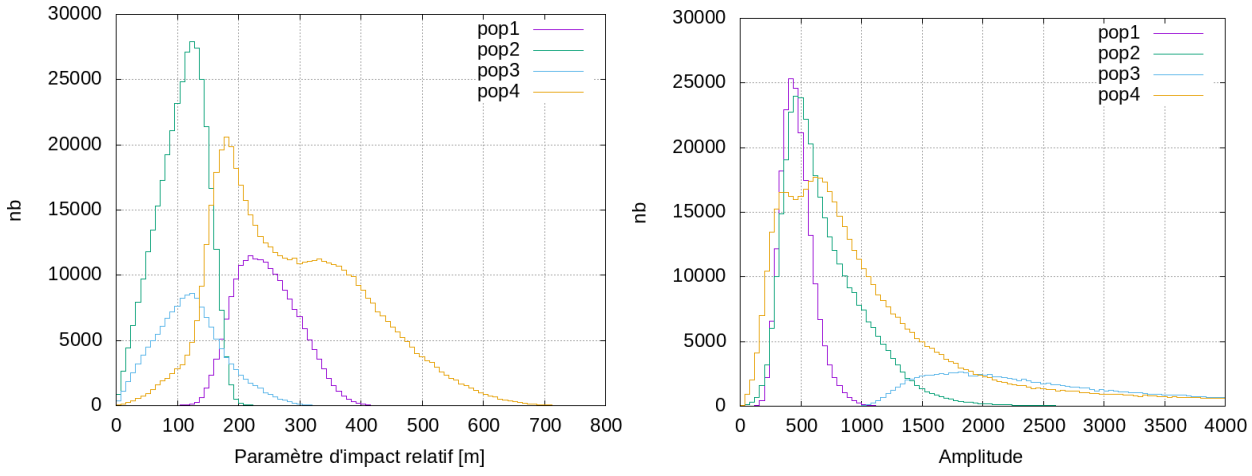


FIGURE 6.21 – **À gauche :** distribution des paramètre d'impact relatif entre les événements et le télescope donné pour les quatre populations. **À droite :** distribution des amplitudes des images dans les caméras pour les quatre populations.

On constate que les événements de la population **1** sont éloignés des caméras et ont une amplitude faible. Leurs images contiennent donc très peu de signal. De ce fait, comme le signal représente moins de 95% de l'information totale contenue dans la caméra, l'information manquante est constituée de la description du bruit par les valeurs singulières. Comme le bruit est difficile à décrire le nombre de valeurs singulières à utiliser est ainsi artificiellement augmenté. En prenant en compte uniquement la description de la gerbe de particules, la population **1** devrait se confondre avec les populations **2** et **3**.

La population **2** est composée d'événements proches du télescope et de faible amplitude. Dans ces conditions, plus l'énergie est faible et plus le spectre SVD décrit de bruit (comme pour la population **1**).

La population **3** est opposée au deux précédentes car elle est composée d'événements proches du télescope mais très énergétiques. De ce fait, la quasi-totalité de l'information à décrire concerne la gerbe de particules. Comme leurs énergies sont importantes (supérieures à 1 TeV) leur description nécessite peu de valeurs singulières.

6.4 Niveaux de parallélisme

L'analyse de données a de nombreux niveaux de parallélisme grâce à l'indépendance des données du fait de la nature stochastique des événements. La figure 6.22 montre les différents niveaux de parallélismes de cette analyse. Les observations sont indépendantes les unes des autres. Dans la première partie de l'analyse les télescopes sont également indépendants et leur images également. La reconstitution des événements nécessite un tri de donnée, de paramètres par télescopes en paramètres par événements (voir section 5.6.1). Cependant, il peut être amélioré en utilisant des fichiers de télescopes qui contiennent un intervalle fixe des identifiants d'événements, ce qui limite le transfert de données pendant le tri.

Ainsi, l'analyse exploite tous ces niveaux de parallélisme de sorte à être massivement parallèle. Même la reconstitution des événements peut utiliser certains d'entre eux sauf la vectorisation car nous utilisons un algorithme de tri par arbre. Toutes ces propriétés garantissent une analyse rapide et multi-échelle.

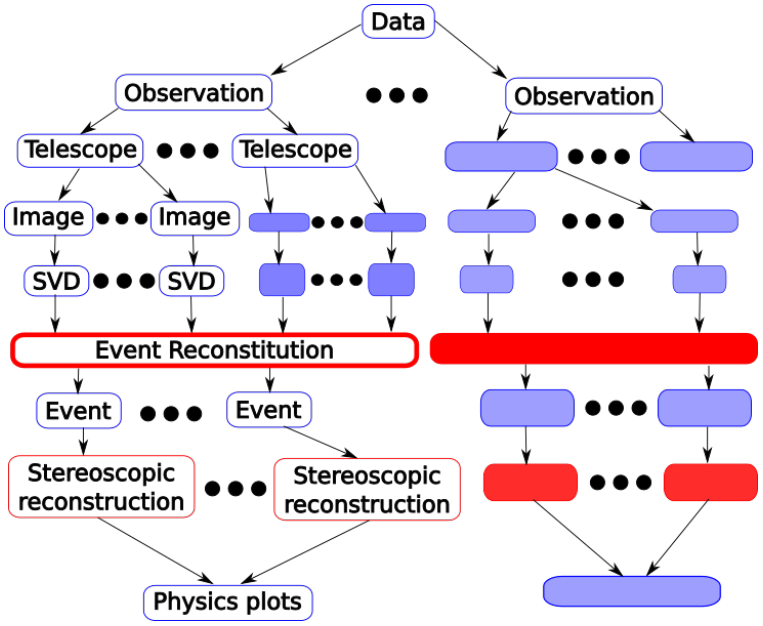


FIGURE 6.22 – Niveaux de parallélismes dans l'analyse SVD.

6.5 Algorithme utilisant des spectres de valeurs singulières moyennées

La première reconstruction SVD est inspirée de l'approche classique de *Model++* (voir section 6.1). Dans cette méthode, les événements sont comparés à des images moyennées.

Dans cette section, nous faisons de même avec les spectres de valeurs singulières. Ici, chaque moyenne de spectre correspond à une configuration de paramètres physiques.

6.5.1 Préparer l'algorithme de reconstruction

La section 6.3.2.1 montre que les valeurs singulières augmentent lorsque les gerbes de particules sont coupées au bord des caméras. De plus, une reconstruction SVD qui n'utilise l'énergie, la hauteur de première interaction et le paramètre d'impact des événements, comme *Model++*, n'est pas aussi efficace que prévu. Cela implique que les spectres de valeurs singulières qui correspondent à des gerbes coupées ne doivent pas être moyennés avec ceux obtenus par des gerbes contenues dans la caméra car la moyenne obtenue est alors entre ces deux possibilités. Le résultat est donc irrémédiablement biaisé.

Un paramètre $\alpha = \sqrt{g_x^2 + g_y^2}$ est ajouté dans le modèle SVD pour distinguer les gerbes coupées dans les caméras. Où (g_x, g_y) est le barycentre de la gerbe dans la caméra (voir section 5.5 ou annexe A). Avec ce paramètre, les gerbes en bord de caméra ne sont pas moyennées avec les autres.

Une différence majeure est que *Model++* utilise un paramètre d'impact absolu, ce qui permet de résoudre partiellement le problème d'asymétrie du développement de la gerbe dû au champ magnétique de la Terre.

En effet, lorsqu'une gerbe de particules se déploie dans l'atmosphère les particules chargés qui la composent (électrons/positrons pour les gerbes produites par des rayons gamma) interagissent avec le champ magnétique terrestre. Ce dernier les dévie légèrement de leur trajectoire, mais cette déviation se ressent car la gerbe de déploie sur une quinzaine de kilomètres.

Cependant, CTA a beaucoup plus de télescopes. Par conséquent, un modèle SVD absolu serait fortement statistiquement dégradé. Nous avons donc ajouté l'angle entre la position du paramètre d'impact et la position du télescope courant (θ) pour lever la dégénérescence.

Ce paramètre θ est défini par :

$$\theta = \arctan\left(y_{core} - y_{tel}, x_{core} - x_{tel}\right) \tag{6.17}$$

Où, (x_{core}, y_{core}) est la position absolue du paramètre d'impact de la gerbe de particules au sol et (x_{tel}, y_{tel}) la position du télescope courant.

Il permet théoriquement une meilleure reconstruction de la hauteur de première interaction tout en évitant un modèle absolu. La statistique disponible est donc moins réduite. Cependant, ce paramètre ne devra pas être trop découpé pour assurer que les moyennes ont un sens.

6.5.2 Création d'un espace des modèles SVD

L'espace des modèles est créé en moyennant les spectres de valeurs singulières qui correspondent à des paramètres physiques proches.

Le modèle est découpé selon les paramètres suivants :

— α : distance entre le barycentre de la gerbe (g_x, g_y) et le centre de la caméra.
— ρ : paramètre d'impact relatif (distance entre la position du paramètre d'impact absolu et le télescope courant).
— θ : angle entre le paramètre d'impact et le télescope courant (défini dans la section 6.5.1).

Les écarts types associés aux moyennes sont également calculés.

Chaque caméra à son propre modèle SVD.

6.5.3 Conception

La figure 6.23 illustre la structure des données dans un modèle SVD moyenné.

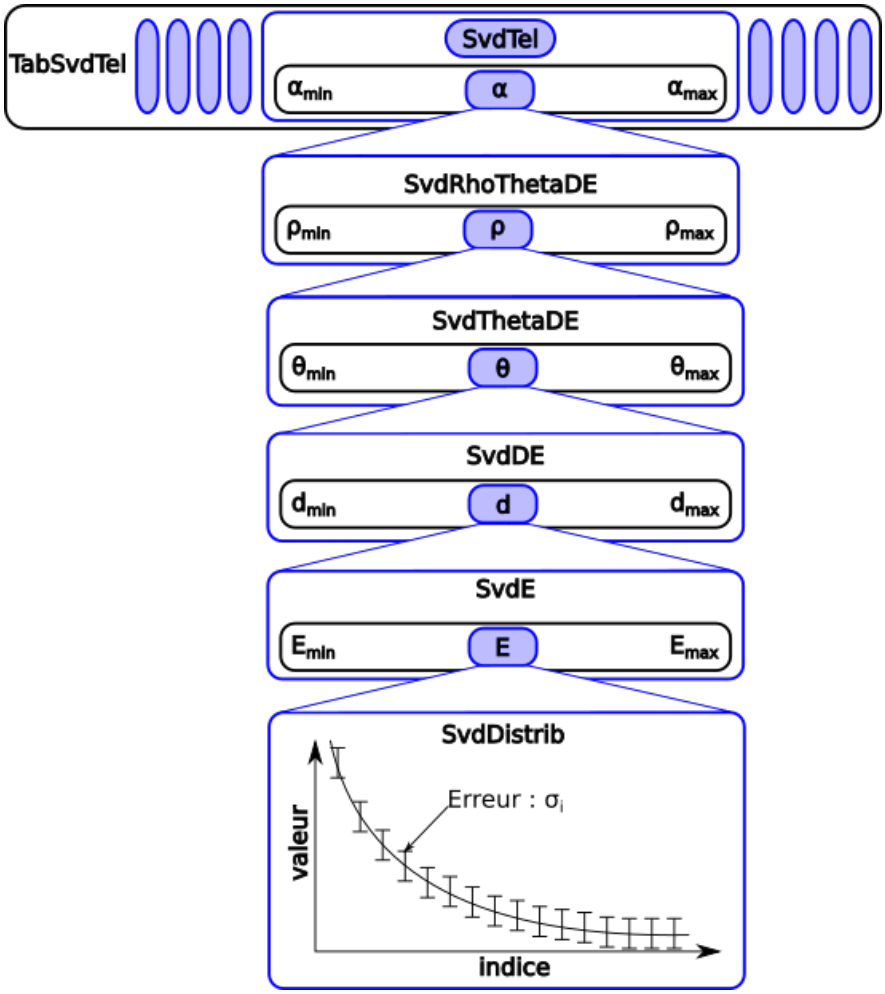


FIGURE 6.23 – Conception du modèle SVD moyen.

Ce format de données a été créé à l'aide des générateurs décrits dans le chapitre 3.

Tout d'abord, la classe principale ***TabSvdTel*** décrit l'ensemble des types de caméras disponibles *via* un tableau. Nous l'avons utilisé dans le but de séparer les différents types de caméras mais il serait également possible de séparer tous les télescopes et ainsi créer un modèle absolu à la manière de *Model++*[7].

Chaque modèle de caméra est décrit par la classe ***SvdTel***. Celle-ci contient un tableau qui sépare les modèles pour différentes valeurs de $\alpha = \sqrt{g_x^2 + g_y^2}$, où (g_x, g_y) est la position du barycentre de la gerbe de particules dans la caméra. Il permet de séparer les spectres obtenus avec des gerbes coupées et ainsi d'améliorer la pertinence du calcul de la moyenne.

7. Mais nous ne disposons pas de suffisamment de simulations pour l'instant.

La classe ***SvdRhoThetaDE*** sépare à nouveau les modèles sur le paramètre d'impact relatif ρ[8]. Ce paramètre est également utilisé dans *Model++*.

Ensuite, les modèles sont séparés sur la variable θ par la classe ***SvdThetaDE***. Elle décrit l'angle entre le paramètre d'impact absolu et la position du télescope courant par rapport à l'axe des abscisse en direction du nord.

Toutes les variables précédentes sont préalablement déterminées. Les variables de séparation suivantes seront systématiquement testées.

La classe ***SvdDE*** sépare encore les modèles sur la hauteur de première interaction, d, de la particule initiale. Et la classe ***SvdE*** finalise la séparation sur le logarithme de l'énergie E de la particule initiale.

Finalement, la classe ***SvdDistrib***, décrit une distribution moyenne de valeurs singulières pour le jeu de paramètres $(\alpha, \rho, \theta, d, E)$ ainsi que l'erreur σ_i associée à chaque valeur singulière moyenne. Cette erreur est l'écart type de la distribution des valeurs singulières qui a permis de calculer la moyenne, corrigée avec le coefficient de Student dans le cas où le nombre de valeurs utilisées pour le calcul est faible (inférieur à 100).

Toutes les variables décrites précédemment ont des bornes minimales et maximales associées. Le programme qui créé les modèles à partir des simulations Monte-Carlo permet à l'utilisateur de décider lui-même des bornes à prendre en compte ou les détermine automatiquement par défaut en fonction des simulations utilisées.

6.5.4 Utilisation de l'espace des modèles SVD

Dans cette section, les paramètres α, ρ, θ du modèle sont déterminés par les simulations de gerbes de particules. Lors de l'étape de reconstruction ces paramètres sont reconstruits à partir des images et leurs valeurs déterminent le modèle à choisir pour la reconstruction. Cependant, ces paramètres reconstruits peuvent être biaisés à cause de la précision intrinsèque de la reconstruction.

6.5.4.1 Comparaison des spectres de valeurs singulières

Les spectres et leurs écarts types associés sont choisis dans le modèle associé à la caméra du télescope courant en fonction des paramètres α, ρ, θ. L'ensemble des modèles qu'il reste à évaluer est définis en fonction de E et d (qui sont respectivement l'énergie et la hauteur de première interaction).

Ensuite, un χ^2 est calculé pour comparer les modèles aux spectres de l'événement à analyser. Cela produit une matrice de χ^2 pour chaque couple (E, d) et pour chaque télescope de l'événement.

8. Distance entre le point d'impact absolu de la gerbe au sol et le télescope courant.

Finalement, le χ^2 stéréoscopique est obtenu en sommant les matrices de χ^2 de chaque télescope. L'énergie (E) et la hauteur de première interaction (d) sont déterminés par la configuration physique du χ^2 minimal.

6.5.4.2 Détermination du paramètre d'impact

Comme nous l'avons mentionné dans la section précédente, le paramètre d'impact et le paramètre θ sont reconstruits avec une incertitude qui dépend de l'événement.

Il arrive parfois que le paramètre d'impact ne soit pas déterminé, et ne puisse pas être utilisé comme paramètre d'entrée du modèle. Dans ce cas, tous les paramètres d'impact possibles sont testés pendant la reconstruction avec une précision (d_x, d_y).

Ensuite, l'algorithme est le même qu'à la section 6.5.4.1, mais les paramètres finaux ont été déterminés avec un χ^2 minimal sur l'ensemble des paramètres d'impact possible.

6.5.5 Résultats

L'analyse traite 350 GO en 20 ou 40 minutes sur huit threads selon la machine utilisée dans le centre de calcul MUST [9].

La figure 6.24 montre la résolution en énergie d'un modèle SVD moyen pour le site nord.

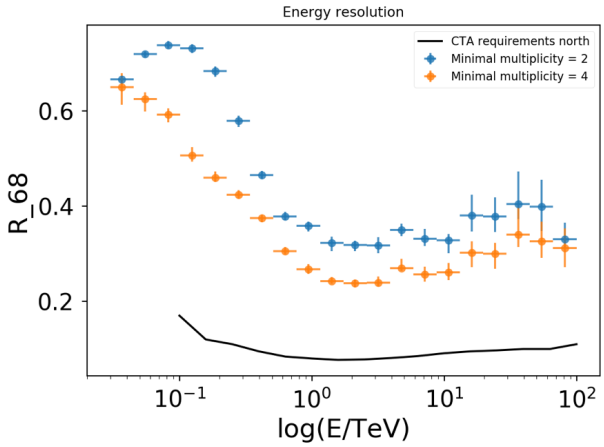


FIGURE 6.24 – Résolution en énergie avec deux version du modèle SVD moyen.

Cette méthode utilise 4 intervalles sur le paramètre α, 4 en *depth*, 20 en paramètre d'impact relatif, 70 en énergie et 12 valeurs singulières. Il a été construit avec 2 192 fichiers de simulations de sources ponctuelles de rayons gamma, ce qui offre une meilleure statistique que pour le modèle précédent. Nous avons donc pu diviser d'avantage les intervalles en profondeur de première interaction (X_max), en paramètre d'impact et en énergie.

Ce modèle est le premier à atteindre une résolution inférieure à 40% au delà de 1 TeV. Bien que cela soit très différent des performance de physique atteinte par l'analyse de données

9. Centre de calcul du Laboraroire d'Annecy de Physique des Particules, voir section 1.5

de l'expérience MAGIC, MARS, elle nous a permis de mieux appréhender l'importance des différents paramètres.

Malheureusement, il y a un manque de configurations pour les moyennes et les écarts types soient tous exploitables. Tant qu'il n'y aura pas plus de spectres pour créer les points du modèle, il ne pourra pas mieux reconstruire l'énergie.

6.6 Algorithme basé sur un dictionnaire de spectre de valeurs singulières

Cette section montre comment créer un modèle de spectres de valeurs singulières et l'utiliser comme un dictionnaire.

La section précédente montre que moyenner des spectres de valeurs singulières dégrade la statistique disponible ce qui rend certaines parties inutilisables. L'ajout d'un paramètre supplémentaire θ lève un peu la dégénérescence entre l'énergie et la hauteur de première interaction. Il permet d'améliorer globalement la résolution en énergie mais en dégradant la statistique disponible.

La multiplication des paramètres pour améliorer la qualité des moyennes de spectres diminue la statistique globale dans chaque point du modèle et finalement dégrade quand même sa résolution.

De plus, il nous est apparu *a posteriori* que la hauteur de première interaction était moins pertinente que le paramètre X_{max} qui décrit la quantité d'atmosphère traversée par la particule initiale avant son interaction avec l'atmosphère (voir figure 6.25).

L'utilisation d'un dictionnaire permet de conserver la statistique disponible tout en séparant les gerbes coupées des autres. Mais avec cette approche, les spectres sont comparés à d'autres spectres qui ne sont pas moyennés, en conséquence les fluctuations pourraient augmenter.

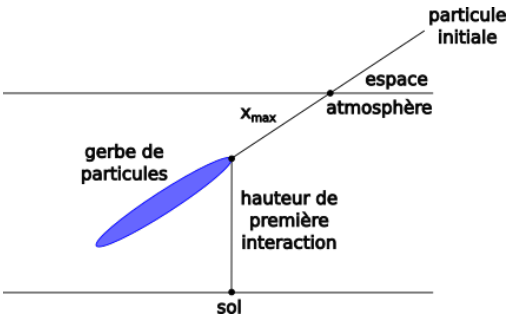


FIGURE 6.25 – Pour une gerbe de particules, le X_{max} correspond à la quantité d'atmosphère traversée par la particule incidente avant son interaction avec l'atmosphère.

6.6.1 Création d'un dictionnaire de valeurs singulières

La figure 6.26 montre la structure des données utilisée pour le dictionnaire de valeurs singulières.

La classe ***TabDicoTel*** décrit l'ensemble des types de caméras utilisables pour la reconstruction. De la même manière que précédemment, elle pourrait également être utilisée pour tous les télescopes et ainsi décrire un dictionnaire de modèles absolu.

Un télescope est décrit par la classe ***DicoTel***. Elle contient une matrice qui trie les différents dictionnaires par paramètre d'impact relatif, ρ, et par l'angle antre le paramètre d'impact absolu et ce télescope, θ. Ces paramètres varient respectivement dans $[\rho_{min}, \rho_{min}]$ et $[\theta_{min}, \theta_{min}]$. Nous avons choisi une approche matricielle car elle simplifie l'arborescence des modèles à utiliser.

Chaque dictionnaire est décrit par une classe ***DicoSv*** qui contient également une matrice. Chaque ligne de cette matrice est une entrée du dictionnaire. Les m premières colonnes sont les m premières valeurs propres des images des simulations Monte-Carlo, et sont suivies par les paramètres physiques $(\rho, \theta, E, X_{max})$, respectivement, le paramètre d'impact relatif, l'angle θ évoqué précédemment, l'énergie de la particule incidente (en TeV) et la profondeur de première interaction, X_{max}, (voir figure 6.25).

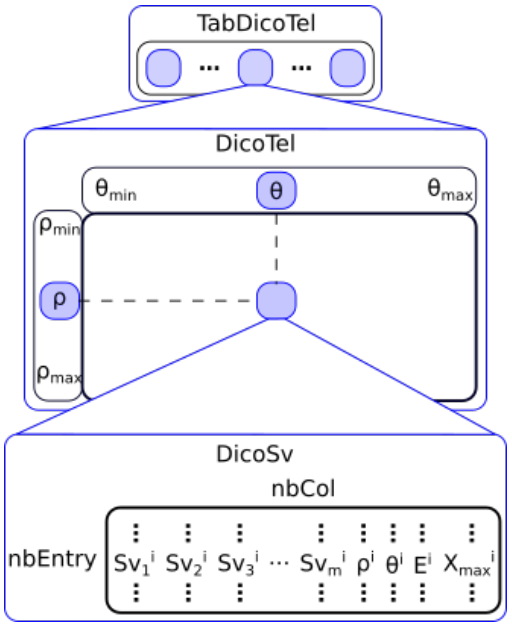


FIGURE 6.26 – Structure de données du dictionnaire de valeurs singulières.

L'utilisation d'un dictionnaire sépare naturellement les différents spectres, le paramètre α de la section précédente n'est donc pas utile car les spectres associés aux gerbes coupées seront éloignés des autres.

6.6.2 Reconstruction de l'énergie de la particule incidente

Nous avions commencé par utiliser le même découpage d'intervalle que dans la section précédente. Les résultats étaient légerement meilleurs, mais au alentour de 30% ce qui ne suffisait toujours pas.

Nous n'utilisons donc qu'un seul dictionnaire. Celui-ci regroupe toutes les valeurs possibles du paramètre d'impact et de l'angle θ.

Contrairement à ce que nous faisions précédemment, nous réalisons le tri sur les paramètres ρ et θ *a posteriori*.

Cette méthode de reconstruction évalue la pertinence d'un spectre du modèle en deux étapes :

— Le meilleur modèle est trouvé par dichotomie. Cela permet de le trouver très rapidement même si le dictionnaire contient de nombreuses entrées.

— L'exploration des modèles qui sont compatibles autour du meilleur en fonction d'une tolérance δ_ρ et δ_θ respectivement sur les paramètres ρ et θ.

La figure 6.27 montre l'évolution du χ^2 entre les différents modèles par rapport à un modèle choisi (ici le 1 078 345) et à l'indice des autres modèles. Nous observons très clairement une vallée, dessinée par de fortes oscillations.

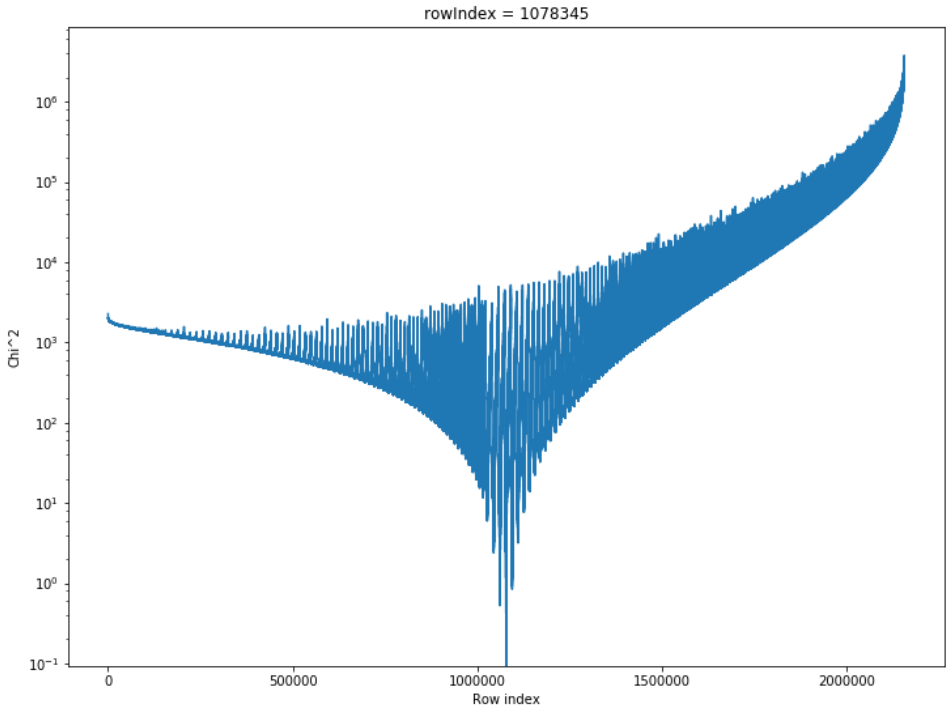


FIGURE 6.27 – Valeur du χ^2 entre les différents modèles d'un même dictionnaires en fonction de l'indice du modèle.

6.6.2.1 Validation de la méthode

Nous avons tout d'abord testé la méthode de reconstruction par dictionnaire de valeurs SVD sur le même lot de données que celui utilisé pour créer ce dictionnaire. Cela nous a permis de vérifier la pertinence de la méthode dans un cadre très simplifié.

Dans ce cas, nous avons conserver les 7 premières valeurs singulières dans le dictionnaire.

La figure 6.28 montre le résultat de ce test.

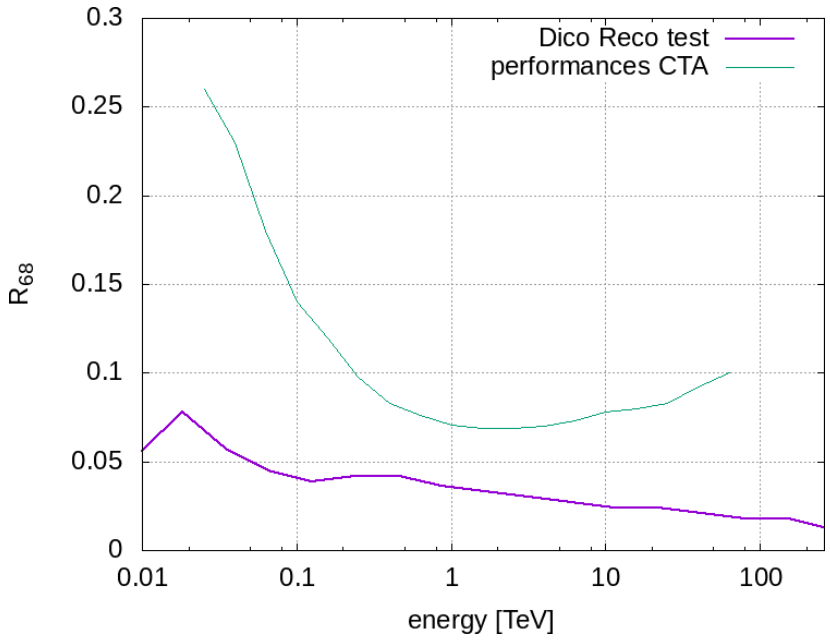


FIGURE 6.28 – Premier test de la reconstruction SVD par dictionnaire. La résolution en énergie en fonction de l'énergie simulée (violet gras) comparée à titre indicatif aux performances de CTA (vert fin).

On remarque que la résolution en énergie est améliorée au fur et à mesure que l'énergie augmente. Cela est du au fait que nous n'utilisons que les 7 premières valeurs SVD. En effet, l'utilisation de peu de valeurs ne permet de rendre compte de l'image de départ que si l'énergie de la particule est suffisante.

Comme cette première phase de test est cohérente, nous l'avons testée dans des conditions plus réalistes.

6.6.2.2 Résulats préliminaires

La figure 6.29 montre la résolution en énergie de la méthode de reconstruction SVD par dictionnaire dans sa version la plus simple. Ici, nous utilisons uniquement les valeurs singulières pour déterminer l'énergie de la particule initiale. Cela explique l'éloignement du résultat par rapport à l'objectif de CTA, bien qu'il soit meilleur que le précédent.

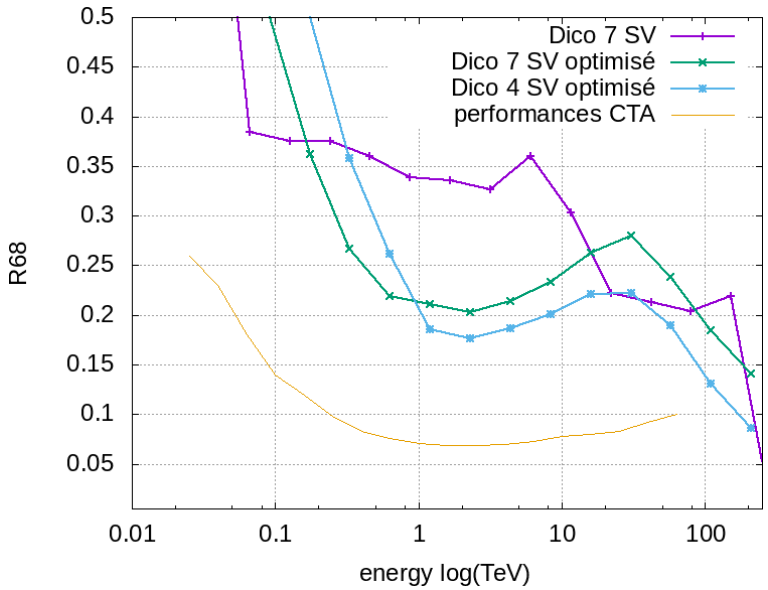


FIGURE 6.29 – Première résolution en énergie de la méthode par dictionnaire SVD. La résolution avec 7 valeurs SVD sans optimisation est en violet (+), avec optimisation par essaim en vert (×) et avec 4 valeurs SVD en bleu (■) comparées au performances de CTA en trait fin orange.

Les résultats sont éloignés des performances actuelles des reconstructions de CTA, et également de celle que nous avons développée et présenté au chapitre 5. Malgré cela, l'optimisation physique de cette méthode basée sur un algorithme d'essaim particulaire (aussi appelé algorithme en essaim d'abeilles) montre qu'un jeu de coupures relativement simple permet d'améliorer sensiblement ces résultats :

— La tolérance relative de différence entre deux valeurs singulière de même indice.
— La tolérance absolue sur le paramètre d'impact reconstruit.
— Une coupure sur la cohésion des énergies données par le dictionnaire.
— Le nombre de configurations à tester autour de la meilleure trouvée.
— Le nombre d'entrées minimale acceptées par les coupures précédentes pour valider la reconstruction.

Ces coupures ne changent en aucun cas les résultats classiques que l'on obtient avec les paramètres Hillas. Les résultats SVD ont été obtenus avec les spectres SVD et le paramètre d'impact reconstruit par la méthode Hillas. Aucun autre paramètre n'a été utilisé, cela implique que les valeurs SVD contiennent l'information de l'énergie mais que la stéréoscopie et l'exploitation des valeurs SVD peuvent encore être améliorée.

La cause principale de la dégradation du résultat vient du fait que les quantités physiques telles que l'énergie et la hauteur de première interaction sont dégénérées. C'est à dire que différentes combinaisons de valeurs produisent des spectres de valeurs singulières semblables.

En effet, nous avons vu dans la section 6.3.2 que les spectres de valeurs singulières ne sont que très peu influencés par la position de la gerbe dans la caméra. La dégénérescence n'est donc pas levée par l'utilisation des spectres seuls.

6.6.2.3 Prise en compte de la profondeur de première interaction

La profondeur de première interaction, X_{max}, décrit la quantité d'atmosphère traversée par la particule incidente avant interaction.

La figure 6.30 montre la corrélation entre ce paramètre, X_{max} et l'énergie de la particule incidente.

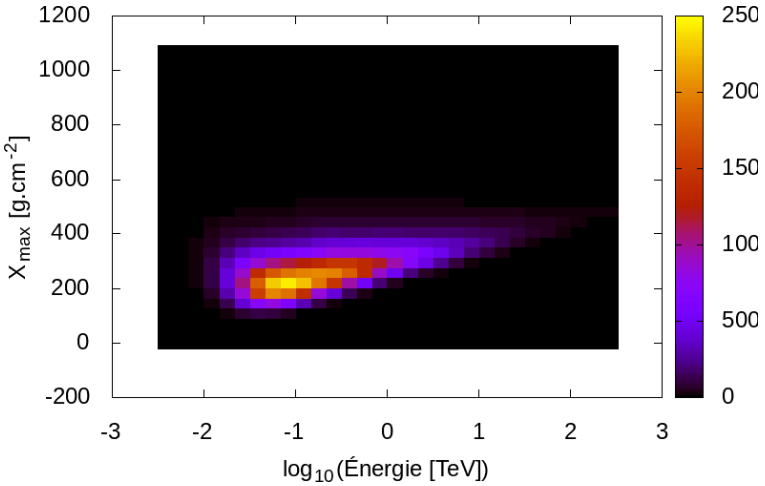


FIGURE 6.30 – Corrélation entre le X_{max} et l'énergie de la particule incidente.

Cette corrélation n'est pas totale, mais permet d'évaluer une borne inférieure de l'énergie au même titre que les paramètres précédents.

Son calcul nécessite de reconstruire la hauteur de première interaction où une quantité équivalente comme la hauteur où se situe le maximum d'émission de la gerbe de particules.

Pour cela, nous avons adapté l'algorithme d'intersection de droites à deux dimensions (section 5.6.2.4) à trois dimensions (voir annexe D).

La figure 6.31 montre que la distribution des hauteurs de première interactions est proche ce celle simulée. Nous notons néanmoins un excès d'événements aux alentours de 50 000 m. Cela montre que la méthode de reconstruction de la hauteur de première interaction est valide.

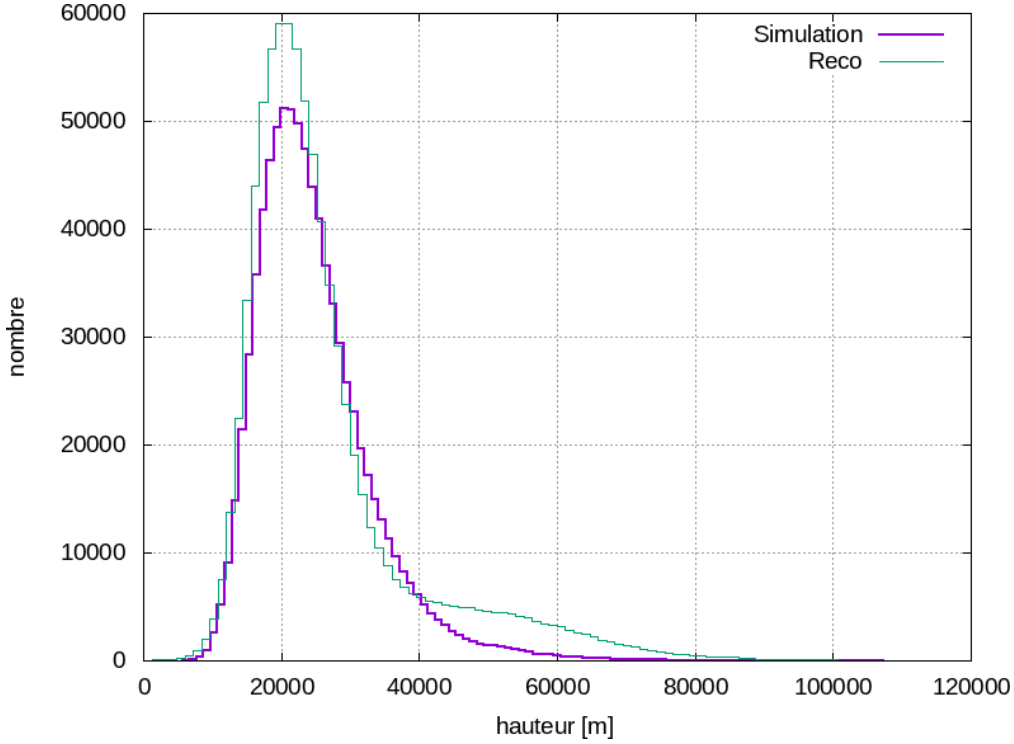


FIGURE 6.31 – Distribution des différentes hauteurs de première interactions reconctruites (vert fin) comparée à celle simulée (violet gras).

La figure 6.32 montre la résolution de la hauteur de première interaction en fonction de l'énergie de la particule incidente. Sans surprise, elle est fortement dégradée à basse énergie à cause de la taille réduite des ellipses dans les caméras.

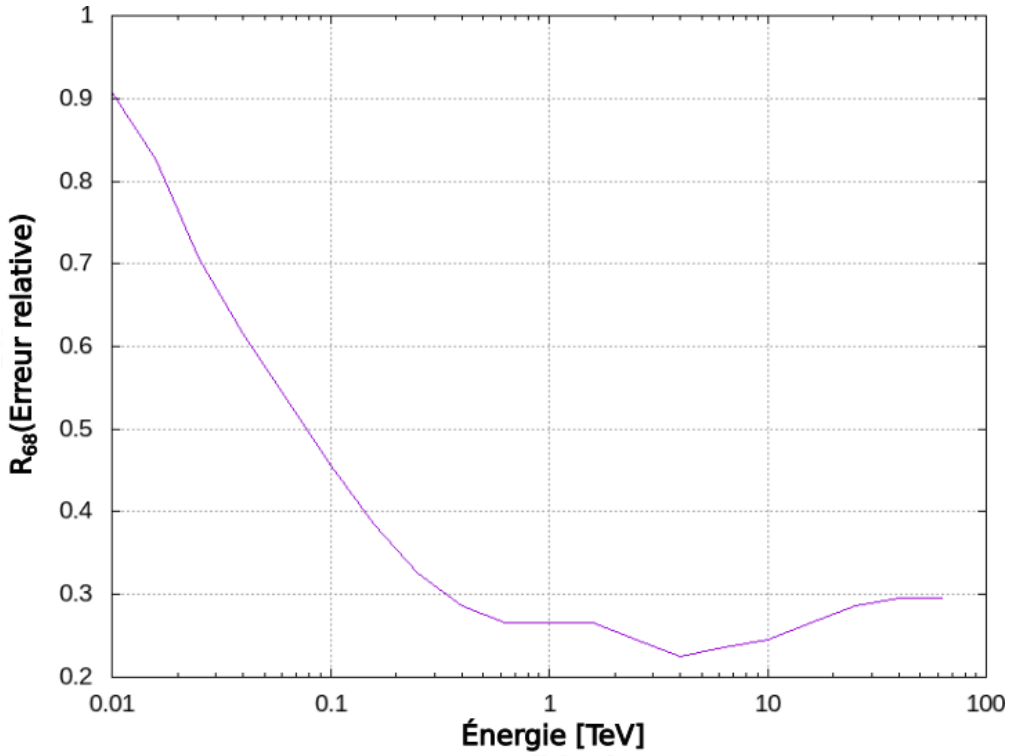


FIGURE 6.32 – Résolution de la reconstruction de la hauteur de première interaction en fonction de l'énergie de la particule incidente.

Comme la reconstruction de la hauteur de première interaction fonctionne, nous avons tester le dictionnaire en ajoutant le paramètre X_{max} simulé pour vérifier sont utilité. La figure 6.33 montre les résulats obtenus avec ce nouveau paramètre, après une optimisation par essaim d'abeilles.

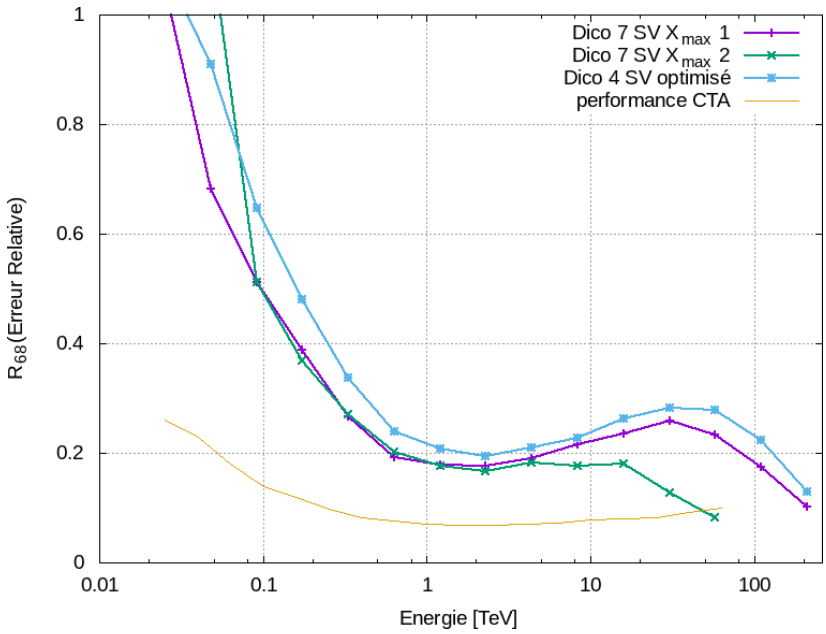


FIGURE 6.33 – La résolution en énergie en fonction de l'énergie simulée avec 7 valeurs SVD et le X_{max} (violet gras, + et vert gras, ×) comparée au modèle précédent qui utilise 4 valeurs SVD (vert gras, ■) et aux performances de CTA (orange fin).

Les différentes tolérances obtenues sont les suivantes :

	Configuration 1	Configuration 2
Tolérance valeurs SVD	66.6518%	167.435%
Tolérance paramètre d'impact	53.7787 m	58.8017 m
Tolérance X_{max}	99.2479%	72.6522%
Étalement en énergie	43.0361%	6.3164%
Nombre de configuration à vérifier	8547	8583
Nombre d'entrée minimales	50	35

On remarque immédiatement que la tolérance sur le paramètre X_{max} de la première configuration est trop permissive. La faible amélioration des résultats tend à montrer que ce paramètre est finalement moins utile que prévu.

La deuxième configuration améliore sensiblement la résolution à haute énergie. On note que la tolérance sur les valeurs SVD est beaucoup plus lâche, tandis que celle sur l'énergie est bien plus faible (quelque pourcents contre quelques dixaines de pourcents pour la configuration 1).

Malgré le fait que la seconde configuration semble montrer le bon chemin vers l'optimisation de la reconstruction en énergie avec uniquement les valeurs SVD et bien que ces différentes méthodes de reconstruction en énergie nous ont permis de mieux appréhender

l'utilisation des valeurs singulières, l'investissement en temps nécessaire à cette optimisation complète est au delà d'une thèse en informatique. Nous avons donc testé une implémentation qui utilise des arbres de décision.

6.7 Reconstruction en énergie par arbres de décision

Les méthodes très simples que nous avons testées montrent que les valeurs singulières seules apportent des informations sur l'énergie de la particule incidente.

La figure 6.34 montre la résolution en énergie que nous obtenons en combinant les paramètres Hillas aux valeurs singulières dans la même architecture basée sur des arbres de décision que nous avons préalablement présentée dans la section 5.6.4.2.

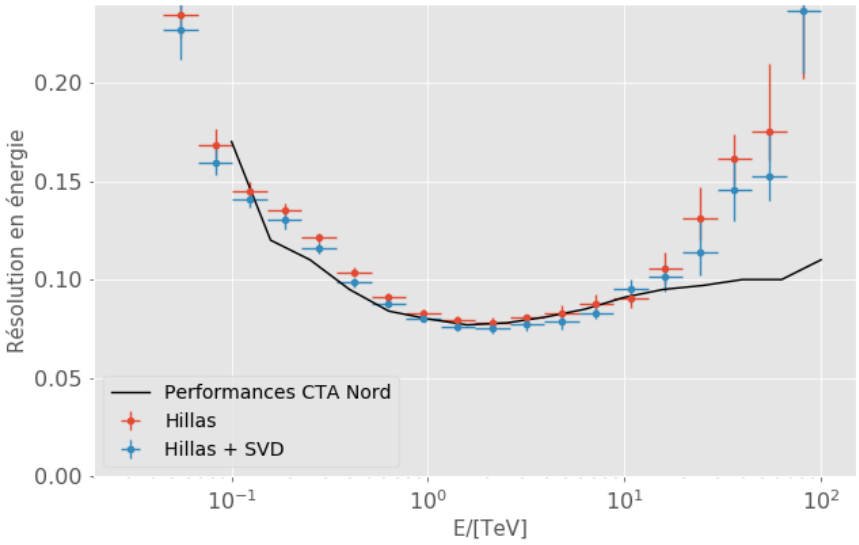


FIGURE 6.34 – La résolution en énergie en fonction de l'énergie simulée avec les paramètres Hillas, points rouges, avec les paramètres Hillas et les valeurs singulières, points bleus comparées aux performances de CTA.

Les résultats sont en accord avec les performances de CTA jusqu'à 10 TeV. Comme précédemment, une coupure pourrait être appliquée sur les hautes énergies pour obtenir un accord parfait.

Nous observons que les valeurs singulières améliorent globalement la résolution en énergie, et d'autant plus à plus haute énergie.

Les résultats étant de bonne qualité, nous nous somme consacrés à la discrimination qui est l'apport le plus important de l'analyse *Model++*.

6.8 Discrimination avec SVD

Les valeurs SVD décrivent les images des télescopes avec un niveau de détail arbitraire qui dépend du nombre de valeurs sélectionnées. Comme ce niveau de détail est supérieur à celui fournit par la méthode Hillas, nous pouvons espérer une amélioration de la discrimination.

La discrimination s'effectue de manière similaire à la section 5.6.5. Nous utilisons deux arbres de décisions (LST-CAM et NECTAR-CAM) avec les paramètres Hillas et les spectres SVD pour chacun des télescopes des événements. Ces paramètres deviennent les entrées d'un arbre de décision.

Le nombre de fichiers utilisés pour les différentes phases d'entraînement et de tests sont décrit par le tableau 6.1.

Simulations	Entraînement	Test
Gamma ponctuels	2 192	272
Gamma diffus	12 240	1 360
Protons diffus	6 810	790

TABLE 6.1 – Nombre de fichiers utilisés pour les toutes les phases d'entraînement et de test de nos analyses de données.

6.8.1 Discrimination sur une source ponctuelle

La discrimination a pour but de séparer les événements produits par des rayons gamma de ceux produits par des protons (principalement). Ces derniers représentent du bruit qu'il faut soustraire pour obtenir des cartes du ciels pertinentes.

L'utilisation d'arbres de décision fournit des outils comme les courbes ROC (Receiver Operating Characteristic) qui permettent d'évaluer la meilleure coupure sur la probabilité que renvoient les arbres de décision. Cette courbe donne la proportion des événements vrais positifs (gamma) et celui des faux positifs (protons). Le but est donc de maximiser le rapport signal sur bruit tout en conservant le plus de rayons gamma possibles. La figure 6.35 montre les différentes courbes ROC que nous obtenons avec différents paramètres d'entrée.

Toutes les configurations utilisent les paramètres stéréoscopiques qui sont : le paramètre d'impact et la multiplicité.

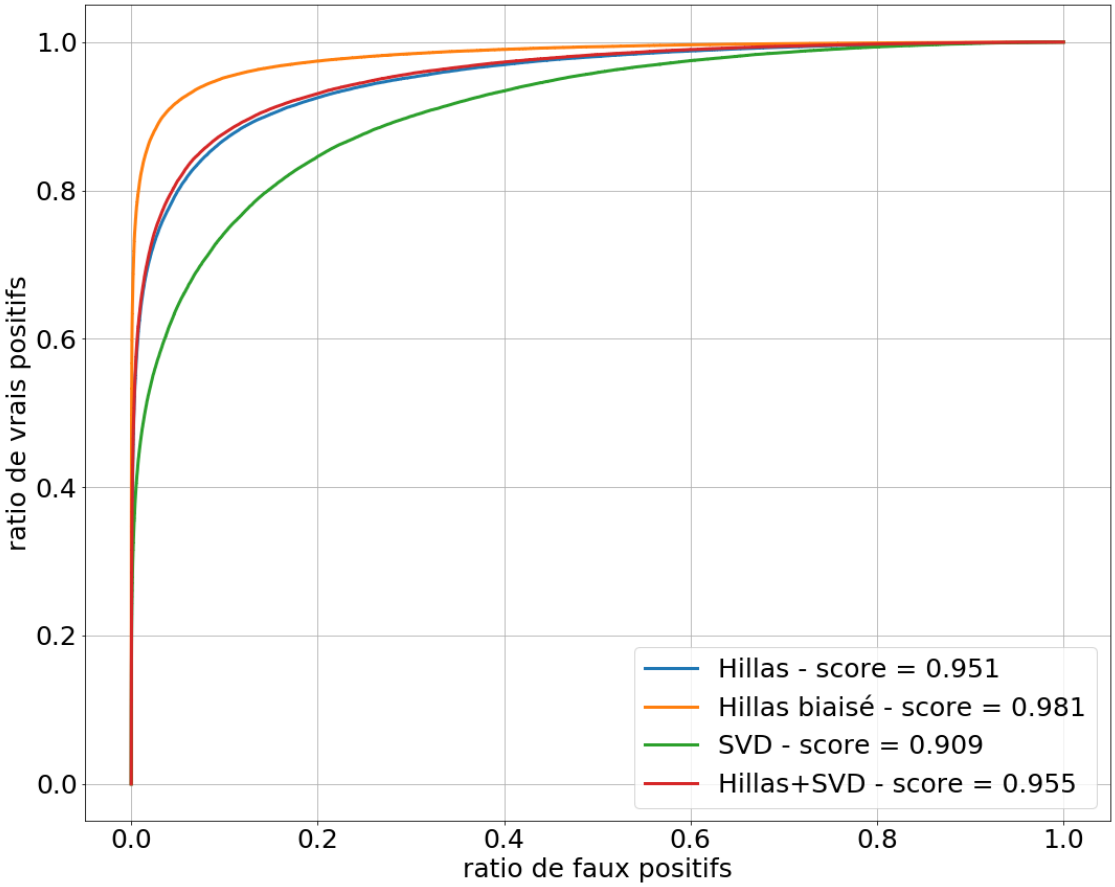


FIGURE 6.35 – Coubres ROC pour différentes configurations Hillas et SVD. Le score correspond à l'aire sous la courbe.

La configuration **Hillas biaisé** n'est présentée qu'à titre indicatif puisque c'est la configuration qui est classiquement utilisée. Cependant, cette configuration comprend des paramètres géométriques tels que le barycentre des ellipses dans les caméras ainsi que la direction reconstruite de la gerbes de particules. D'ordinaire, cela convient, mais dans le cas d'une discrimination entre une source ponctuelle de rayons gamma (au centre du champ de vue des télescopes) et des protons diffus [10] un simple calcul de rayon permet de discriminer, ce qui ne devrait pas être le cas.

La configuration **Hillas** corrige les biais évoquées précédemment en supprimant les paramètres les plus biaisés :

— (g_x, g_y) : la position des ellipses dans les caméras et $g_r = \sqrt{g_x^2 + g_y^2}$ sa norme
— La direction reconstruite de la gerbe de particules

La configuration **SVD** n'utilise que les valeurs SVD. Ce qui implique qu'elle est la moins biaisée.

10. À ces énergies, les protons sont forcément diffus de par la présence des champs magnétiques galactiques et extra-galactiques.

La configuration **Hillas+SVD** ajoute les valeurs singulières aux paramètres Hillas non biaisés. Cela augmente de 5% l'efficacité de la configuration **Hillas**.

Contrairement aux configurations Hillas qui utilisent quasiment tous les paramètres Hillas par ordre d'importance quasiment identique quelque soit les caméras en jeu, l'utilisation des paramètres dans cette configuration est très différents entre les LST-CAM et les NECTAR-CAM. Pour les LST-CAM (à gauche) et les NECTAR-CAM (à droite, les paramètres les plus utiles sont :

Ordre d'importance	LST-CAM	NECTAR-CAM
1	Distance au point d'impact	Valeur singulière 1
2	Valeur singulière 1	Nombre de LST
3	Nombre de pixels sélectionnés	Nombre de pixels sélectionnés
4	Nombre de MST	Distance au point d'impact
5	Nombre de LST	Ordonnée du point d'impact
6	Multiplicité	Valeur singulière 2

Nous remarquons que les NECTAR-CAM utilisent d'avantage les informations SVD et géométriques tandis que les LST utilisent le nombre d'une sorte de télescopes qui a déclenché ou la multiplicité de l'événement. L'ordre d'apparition des valeurs singulières est aussi très différents :

— **LST-CAM** : 1, 2, 13, 4, 3, 21, 14
— **NECTAR-CAM** : 1, 2, 4, 5, 3, 6, 7

On constate que, dans les deux cas, la troisième valeur singulière est moins utile comparée à certaines qui ont des indices plus élevés car elle décrit moins les détails que les suivantes.

La configuration **SVD + Hillas** utilise à la fois les paramètres Hillas et les valeurs singulières. Elle donne des résultats très similaires à la configuration **4** puisque leurs méthodes d'optimisation sont identiques et leurs paramètres également mis à par les valeurs singulières.

Dans les deux cas, les paramètres les plus utiles sont géométriques :

LST-CAM	NECTAR-CAM
Distance au point d'impact	Largeur de la gerbe
Largeur de la gerbe	Distance au point d'impact
Longueur de la gerbe	Longueur de la gerbe

Nous constatons que les paramètres les plus importants décrivent la distance de la gerbe de particules aux télescopes que l'on considère, ainsi que sa largeur. Ce dernier paramètre est directement due à la différence entre une gerbe de particules produite par un proton à une autre produite par un rayon gamma (voir figure 1.13 au chapitre 1).

Néanmoins, la configuration qui n'utilise que les valeurs singulières offre des taux de rejets intéressants, comme :

— Conserver 60% des rayons gamma en acceptant moins de 10% de protons.
— Conserver 80% des rayons gamma en acceptant 20% de protons.

Comme les paramètres associés aux simulations de sources ponctuelles ont un biais certain, nous avons testé la discrimination avec des simulations de rayons gamma diffus.

6.8.2 Discrimination sur des rayons gamma diffus

La discrimination sur des simulations de rayons gamma diffus est *a priori* nettement moins biaisée que celle qui utilise des sources de rayons gamma ponctuelles car les paramètres géométriques comme g_x, g_y ou g_r ne sont plus associés à une position fixe qui est aisément discernable d'une distribution uniforme de protons par un arbre de décision.

Nous avons donc utilisé la même architecture que précédemment avec plusieurs configurations :

— **Hillas** : Arbres de décision, paramètres Hillas, paramètre d'impact, multiplicité, altitude et azimut reconstruits.
— **SVD** : Arbres de décision, valeurs SVD, paramètre d'impact, multiplicité.
— **SVD+Hillas** : Arbres de décision, valeurs SVD, paramètres Hillas, paramètre d'impact, multiplicité, altitude et azimut reconstruits.

La figure 6.36 montre les courbes ROC obtenues.

L'ensemble des courbes décrivent des discriminations moins efficaces que précédemment. Cela est principalement causé par le fait que l'analyse de données a été optimisée physiquement pour une source ponctuelle placée au centre du champ de vue des caméras, et également par le fait que les paramètres fortement biaisés (g_x, g_y ou g_r) ne le sont plus.

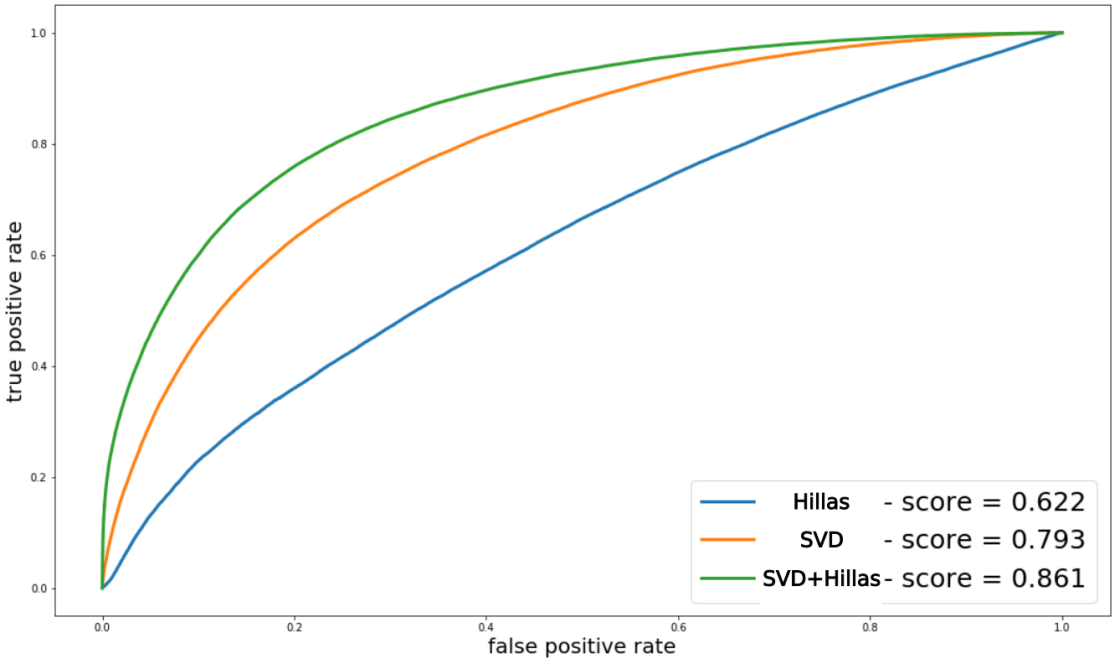


FIGURE 6.36 – Coubres ROC pour différentes configurations Hillas et SVD avec des simulations de rayons gamma diffus. Le score correspond à l'aire sous la courbe.

Nous constatons immédiatement que la discrimination Hillas est très dégradée. Cela peut être dû au nombre de gerbes coupées qui est bien plus important que dans le cas d'une source ponctuelle au centre de la caméra.

La discrimination qui utilise uniquement les valeurs singulières obtient de bien meilleurs résultats car la SVD n'est que très peu sensible aux caractéristiques géométriques des images.

La combinaison des deux configurations précédentes améliore encore la discrimination. Ce cas, est très différent de celui présenté dans la section précédente, où les valeurs SVD n'apportaient quasiment aucune information comparée aux paramètres Hillas, puisque c'est le contraire : le SVD apporte la plupart de l'information utilisable et Hillas renseigne une partie plus faible.

Nous pouvons donc conclure que le SVD permet d'obtenir une discrimination efficace dans le cas de rayons gamma diffus qui est un cas bien plus proche de la réalité qu'une simulation de source ponctuelle au centre de la caméra.

6.9 Conclusion

Ce chapitre présente l'optimisation d'une méthode d'analyse de l'expérience H.E.S.S., *Model++*, dont la discrimination est très efficace, ce qui a pour effet d'améliorer également les reconstructions géométrique et angulaire. Cette méthode, malheureusement lente, se base sur une comparaison d'images produites par des simulations Monte-Carlo au images de l'évé-

nement à analyser.

Nous avons proposé une méthode d'optimisation de comparaison d'image basée sur la décomposition en valeurs singulières (SVD). Cette méthode permet un traitement des images massivement parallèle, ce qui répond aux objectifs de calcul de CTA.

Nous avons commencé par étudier les propriétés de SVD en proposant deux méthodes pour reconstruire l'énergie de la particule incidente en utilisant uniquement les valeurs singulières et la distance au point d'impact.

La première est proche de *Model++* car elle compare les spectres SVD de l'événement à reconstruire à des spectres moyens qui pavent un espace de paramètres physiques connus. La résolution obtenue, environ 40%, ne suffit pas pour répondre aux objectifs de CTA mais montre tout de même que certaines caractéristiques physiques des gerbes de particules sont encodées dans le spectre SVD.

La deuxième méthode fonctionne à base d'un dictionnaire, où chaque spectre SVD est une clé vers une configuration physique. Elle est bien plus précise de la précédente, une résolution en énergie d'environ 20%. Malgré cette amélioration, ces performances ne sont toujours pas suffisantes pour CTA mais elles ont permis de mieux cerner la description SVD de nos images.

Cependant, la force de la méthode *Model++* réside dans la discrimination des rayons gamma et des protons. Une discrimination de qualité améliore indirectement la résolution angulaire et la résolution en énergie de l'analyse de données.

C'est dans cette optique que nous avons testé la discrimination d'événements à partir des spectres SVD.

Dans le cas de simulations d'une source ponctuelle de rayons gamma, l'apport de SVD est minime par rapport aux paramètres Hillas, mais cela est dû aux biais importants causés par cette configuration.

Des tests à base de simulations de rayons gamma diffus montrent que les valeurs SVD améliorent sensiblement la qualité de la discrimination, devant la méthode Hillas.

La combinaison de ces deux méthodes donne les meilleurs résultats.

Ce chapitre à présenté une nouvelle méthode d'optimisation de la méthode template. L'accent a pourtant été mis sur les performances de physique de cet algorithme massivement parallèle car c'est le seul moyen de convaincre la communauté des physiciens de CTA que le HPC à sa place dans les analyses de physique.

Néanmoins, cette analyse pourrait encore être optimisée, en temps de calcul. Il est possible de ne calculer que N valeurs singulières (typiquement 7 ou 15) au lieu de tout le spectre (55). L'utilisation de batch de calculs distribués permettraient de calculer les spectres de valeurs singulières plus efficacement.

Comme nous l'avons mentionné précédemment, une analyse de données rapide ouvre la voie à de nouvelle techniques d'optimisation afin d'obtenir de meilleurs résultats de physique.

L'optimisation des coupures peut se faire avec des algorithmes génétiques (que nous avons testés), des essaims particulaires ou ruches qui nécessites de multiples exécutions de l'analyse de données.

L'essentiel du chapitre 6 –

Ce chapitre expose l'optimisation d'une analyse de données basée sur la comparaison d'images. L'algorithme actuel, utilisé dans l'expérience H.E.S.S., n'est pas adapté à la nouvelle échelle et à la nouvelle structure de données de CTA. Nous présentons deux méthodes de comparaison d'images basée sur la décomposition en valeurs singulières (SVD). Les évaluations des reconstructions en énergie nous ont permis de mieux comprendre la description d'une gerbe de particules en terme de spectre SVD. Nous avons utilisé ces spectres afin d'améliorer la discrimination de notre analyse.

Conclusion et perspectives

L'expérience Cherenkov Telescope Array (CTA) deviendra le plus grand observatoire d'astronomie gamma au sol lors de sa mise en production en 2021. Ce projet représente une série de défis techniques dont le traitement, la compression, le stockage et l'analyse des données récoltées.

Le travail présenté dans ce manuscrit s'est concentré sur le traitement de ces données et plus particulièrement sur leur analyse.

L'optimisation de l'analyse de données ne peut être pertinente que par l'utilisation d'un format de données adapté. C'est dans cette optique sue nous avons développé des générateurs de formats de données afin d'automatiser leur optimisation.

Le traitement des données en amont de l'analyse peut porter préjudice à cette dernière si elle n'est pas suffisamment optimisée. Comme ce traitement consistera en la compression et la décompression des données, nous avons donc développé une méthode de compression et de décompression rapide d'entiers non signés qui peut également être utilisée comme préconditionnement pour d'autres algorithmes comme LZMA.

Nous avons d'abord optimiser l'analyse de données la plus simple : la méthode de Hillas. La vitesse de cette analyse, essentiellement mathématique, a été grandement améliorée (600 fois plus rapide que l'analyse officielle de CTA) par les méthodes de calcul haute performance que nous avons utilisées. Cela nous a permis de montré que notre analyse, qui utilise un format de données adapté, n'est en aucun cas dominée par la lecture des données. En effet, cette dernière représente moins de 1% du temps de calcul total de notre analyse.

Nous avons ensuite développé une nouvelle méthode de comparaison d'image pour optimiser l'analyse la plus évoluée dans l'expérience H.E.S.S. qui est dite "template". Bien que les tests réalisés avec les simulations de l'expérience H.E.S.S. soient prometteurs, ceux qui utilisaient les simulations de CTA n'offraient pas la précision nécessaire pour la mise en production d'une telle analyse.

Le travail faisant suite à cette thèse consistera à inclure nos fonctions de calcul haute performance dans l'analyse officielle de CTA (écrite en Python). Cela permettra ainsi à la communauté de CTA de réduire le temps de leurs analyses de données.

Parallèlement à cela, et aux vues de nos résultats, nous avons été sollicité pour participer au développement de l'analyse de données du prototype de grand télescope (LST) qui sera mis en service à la mi-Octobre 2018. Et nous prendrons également part au développement de l'analyse en temps réel de ce prototype ce qui nous permettra de tester nos algorithmes sur le terrain.

Nous participerons également à la définition des propriétés des nouvelles simulations pour CTA, ce qui nous permettra à terme de tester notre reconstruction template SVD dans de meilleures conditions.

Appendices

Annexe A

Calcul des paramètres Hillas

Soit un vecteur $\mathbf{s} \in \mathbb{R}^N$ composé de N pixels, qui décrit le signal calibré d'une caméra, donné par l'équation :

$$s = \begin{cases} \dfrac{(Adc_{hi} - Ped_{hi}) \cdot Gain_{hi}}{Flatfield_{hi}} & \text{si la voie haut gain n'est pas saturée} \\ \dfrac{(Adc_{lo} - Ped_{lo}) \cdot Gain_{lo}}{Flatfield_{lo}} & \text{si la voie haut gain est saturée mais pas la voie bas gain} \\ -1 & \text{sinon} \end{cases} \quad \text{(A.1)}$$

Un vecteur $\mathbf{c} \in [\![0, 1]\!]$ définit si un pixel doit être pris en compte dans les calculs (1), ou non (0). Les positions des pixels, dans la caméra sont décrites par le vecteur $\mathbf{p} \in \mathbb{R}^N$

Le nombre de pixels sélectionnés, n_p, et l'amplitude de la gerbe de particules dans la caméra, $\mathcal{A}$, sont donnés par :

$$\begin{aligned} n_p &= \sum_{i=0}^{N} \mathbf{c}_i \\ \mathcal{A} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \mathbf{c}_i \end{aligned}$$

Si $n_p = 0$ ou $\mathcal{A} = 0$ l'image est rejetée. Moins de 20% des événements sont rejetés par cette coupure.

Quelques variables préliminaires sont nécessaires au calcul des paramètres Hillas :

$$\begin{aligned}
s_x &= \sum_{i=0}^{N} \mathbf{p}_x^i \cdot \mathbf{s}_i \cdot \mathbf{c}_i \\
s_y &= \sum_{i=0}^{N} \mathbf{p}_y^i \cdot \mathbf{s}_i \cdot \mathbf{c}_i \\
s_{xx} &= \frac{1}{\mathcal{A}} \left(\sum_{i=0}^{N} \left(\mathbf{p}_x^i\right)^2 \cdot \mathbf{s}_i \cdot \mathbf{c}_i \right) - \left(\frac{s_x}{\mathcal{A}}\right)^2 \\
s_{yy} &= \frac{1}{\mathcal{A}} \left(\sum_{i=0}^{N} \left(\mathbf{p}_y^i\right)^2 \cdot \mathbf{s}_i \cdot \mathbf{c}_i \right) - \left(\frac{s_y}{\mathcal{A}}\right)^2 \\
s_{xy} &= \frac{1}{\mathcal{A}} \left(\sum_{i=0}^{N} \mathbf{p}_x^i \cdot \mathbf{p}_y^i \cdot \mathbf{s}_i \cdot \mathbf{c}_i \right) - \left(\frac{s_x \cdot s_y}{\mathcal{A}}\right)^2
\end{aligned}$$

Le barycentre, (x, y), de la gerbe de particules dans la caméra est :

$$\begin{aligned}
x &= \frac{1}{\mathcal{A}} s_x \\
y &= \frac{1}{\mathcal{A}} s_y
\end{aligned}$$

Son orientation, ϕ, est calculée à l'aide de plusieurs autres variables :

$$\begin{aligned}
rk &= s_{yy} - s_{xx} \\
rl &= \sqrt{rk^2 + 4 \cdot s_{xy}^2} \\
up &= (rk + rl) \cdot s_y + 2 \cdot s_{xy} \cdot s_x \\
down &= 2 \cdot s_{xy} \cdot s_y - (rk - rl) \cdot s_x
\end{aligned}$$

$$\phi = \arctan(up, down)$$

La longueur, l, et la largeur, w de l'ellipse sont calculées par :

$$aval = \frac{(rk + rl)}{2 \cdot s_{xy}}$$

$$\begin{aligned}
w &= \sqrt{\frac{\|s_{yy} + aval^2 \cdot s_{xx} - 2 \cdot aval \cdot s_{xy}\|}{aval^2 + 1}} \\
l &= \sqrt{\frac{\|s_{xx} + aval^2 \cdot s_{yy} - 2 \cdot aval \cdot s_{xy}\|}{aval^2 + 1}}
\end{aligned}$$

Si la longueur ou la largeur de l'ellipse calculées ne sont pas infinie, d'autres paramètres sont introduits :

$$\begin{aligned}
x_{\phi}^{i} &= \cos(\phi) \cdot \left(\mathbf{p}_{x}^{i} - sx\right) + \sin(\phi) \cdot \left(\mathbf{p}_{y}^{i} - sy\right) \\
y_{\phi}^{i} &= -\sin(\phi) \cdot \left(\mathbf{p}_{x}^{i} - sx\right) + \cos(\phi) \cdot \left(\mathbf{p}_{y}^{i} - sy\right) \\
sx_{\phi^2} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \left(x_{\phi}^{i}\right)^2 \cdot \mathbf{c}_i \\
sx_{\phi^3} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \left(x_{\phi}^{i}\right)^3 \cdot \mathbf{c}_i \\
sy_{\phi^2} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \left(y_{\phi}^{i}\right)^2 \cdot \mathbf{c}_i \\
sy_{\phi^3} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \left(y_{\phi}^{i}\right)^3 \cdot \mathbf{c}_i \\
sy_{\phi^4} &= \sum_{i=0}^{N} \mathbf{s}_i \cdot \left(y_{\phi}^{i}\right)^4 \cdot \mathbf{c}_i
\end{aligned}$$

L'asymétrie, s_{ϕ}, de la gerbe de particules et la séparation de Hofmann sont données par :

$$\begin{aligned}
s_{\phi} &= sx_{\phi^3} \frac{\sqrt{\mathcal{A} \cdot sx_{\phi^2}}}{sx_{\phi^2}^2} \\
h_s &= \frac{sy_{\phi^4} \cdot \mathcal{A}}{sy_{\phi^2}^2 \cdot n_p}
\end{aligned}$$

La direction de la gerbe de particules est donnée par :

$$\begin{aligned}
bval &= (s_y - aval \cdot s_x) \\
d &= \arctan\left(bval, \frac{bval}{aval}\right)
\end{aligned}$$

L'asymétrie de l'image, s, et l'excès de Kurtosis sont calculés comme suit :

$$\begin{aligned}
xp_i &= \cos(d) \cdot (x - sx) + \sin(d) \cdot (y - sy) \\
sx_2 &= \sum_{i=0}^{N} \mathbf{s}_i \cdot xp_i^2 \cdot \mathbf{c}_i \\
sx_3 &= \sum_{i=0}^{N} \mathbf{s}_i \cdot xp_i^3 \cdot \mathbf{c}_i \\
sx_4 &= \sum_{i=0}^{N} \mathbf{s}_i \cdot xp_i^4 \cdot \mathbf{c}_i
\end{aligned}$$

Finalement, les paramètres s et k sont donnés par :

$$\begin{aligned} s &= \frac{sx_3}{sx_2^{1.5}} \\ k &= \frac{sx_4}{sx_2^2} - 3 \end{aligned}$$

Annexe B

Mathématiques

B.1 Matrice Jacobienne

Soit une fonction F composée de m fonctions composantes dans $\mathbb{R}$:

$$F : \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \longmapsto \begin{pmatrix} f_1(x_1, \cdots, x_n) \\ \vdots \\ f_m(x_1, \cdots, x_n) \end{pmatrix} \tag{B.1}$$

Où les fonctions f_i sont définies comme suit :

$$\begin{array}{rccc} f_i & : & \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ & & (x_1, \cdots, x_n) & \longmapsto & z_i \end{array} \tag{B.2}$$

La matrice Jacobienne, J_F, de la fonction F est :

$$J_F(M) = \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{pmatrix} \tag{B.3}$$

B.2 Exemple de matrice Jacobienne

Soit une fonction F à 2 composantes :

$$F : \begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} f_1(x, y) = x^2 + y^2 \\ f_2(x, y) = x \cdot y \end{pmatrix} \tag{B.4}$$

La matrice Jacobienne, J_F, de la fonction F est :

$$J_F(x,y) = \begin{pmatrix} 2x & 2y \\ y & x \end{pmatrix} \tag{B.5}$$

La transposé de la matrice J est J_F^T :

$$J_F^T(x,y) = \begin{pmatrix} 2x & y \\ 2y & x \end{pmatrix} \tag{B.6}$$

Le produit matriciel $J_F^T \times J_F$ donne :

$$J_F^T(x,y) \times J_F(x,y) = \begin{pmatrix} 2x & y \\ 2y & x \end{pmatrix} \times \begin{pmatrix} 2x & 2y \\ y & x \end{pmatrix} \tag{B.7}$$

$$= \begin{pmatrix} 4x^2+y^2 & 4xy+xy \\ 4xy+xy & 4x^2+y^2 \end{pmatrix} \tag{B.8}$$

$$= \begin{pmatrix} 4x^2+y^2 & 5xy \\ 5xy & 4x^2+y^2 \end{pmatrix} \tag{B.9}$$

B.3 Méthode de Gauss-Newton

L'algorithme de Gauss-Newton permet de résoudre des problèmes non linéaires. C'est la généralisation multi dimensionnelle de la méthode de Newton-Raphson.

L'algorithme de Gauss-Newton permet de trouver le minimum d'une fonction dans calculer ses dérivées secondes. Elle peut être utilisée pour ajuster des courbes sur des données.

Soient m fonctions $r_i\,(i = 1, \cdots, m)$ de n variables $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$. Chaque β donne un vecteur de résidus qui doit être minimisé pour ajuster les données. Si $m \geq n$ l'algorithme de Gauss-Newton permet de trouver le minimum des carrés :

$$S(\beta) = \sum_{i=1}^{m} r_i^2(\beta) \tag{B.10}$$

Le minimum de β est $\beta^{\mathbf{0}}$. Alors, une itération est défini comme :

$$\beta^{\mathbf{s+1}} = \beta^{\mathbf{s}} + \delta\beta \tag{B.11}$$

Où $\delta\beta$ satisfait l'équation normale :

$$\left(J_{\mathbf{r}}^T J_{\mathbf{r}}\right)\delta\beta = -J_{\mathbf{r}}^T \mathbf{r} \tag{B.12}$$

Où $\mathbf{r} = (r_1, r_2, \cdots, r_m)$, et $J_{\mathbf{r}}$ la matrice Jacobienne $m \times n$ de $\mathbf{r}$ en fonction de β, et $J_{\mathbf{r}}^T$ sa matrice transposée. Les paramètres β d'un modèle $y = f(x, \beta)$ doivent être trouvés pour ajuster les données (x_i, y_i). Les résidus, r_i, sont donnés par :

$$r_i(\beta) = y_i - f(x_i, \beta) \tag{B.13}$$

Donc, $\delta\beta$ s'écrit comme une fonction de f :

$$\left(J_{\mathbf{f}}^T J_{\mathbf{f}}\right) \delta\beta = J_{\mathbf{f}}^T \mathbf{r} \tag{B.14}$$

Remplacer l'équation B.14 dans B.11 donne :

$$\beta^{\mathbf{s+1}} = \beta^{\mathbf{s}} - \left(J_{\mathbf{r}}^T J_{\mathbf{r}}\right)^{-1} J_{\mathbf{r}}^T \mathbf{r} \tag{B.15}$$

Cette méthode est une sorte d'algorithme de Newton-Raphson à N dimensions. Si $N = 1$, on retrouve l'équation classique de Newton-Raphson :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{B.16}$$

Cette méthode est très efficace si l'erreur quadratique est faible ou si les fonctions utilisées ne sont pas trop non linéaires. Certains problèmes de convergences peuvent apparaître si le point de départ est trop éloigné du point de convergence ou si la matrice $J_{\mathbf{r}}$ est mal conditionnée[1]. Dans ce dernier cas, une matrice d'échelle peut être utilisée afin d'obtenir un meilleur conditionnement de la matrice $J_{\mathbf{r}}$.

B.4 Méthode de Levenberg-Marquardt

La méthode de Levenberg-Marquardt est proche de la méthode du gradient. Elle améliore celle de Gauss-Newton en permettant de changer les pas de convergence au fur et à mesure des résultats obtenues lors de cette convergence. La propriété Mathématique forte, est que cet algorithme peut converger même si le point de départ est loin du point de convergence. Mais il peut être un peu plus lent sur des fonctions irrégulières.

Le cas général est donné par :

$$\beta^{\mathbf{s+1}} = \beta^{\mathbf{s}} + \alpha\delta\beta \tag{B.17}$$

Il peut arriver que le paramètre α de l'équation B.17 soit très proche de 0. L'astuce de la méthode de Levenberg-Marquardt est de modifier l'équation B.15 de sorte à ce que l'incrément se dirige vers le gradient le plus grand.

$$\left(J^T J + \lambda D\right) \delta\beta = J^T r \tag{B.18}$$

1. Si le rapport entre la plus grande valeur et la plus petite n'est pas trop grand.

Où D est une matrice diagonale positive (généralement la matrice identité), et λ, le paramètre de Marquardt. Ce paramètre peut être déterminé avec une recherche linéaire mais cela ralenti l'algorithme car il doit être recalculé à chaque itération.

La méthode la plus simple consiste à augmenter λ quand la diminution du résidu s'accélère, et le diminuer lorsque celle-ci décélère.

Annexe C

Effet des pixels morts sur les spectres SVD

Les simulations que nous utilisons modélisent des caméras parfaites. Or, celles-ci peuvent subir des pannes notamment au niveaux des pixels. Tous les pixels momentanément inutilisables sont appelés pixels morts car ils doivent être mis à l'écart pour l'analyse de données. Nous devons donc étudier l'effet de ces pixels morts sur les valeurs singulières afin de préparer au mieux l'utilisation des caméra réelles.

Les caméras ne sont pas parfaites. Certains pixels peuvent être saturés à cause d'une étoile et sont donc rejeter à l'étape de calibration (voir section 5.3). D'autres peuvent avoir des défauts d'alimentation comme des pannes ou des problèmes de variations de la tension électriques. Certains peuvent être hors service à cause du sable, des hautes températures ou de mauvaises manipulations.

Dans tout les cas, ces pixels ne doivent pas être utilisés dans l'analyse de données, ce sont des pixels morts. Les caméras de H.E.S.S. ont jusqu'à 10% de pixels morts, c'est pour cela que l'analyse de données doit être robuste.

Nous avons réalisé des simulations avec 1%, 5% et 10% de pixels morts distribués uniformément dans une caméra LST-CAM. Les gerbes de particules utilisées ont des énergies de 0.106206 TeV, 0.21732 TeV et 0.513144 TeV, car leur effets sont plus important dus à leurs tailles dans la caméra.

C.1 Étalement de la distribution des valeurs singulières

Les figures C.1, C.2 et C.3 montrent l'influence des pixels morts sur les quatre premières valeurs singulières pour des gerbes d'énergie : 0.106206 TeV, 0.21732 TeV et 0.513144 TeV respectivement.

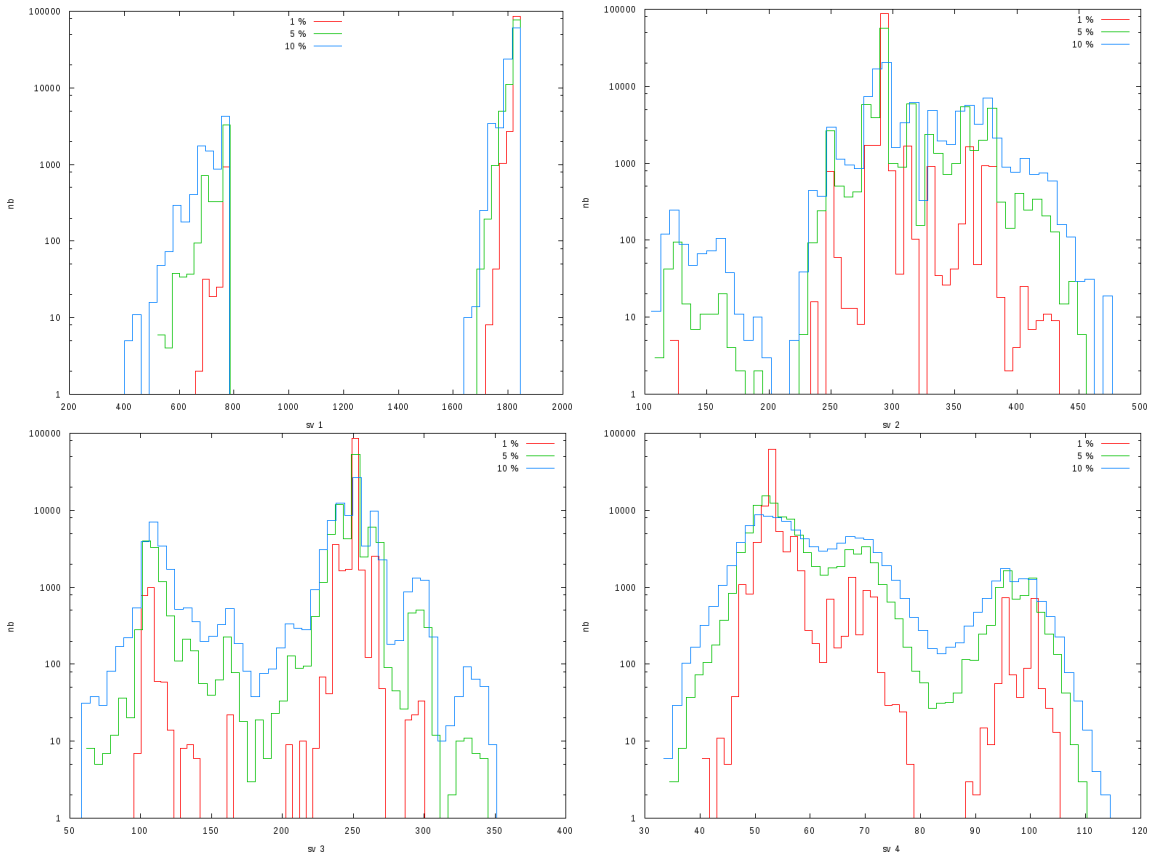


FIGURE C.1 – Évolution des première (en haut à gauche), deuxième (en haut à droite), troisième (en bas à gauche) et quatrième (en bas à droite), valeurs singulières pour une gerbe à 0.106206 TeV avec $1, 5, 10\%$ de pixels morts, respectivement montré par les courbes rouge, verte et bleue.

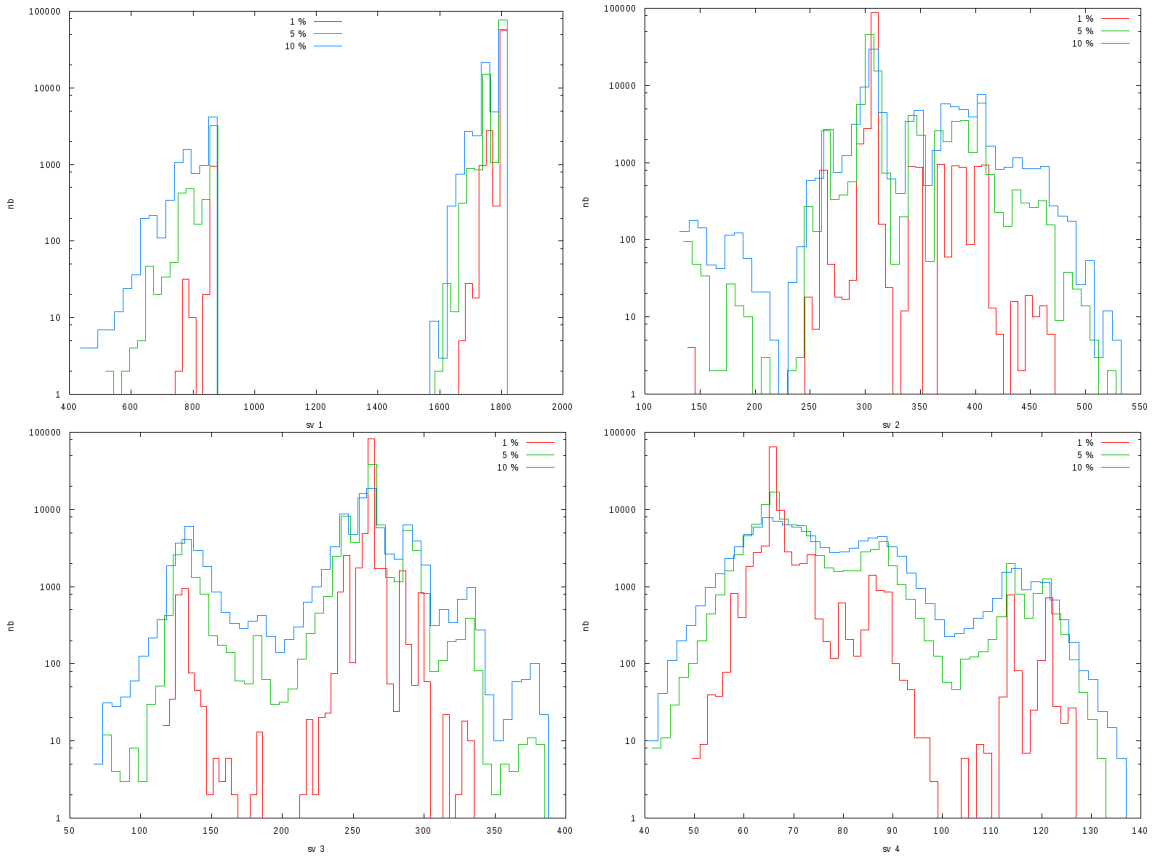


FIGURE C.2 – Évolution des première (en haut à gauche), deuxième (en haut à droite), troisième (en bas à gauche) et quatrième (en bas à droite), valeurs singulières pour une gerbe à 0.21732 TeV avec $1, 5, 10\%$ de pixels morts, respectivement montré par les courbes rouge, verte et bleue.

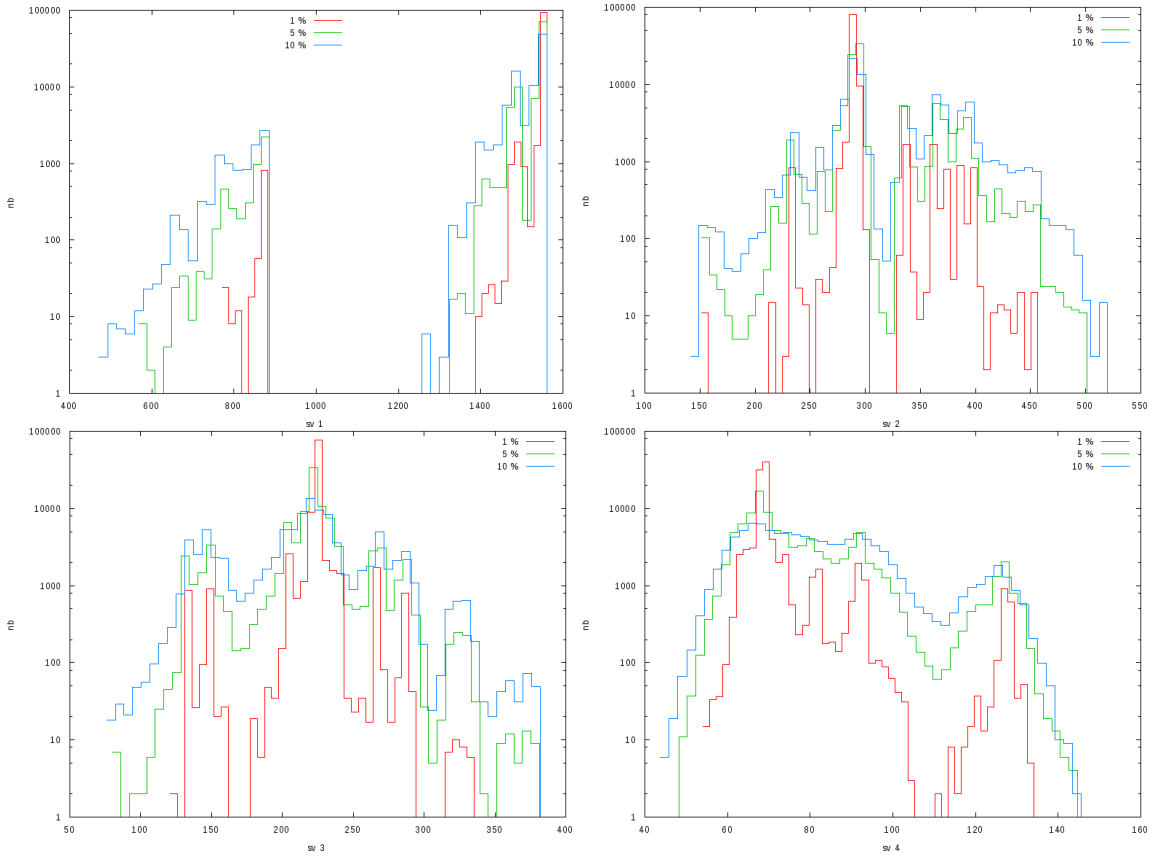


FIGURE C.3 – Évolution des première (en haut à gauche), deuxième (en haut à droite), troisième (en bas à gauche) et quatrième (en bas à droite), valeurs singulières pour une gerbe à 0.513144 TeV avec $1, 5, 10\%$ de pixels morts, respectivement montré par les courbes rouge, verte et bleue.

C.2 Influence sur la discrimination de l'énergie

Les pixels morts dégradent généralement la qualité de la reconstruction en énergie, spécialement pour les faibles énergies car leurs gerbes associées sont plus petites dans la caméra, donc un pixel mort occulte proportionnellement plus de signal.

La figure C.4 montre la sensibilité de la première valeur singulière en fonction de différentes proportions de pixels morts sur trois énergies.

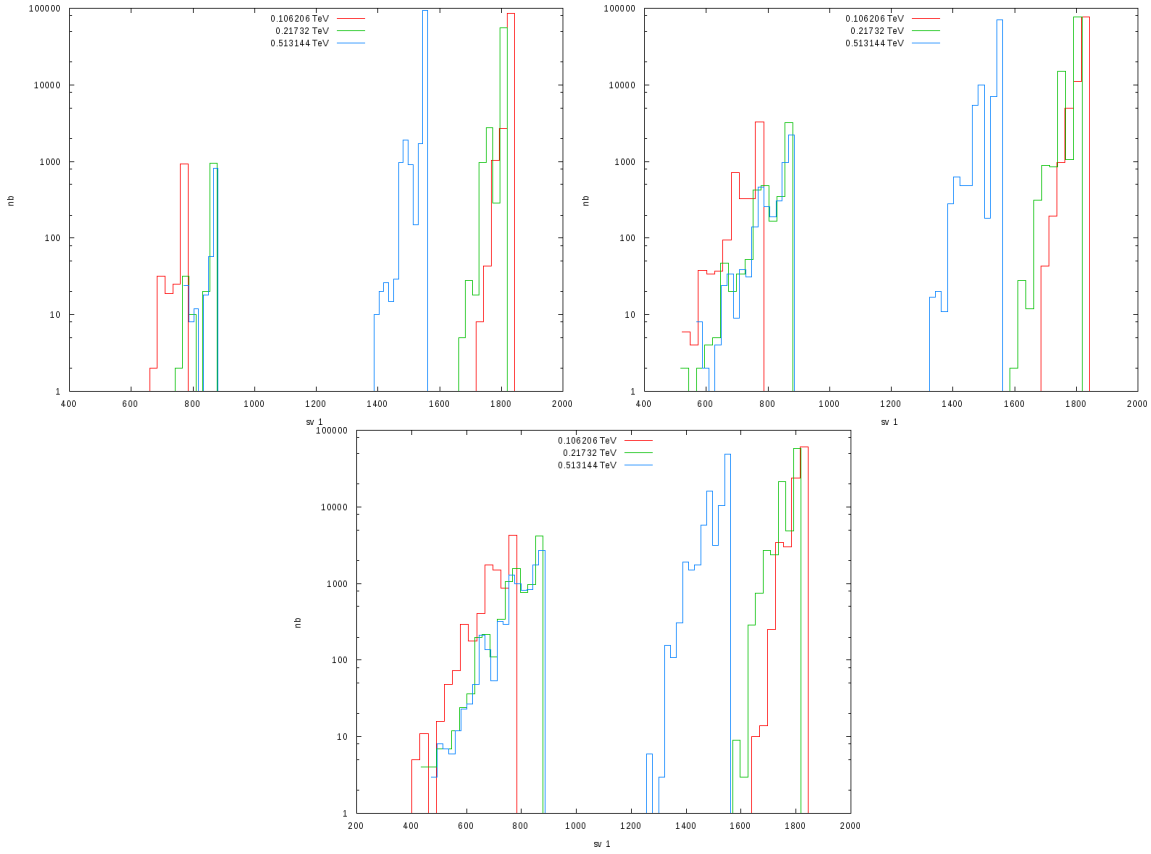


FIGURE C.4 – Évolution de la première valeur singulière sur une faible énergie 0.106206 TeV (rouge), 0.21732 TeV (vert) et 0.513144 TeV (bleu) dans une LST-CAM avec $1, 5, 10\%$ de pixels morts respectivement en haut à gauche, en haut à droite et en bas.

C.3 Correction de l'effet des pixels morts

Une correction assez simple peut être effectuée pour atténuer l'effet des pixels morts sur la magnitude des valeurs singulières. La valeur d'un pixel mort est remplacée la moyenne de ses voisins. La figure C.5 montre comment cela diminue l'erreur due à 10% de pixels morts.

Cependant, ces corrections sont limitées si plusieurs pixels adjacents sont manquants, mais cette méthode fonctionne dans la plupart des cas.

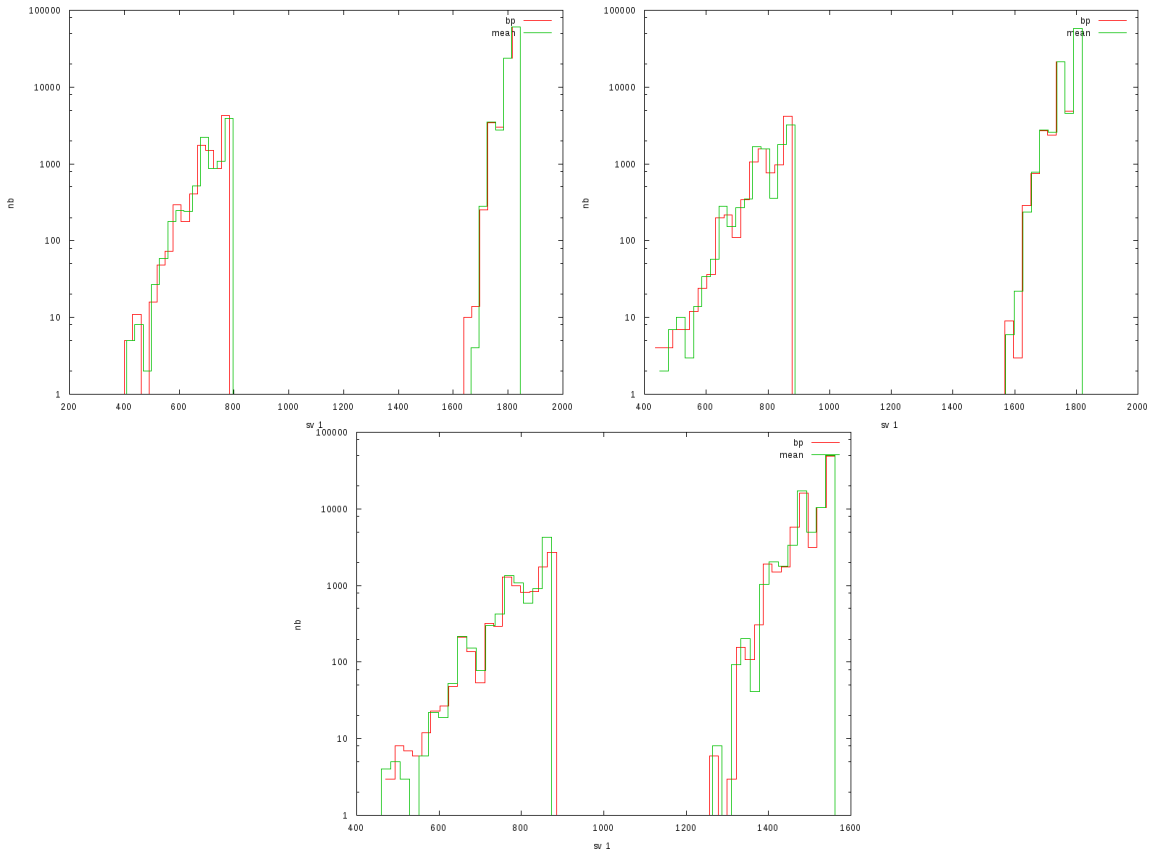


FIGURE C.5 – Évolution de la première valeur singulière à basse énergie 0.106206 TeV (en haut à gauche), 0.21732 TeV (en haut à droite) et 0.513144 TeV (en bas) dans une LST-CAM avec 10% de pixel morts, sans correction en rouge et avec une moyenne en vert.

Annexe D

Intersection multiple de droites à trois dimensions

Cette annexe propose le calcul explicite d'une intersection de plusieurs droites qui ne se croisent pas forcément en un seul point.

Soit une droite D, passant par un point A, de vecteur directeur $\mathbf{u}$. La distance entre un point P et la droite $\mathcal{D}$ est :

$$d = \frac{\|\mathbf{AP} \wedge \mathbf{u}\|}{\|\mathbf{u}\|}$$

Nous nous plaçons dans le cas où $\|\mathbf{u}\| = 1$, donc :

$$\begin{aligned} d &= \|\mathbf{AP} \wedge \mathbf{u}\| \\ &= \left\| \begin{pmatrix} p_x - a_x \\ p_y - a_y \\ p_z - a_z \end{pmatrix} \wedge \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \right\| \\ &= \left\| \begin{pmatrix} (p_y - a_y)\, u_z - (p_z - a_z)\, u_y \\ (p_z - a_z)\, u_x - (p_x - a_x)\, u_z \\ (p_x - a_x)\, u_y - (p_y - a_y)\, u_x \end{pmatrix} \right\| \end{aligned}$$

On obtient :

$$d^2 = \left\| \begin{pmatrix} (p_y - a_y)\, u_z - (p_z - a_z)\, u_y \\ (p_z - a_z)\, u_x - (p_x - a_x)\, u_z \\ (p_x - a_x)\, u_y - (p_y - a_y)\, u_x \end{pmatrix} \right\|^2$$

$$\begin{aligned} d^2 &= \left[(p_y - a_y)\, u_z - (p_z - a_z)\, u_y\right]^2 \\ &+ \left[(p_z - a_z)\, u_x - (p_x - a_x)\, u_z\right]^2 \\ &+ \left[(p_x - a_x)\, u_y - (p_y - a_y)\, u_x\right]^2 \\ &= (p_y - a_y)^2 u_z^2 + (p_z - a_z)^2 u_y^2 - 2\,(p_y - a_y)\,(p_z - a_z)\, u_y u_z \\ &+ (p_z - a_z)^2 u_x^2 + (p_x - a_x)^2 u_z^2 - 2\,(p_z - a_z)\,(p_x - a_x)\, u_x u_z \\ &+ (p_x - a_x)^2 u_y^2 + (p_y - a_y)^2 u_x^2 - 2\,(p_x - a_x)\,(p_y - a_y)\, u_x u_y \\ &= \left(p_y^2 + a_y^2 - 2p_y a_y\right) u_z^2 + \left(p_z^2 + a_z^2 - 2p_z a_z\right) u_y^2 - 2\,(p_y p_z - a_z p_y - a_y p_z + a_y a_z)\, u_y u_z \\ &+ \left(p_z^2 + a_z^2 - 2p_z a_z\right) u_x^2 + \left(p_x^2 + a_x^2 - 2p_x a_x\right) u_z^2 - 2\,(p_z p_x - p_z a_x - a_z p_x + a_x a_z)\, u_x u_z \\ &+ \left(p_x^2 + a_x^2 - 2p_x a_x\right) u_y^2 + \left(p_y^2 + a_y^2 - 2p_y a_y\right) u_x^2 - 2\,(p_x p_y - p_x a_y - a_x p_y + a_x a_y)\, u_x u_y \\ &= p_y^2 u_z^2 + a_y^2 u_z^2 - 2p_y a_y u_z^2 + p_z^2 u_y^2 + a_z^2 u_y^2 - 2p_z a_z u_y^2 - 2p_y p_z u_y u_z \\ &+ 2a_z p_y u_y u_z + 2a_y p_z u_y u_z - 2a_y a_z u_y u_z \\ &+ p_z^2 u_x^2 + a_z^2 u_x^2 - 2p_z a_z u_x^2 + p_x^2 u_z^2 + a_x^2 u_z^2 - 2p_x a_x u_z^2 - 2p_z p_x u_x u_z \\ &+ 2p_z a_x u_x u_z + 2a_z p_x u_x u_z - 2a_x a_z u_x u_z \\ &+ p_x^2 u_y^2 + a_x^2 u_y^2 - 2p_x a_x u_y^2 + p_y^2 u_x^2 + a_y^2 u_x^2 - 2p_y a_y u_x^2 - 2p_x p_y u_x u_y \\ &+ 2p_x a_y u_x u_y + 2a_x p_y u_x u_y - 2a_x a_y u_x u_y \end{aligned}$$

La somme des carrés des distances, $D_i^2(P)$, entre un point, P, et N droites est donné par :

$$D^2(P) = \sum_{i=1}^{N} D_i^2(P)$$

Cette distance est minimale lorsque :

$$\frac{\partial D^2(P)}{\partial p_j} = 0 \quad j = x, y, z$$
$$\sum_{i=1}^{N} \frac{D_i^2(P)}{\partial p_j} = 0 \quad j = x, y, z$$

Or :

$$\begin{aligned}
\frac{D_i^2(P)}{\partial p_x} &= 2p_x u_z^2 - 2a_x u_z^2 - 2p_z u_x u_z + 2a_z u_x u_z + 2p_x u_y^2 - 2a_x u_y^2 - 2p_y u_x u_y + 2a_y u_x u_y \\
&= 2p_x\left(u_z^2 + u_y^2\right) - 2p_y u_x u_y - 2p_z u_x u_z + 2\left(u_z\left(a_z u_x - a_x u_z\right) + u_y\left(a_y u_x - a_x u_y\right)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{D_i^2(P)}{\partial p_y} &= 2p_y u_z^2 - 2a_y u_z^2 - 2p_z u_y u_z + 2a_z u_y u_z + 2p_y u_x^2 - 2a_y u_x^2 - 2p_x u_x u_y + 2a_x u_x u_y \\
&= -2p_x u_x u_y + 2p_y\left(u_x^2 + u_z^2\right) - 2p_z u_y u_z + 2\left(u_z\left(a_z u_y - a_y u_z\right) + u_x\left(a_x u_y - a_y u_x\right)\right)
\end{aligned}$$

$$\begin{aligned}
\frac{D_i^2(P)}{\partial p_z} &= 2p_z u_y^2 - 2a_z u_y^2 - 2p_y u_y u_z + 2a_y u_y u_z + 2p_z u_x^2 - 2a_z u_x^2 - 2p_x u_x u_z + 2a_x u_x u_z \\
&= -2p_x u_x u_z - 2p_y u_y u_z + 2p_z\left(u_x^2 + u_y^2\right) + 2\left(u_y\left(a_y u_z - a_z u_y\right) + u_x\left(a_x u_z - a_z u_x\right)\right)
\end{aligned}$$

Si on pose (où u et a dépendent des indices i) :

$$\begin{aligned}
A &= \sum_{i=1}^{N} u_y^2 + u_z^2 & Q_1 &= -\sum_{i=1}^{N} u_x u_z \\
B &= \sum_{i=1}^{N} u_z^2 + u_x^2 & Q_2 &= -\sum_{i=1}^{N} u_x u_y \\
C &= \sum_{i=1}^{N} u_y^2 + u_x^2 & Q_3 &= -\sum_{i=1}^{N} u_y u_z
\end{aligned}$$

$$\begin{aligned}
D_1 &= \sum_{i=1}^{N} u_z\left[a_z u_x - a_x u_z\right] + u_y\left[a_x u_y - a_y u_x\right] \\
D_2 &= \sum_{i=1}^{N} u_x\left[a_x u_y - a_y u_x\right] + u_z\left[a_y u_z - a_z u_y\right] \\
D_3 &= \sum_{i=1}^{N} u_y\left[a_y u_z - a_z u_y\right] + u_x\left[a_z u_x - a_x u_z\right]
\end{aligned}$$

Comme on demande que l'expression s'annule, on élimine le 2.

On obtient le système :

$$\begin{pmatrix} A & Q_2 & Q_1 \\ Q_2 & B & Q_3 \\ Q_1 & Q_3 & C \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

D'où :

$$\begin{pmatrix} A & Q_2 & Q_1 \\ Q_2 & B & Q_3 \\ Q_1 & Q_3 & C \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = - \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix}$$

Donc :

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = - \begin{pmatrix} A & Q_2 & Q_1 \\ Q_2 & B & Q_3 \\ Q_1 & Q_3 & C \end{pmatrix}^{-1} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix}$$

$$D = A\left(BC - Q_3^2\right) + Q_2\left(Q_2C - Q_3Q_1\right) + Q_1\left(Q_2Q_3 - BQ_1\right)$$

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = -\frac{1}{D} \begin{pmatrix} BC - Q_3^2 & -Q_2C + Q_1Q_3 & Q_2Q_3 - Q_1B \\ -Q_2C + Q_3Q_1 & AC - Q_1^2 & -AQ_3 + Q_1Q_2 \\ Q_2Q_3 - BQ_1 & -AQ_3 + Q_2Q_1 & AB - Q_2^2 \end{pmatrix} \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix}$$

Comme les télescopes sont équipés de miroir, chaque pixel voit un angle solide différent dans le ciel. Par conséquent, le barycentre (g_x, g_y) correspond à une direction en altitude et azimut. Par convention, (g_x, g_y) est un angle dans la caméra. La direction du barycentre projeter de la gerbe est $(t_{alt} - g_x, t_{az} - g_y)$. Les signes moins apparaissent du fait de la présence d'un miroir sur le télescope et (t_{alt}, t_{az}) est la direction de pointé des télescopes en altitude et en azimut.

Bibliographie

[1] Theodor Wulf. Phys. Zeit. 1910.

[2] Victor Franz Hess. Observation of penetrating radiation in seven balloon flights. 1912.

[3] W. Kolhörrster. Ber. Deutsch. Phys. Ges. 1914-19.

[4] Millikan and Holfman. Nat. Accad. Sci. Pro. 1926.

[5] J. Clay. Proceedings accademy of Amsterdam. 1930.

[6] A. H. Compton. A geographic study of cosmic rays. 1933.

[7] Bothe and Kolhörrster. Nature of high altitude radiation. *Z. Phys.*, 1929.

[8] D. V. Skobelzym. A new type of very fast beta rays. *Z. Phys.*, 1929.

[9] B. Rossi. . *Z. Phys.*, 1933.

[10] P.A.M. Dirac. Quantised singularity in electromagnetic field. *Proc. Roy. Soc.*, 1931.

[11] C.D. Anderson. The apparent existence of easly deflectable positives. 1932.

[12] C.D. Anderson and S.H. Neddermeyer. Note of the nature of cosmic-ray particles. *Phys. Rev.*, 1937.

[13] R. Maze et al. P. Auger. Compte rendu de l'accadémie des sciences. 1938.

[14] Asakimori et al. *Proc 33rd inter Cosmic Ray Conf. (Galary)*, 1993.

[15] Jean Noël Capdevielle et al. *Proc 16th ICRC (Kyoto)*, 1979.

[16] R. Batiston. Astro particle physic with ams on the international space station. *J. Phys. G*, 2003.

[17] Halzen et al. The highest energy cosmic ray. *Astropart. Phys.*, 1995.

[18] S. Yoshida et al. The cosmic ray energy spectrum above 3.10^{18} eV measurement by akeno giant air shower array. *Proc. 8th ICRC*, 1964.

[19] Serguei for the collaboration) Pierre Auger Collaboration (Vorobiov. The pierre auger observatory : a new stage to the ultra-high energetic cosmic rays. 2008.

[20] M Nagano. Search for the end of the energy spectrum of primary cosmic rays. *New Journal of Physics*, 11(6) :065012, 2009.

[21] Stefano Gabici and Felix A. Aharonian. Searching for galactic cosmic-ray pevatrons with multi-tev gamma rays and neutrinos. *The Astrophysical Journal Letters*, 665(2) :L131, 2007.

[22] E. Fermi. On the origin of the cosmic radiation. *Phys. Rev. 75*, 1949.

[23] Fermi/LAT Collaboration : W. B. Atwood, et al. The Large Area Telescope on the Fermi Gamma-ray Space Telescope Mission. *Astrophys.J.697 :1071-1102,2009*, 2009.

[24] W. Hofmann. Internationnal cosmic ray conference hamburg. In *conference 27th*, 2001.

[25] B. T. Lilly and K. K. Paliwal. *ROOT - An Object Oriented Data Analysis Framework.* 1997.

[26] *Computationnal centre of IN2P3.*

[27] et al The HESS Collaboration : F.Aharonian. Primary particle acceleration above 100 tev in the shell-type supernova remnant rx j1713.7-3946 with deep h.e.s.s. observations. *Astron.Astrophys.464 :235-243,2007*, 2006.

[28] The HESS Collaboration. Galactic center.

[29] S. Burke-Spolaor, J. Lazio, K. Nyland, L. Blecha, T. Bogdanovic, J. Comerford, X. Liu, G. Taylor, Y. Shen, T. J. Maccarone, L. Chomiuk, and A. Reines. Supermassive Black Hole Binaries : Multi-Messenger Astrophysics and Long Baselines with the Next-Generation Very Large Array. In *American Astronomical Society Meeting Abstracts 231*, volume 231 of *American Astronomical Society Meeting Abstracts*, page 250.26, January 2018.

[30] *Offical CTA data analysis, ctapipe.*

[31] CTA Collaboration. Cherenkov telescope array project.

[32] Patrick Carribault. Achitecture et optimisation de codes. 2014.

[33] Jeffrey Dean and Sanjay Ghemawat. Mapreduce : simplified data processing on large clusters. In *OSDI'04 : PROCEEDINGS OF THE 6TH CONFERENCE ON SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION*. USENIX Association, 2004.

[34] Apache Software Foundation. Hadoop.

[35] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark : Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[36] *Top 500 and Green 500.* 2016.

[37] Giovanni Lamanna. Astronomy ESFRI and Research Infrastructure Cluster.

[38] Proposal for cta raw data model and format. 2014.

[39] William Pence, Rob Seaman, Richard L. White. A tiled-table convention for compressing fits binary tables. 2010.

[40] Pierre Aubert. *PLIBS 8.*

[41] *Simplified Wrapper and Interface Generator (SWIG).*

[42] *Seamless operability between C++11 and Python (pybind11).*

[43] Jean Jacquemier. *Flow-based framework (flow).*

[44] *Library for messages communication.*

[45] *Data format and memory access patterns.*

[46] Karl Kosak. *DL0 interface document.*

[47] Github : plateforme de développement gratuite.

[48] Gitlab : plateforme de développement privée.

[49] Abraham Lempel and Jacob Ziv. Lempel–Ziv–Markov chain algorithm. 1996.

[50] Hierarchical data format, hdf5.

[51] T. Berghofer et al. Towards a Model for Computing in European Astroparticle Physics. 2015.

[52] P. J. Hall (ed). An ska engineering overview. *SKA Memorandum 91*, 2007.

[53] M. L. Ahnen et al. Data compression for the first g-apd cherenkov telescope. 2015.

[54] Jacob Ziv. A constrained-dictionary version of LZ78 asymptotically achieves the finite-state compressibility for any individual sequence. *CoRR*, abs/1409.1323, 2014.

[55] Julian Seward. Burrows–wheeler algorithm with huffman compression. 1996.

[56] Jean-loup Gailly and Mark Adler. Gnu zip. 1992.

[57] Zstandard. 2015.

[58] Y.M. Shtarkov F.M.J. Willems and T.J. Tjalkens. The context-tree weighting method : basic properties. *IEEE Transactions on Information Theory*, 41, 2002.

[59] Abraham Lempel and Jacob Ziv. Lempel–Ziv lossless data compression algorithms. 1977.

[60] Jan Platos and Jiri Dvorský. Word-based text compression. *CoRR*, abs/0804.3680, 2008.

[61] I. Clear and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Information Theory*, COM-32, 1984.

[62] CTA Consortium. Introducing the CTA concept. *Astroparticle Physics*, 43 :3 – 18, 2013.

[63] T. Hassan, L. Arrabito, K. Bernlör, J. Bregeon, J. Hinton, T. Jogler, G. Maier, A. Moralejo, F. Di Pierro, M. Wood, and f. t. CTA Consortium. Second large-scale Monte Carlo study for the Cherenkov Telescope Array. *ArXiv e-prints*, August 2015.

[64] CTA Consortium. CTA data management technical design report version 2.0. 2016.

[65] P. Aubert T. Vuillaume G. Maurin J. Jacquemier G. Lamanna N. Emad. Polynomial data compression for high scale physical experiments. 2018 (Accepted).

[66] *Maqao.*

[67] Wavelet denoising for gamma ray astronomy. 2018.

[68] Ideal Spatial Adaptation by Wavelet Shrinkage. 1993.

[69] A.M. Hillas. Cerenkov light images of eas produced by primary gamma. *Proc. of 19nd I.C.R.C. (La Jolla)*, 3 445, 1985.

[70] *Gnu C Compiler and Gnu C++ Compiler - GCC/G++.*

[71] Stellar Group. High performance parallex - hpx.

[72] Eigen.

[73] Intel. Math kernel library.

[74] Nukri Komin. *"Detection of Gamma Rays from the Supernova Remnant RX J0852.0-4622 with H.E.S.S."*.

[75] Least-Squares Intersection of Lines. 2013.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[77] Mars, the magic analysis and reconstruction software. *31st International Cosmic Ray Conference*, 2009.

[78] Mathieu de Naurois Loïc Rolland. A high performance likelihood reconstruction of gamma-rays for imaging atmospheric cherenkov telescopes. *Astroparticle physics, 32 :231-252*, 2009.

[79] Stéphane Le Bohec and Bernard Degrange. Contribution of the c.a.t. collaboration to the workshop "toward a major cherenkov detector iv". 1995.

[80] G. Navarra et al. Kascade-grande collaboration. *Nucl. Instr. Meth. A 518 207*, 2004.

[81] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math., 2 :164-168*, 1944.

[82] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math., 11 :431-441*, 1963.

[83] W. Gander. Algorithms for the QR-decomposition. Tech. Report 80–02, Angewandte Mathematik, ETH, Zürich, Switzerland, 1980.

Remerciments

Ce travail a été réalisé dans le cadre du projet Astronomy ESFRI and Research Infrastructure Cluster (ASTERICS) supporté par le programme Horizon 2020 de la commission Européenne sur la recherche et l'innovation sous le numéro de contrat 653477.

J'ai tant de personnes à remercier pour le bon déroulement de ce travail que j'espère n'oublier personne. Avant toute chose, je voudrai remercier Giovanni Lamanna qui m'a donné l'opportunité et le privilège de travailler dans le *Laboratoire d'Annecy de Physique des Particules* (LAPP). Je voudrai également remercier mes deux directeurs de thèse, Nahid Emad et Gilles Maurin, pour leur conseils avisés et tout ce que j'ai pu apprendre d'eux. Je remercie les membres de mon jury de thèse, d'avoir accepter d'y participer.

Je voudrai remercier le groupe CTA du LAPP, Jean Jacquemier pour nos longues discussion de conception (même si les tabulations sont mieux que les espaces) - Thomas Vuillaume, Florian Gaté et Gilles Maurin (encore) pour nos conversations de Physique saupoudrées d'astuces mathématiques - Armand Fiasson et David Sanchez qui ont tester la première version HPC de notre analyse de données Python pour CTA - Frederic Girardo qui m'a décrit de nouvelles technologies de bases de données et de stockage - Mickael Jacquemont et Alexandre Benoit pour leur bons conseils et pour m'avoir permis de parfaire ma connaissance du deep-learning - Vincent Poireau qui m'a permis de participer aux écoles ASTERICS (du 6 au 9 juin 2017 et du 4 au 8 juin 2018) et pour ses conseils avisés - Jayesh Whag qui m'a permis de relever mon niveau d'anglais, notamment sur la rédaction des publications.

Je voudrai remercier Volker Beckmann qui a supporté mon travail (et bien d'autres) avec la création du journal *Computing and Software for Big Sciences* et pour la création d'un projet transversal au sein du CNRS qui a pour but de relever les défis de calculs de toutes ces expériences. Ce projet a été conçu et concrétisé par David Chamont et Gilles Grasseau. Je voudrai aussi les remercier pour leur école sur les architectures hétérogènes et le calcul GPU et pour nos discussions d'optimisation avec également Hadrien Grasland et Vincent Lafage. Merci également à Arnaud Beck et son équipe de l'école polytechnique qui ont été les premiers utilisateurs extérieurs de ma librairie HPC et de ses générateurs de formats de données.

Puisque mes meilleures optimisations dépendent des mathématiques, je voudrai remercier Eric Pilon, Jean-Philippe Guillet, Pierre Salati, Luc Frappat, Laurent Gallot, Pascal Chardonnet, Patric Aurenche, Gérard Clément, Pasquale Serpico, Richard Taillet, Franck Thuillier, Jean Avant, Brian Zaldivar, Cédric Delaunais et Fawzi Boudjema pour leurs conseils et astuces qui m'ont permis de renforcer mes astuces mathématiques.

Merci à Nadine Nérou, Eric Fede, Mathieu Gauthier-Lafaye pour leur aide sur le centre de calcul MUST. Je voudrai remercier Fatih Bellachia pour l'intérêt qu'il a porté à mon algorithme de compression rapide et Sébastien Vilalte qui m'a proposé d'inclure cet algorithme dans la nouvelle version du détecteur Atlas.

Je voudrai également remercier Chantal Vallée et Brigitte Putanier qui m'ont aidé lorsque j'avais des requêtes administratives.

Je remercie plus généralement l'ensemble des doctorants du LAPP et du LAPTH ainsi que toutes les personnes qui y travaillent pour leur bonne humeur continuelle et chaleureuse qui m'a rendu ma thèse au LAPP particulièrement agréable.

Je tiens à remercier particulièrement Sébastien Vallat qui m'a appris à programmer en C++ et m'a transmit toute la rigueur nécessaire à cette discipline.

Enfin, je voudrai tout particulièrement remercier ma famille : ma mère, mon père, ma soeur, ma grand-mère. Merci beaucoup, sans vous, je ne serai pas aller aussi loin.