

Détecteur SDHCAL pour le signal $e^+e^- \longrightarrow \nu\nu h$: Optimisation et Adaptation de l'analyse de données pour le Projet FCC

Alexia HOCINE

Étudiante en M2 Physique SUBA à l'UCBL1

Supervisé par Gérard GRENIER

Maître de Conférence - UCBL1

Enseignant-Chercheur - IP2I, CNRS, IN2P3

Membre des collaborations CALICE, CMS et ILD

Corresponsable du groupe SDHCAL au sein de la collaboration CALICE



Rapport de Stage - Master 2 Physique SUBA
Université de Claude Bernard Lyon 1

Avril-Juillet 2022

Préambule

Remerciements

Je souhaite d'abord remercier Gérard GRENIER pour m'avoir donné la chance de montrer ce que je peux faire. Et aussi pour son encadrement, son accompagnement et son temps.

Je souhaite plus largement remercier mon équipe, Gérard GRENIER, Imad LAKTINEH et Clément DEVANNE, pour l'atmosphère positive, détendue et stimulante.

Et plus largement, les employés de l'IP2I pour leur gentillesse et leur accueil.

Participation à la *Geek and Japan Touch*

Au cours de mon stage, j'ai participé à l'atelier tenu par l'IP2I à la *Geek and Japon Touch*, organisé par Stéphanie BEAUCERON (IP2I, CNRS, CMS).

Durant ce week-end, avec 2 autres chercheuses (non physiciennes), Florence BOYER et Liliane DE ARAUJO, on a tenu un débat sur le film *Don't look up : Déni Cosmique* de Adam MCKAY, sur la crédibilité du discours scientifiques.

Ensuite sur le stand, j'ai pu expliquer les bases scientifiques et des recherches menées par le CNRS, CMS, et Virgo au près du grand public.

De plus, j'ai aussi animé le stand de l'association ?? dont l'objectif est d'expliquer les principes de base de la gravité en 2D avec un drap tenu.

Table des matières

1	Introduction	4
1.1	La Physique des Collisionneurs	4
1.1.1	Collisionneurs hadroniques	4
1.1.2	Collisionneurs leptoniques	4
1.2	Physique du boson de Higgs	4
1.2.1	Production du boson de Higgs	4
1.2.2	Détecteur	4
1.2.3	Collisions	5
1.3	Présentation & Objectif du Stage	5
2	Projet ILC	6
2.1	Présentation du Projet ILC	6
2.2	Programme : <code>original</code>	6
2.2.1	Données initiales	7
2.2.2	Conversion des fichiers initiaux en fichiers <code>ROOT</code>	7
2.2.3	Analyses des collisions	8
2.3	Programme : <code>ilcsoft</code>	10
2.3.1	Données	10
2.3.2	Programme : <code>processor</code>	10
2.3.3	Programme : <code>analysis</code>	10
3	Programme FCC	11
3.1	Présentation du Projet FCC	11
3.2	Développement Numérique	11
3.2.1	<code>convert</code>	11
3.2.2	<code>processor</code>	12
3.2.3	<code>analysis</code>	12
4	Outils Numériques	13
4.1	Répertoire <code>script</code>	13
4.2	Répertoire <code>test</code>	13
4.2.1	Programmes <code>testXxCompleted.py</code>	13
4.2.2	Programmes <code>testXxSame.py</code>	14
4.2.3	Répertoire <code>result</code>	14
4.2.4	Résultats	14
5	Résultats Physiques	15
5.1	Physique du Higgs	15
5.1.1	Section efficace	15
5.1.2	Nombre de Higgs attendu	15
5.1.3	Propriétés à Préciser	15
6	Conclusion	16
6.1	Travail effectué	16
6.2	Résultats attendus	16
6.3	Au delà	16

A	Résumé du travail effectué	17
A.1	Bibliographie	17
A.2	Tutoriels	17
A.3	Code initial	17
A.4	Code final	17
B	Organisation du Projet	18
B.1	Organisation initiale	18
B.2	Organisation finale	18
B.3	Le dossier NNH_HOME	18
C	Fichiers ROOT de sorties du programme processor	19

Chapitre 1

Introduction

1.1 La Physique des Collisionneurs

Le principe des collisionneurs est simple, on accélère des particules à des énergies cinétiques suffisantes pour provoquer des collisions inélastiques, et ainsi comprendre les interactions fondamentales et les constituants élémentaires de la matière.

On distingue 2 familles de collisionneurs en fonction des particules qui sont utilisés.

1.1.1 Collisionneurs hadroniques

Les collisionneurs hadroniques utilisent des hadrons, qui sont des particules complexes composées de 3 quarks et de gluons¹. Par exemple au LHC², on utilise des protons, composés de 2 quarks up et d'1 quark down.

Comme il s'agit de particules composites, ce sont pas les protons qui collisionnent directement mais ces constituants, appelés partons. Chacun porte une fraction indéterminée de l'énergie du proton. On ignore donc l'énergie de la collision en amont, il s'agit d'un paramètre libre.

C'est pourquoi, ils sont utiles pour la découverte de nouvelles particules de masse inconnue, puisqu'ils permettent de balayer tout le spectre de masse sous la gamme d'énergie du collisionneur (au LHC $< 14 \text{ TeV}$)³.

1.1.2 Collisionneurs leptoniques

En revanche, les collisionneurs leptoniques utilisent des leptons, qui sont des particules élémentaires. Comme le LEP⁴, qui collisionnait des électrons et des positrons [4].

Cette fois-ci, chaque lepton qui collisionne, possède une énergie complète donc connue. Puisqu'on leur impulse une énergie précise, ainsi on augmente la statistique pour un certain niveau d'énergie. Ces collisionneurs sont donc utilisés pour la recherche de précision.

Les prochaines générations de collisionneurs, comme ILC, CEPC, CLIC et FCC, ce sont des collisionneurs leptoniques. Leur objectif est de préciser les données déjà obtenues, notamment sur le boson de Higgs découvert en 2012 par le LHC⁵.

1.2 Physique du boson de Higgs

1.2.1 Production du boson de Higgs

Concrètement, on ne mesure pas directement le boson de Higgs mais ses produits de désintégrations sous la forme de jet. Ainsi on cherche à améliorer la résolutions en énergie de ces jets que l'on détecte [7].

1.2.2 Détecteur

En physique des particules, on utilise des détecteurs appelés calorimètres pour mesurer l'énergie des particules. Cette énergie va être déposer par ionisation dans le matériau le long de la trajectoire des particules qui

1. Gluon : boson médiateur de l'interaction forte qui maintiennent les quarks ensembles.

2. LHC : Large Hadron Collider, CERN

3. D'où l'intérêt de nouveaux collisionneurs à des énergies plus élevées et donc des masses de particules produites plus lourdes.

4. LEP : Large Electron-Positron, le prédécesseur du LHC, même tunnel

5. Higgs : était la pièce manquante du modèle standard des particules, car il permet aux particules d'acquérir une masse

le traverse. Il faut donc des algorithmes de reconstruction pour déduire les énergies, les types de particules et les trajectoires.

Pour cela, on utilise des calorimètres à grande granularité qui permet une très bonne performance des Algorithmes de Flux de Particules (PFA) [7].

C'est dans ce cadre que la collaboration internationale CALICE, à développer le premier prototype de la famille de calorimètre granulaire SDHCAL, pour Semi-Digital Hadronic CALorimeter, qui a été développé en grande partie à l'IP2I dans l'équipe CMS, auquel j'appartiens pour ce stage.

1.2.3 Collisions

Au cours, de ce stage, je me suis concentrée sur les collisions de type **nnh** pour neutrino-neutrino-higgs. Dont voici les diagrammes de Feynman :

FIGURE 1.1 – Diagramme de Feynmann de collision $e^+e^- \rightarrow \nu\nu h$

1.3 Présentation & Objectif du Stage

Pour ce stage, j'ai récupéré les codes de Guillaume GARILLOT, qui les a développés en 2021 au cours de son post-doctorat à l'IP2I. Ils sont en libre accès à l'adresse <https://github.com/ggarillot/nnhAnalysis/tree/refactor>.

Ce programme **nnhAnalysis** permet l'étude de fichiers **SLCIO** pour la collision :

$$e^-e^+ \rightarrow \nu\nu h \quad (1.1)$$

Et l'analyse des canaux de désintégration :

$$h \rightarrow WW^* \rightarrow qq\bar{q}\bar{q} \quad (1.2)$$

$$h \rightarrow b\bar{b} \quad (1.3)$$

Pour cela, il a utilisé les suites logicielles de **iLCSoft**, <https://github.com/iLCSoft> (plus précisément **LCIO** et **Marlin**), qui sont les anciennes suites logicielles. Mais les nouveaux projets de collisionneurs utiliseront **Key4HEP** et **Gaudi**.

Mon objectif est double. Dans un premier temps, comprendre et optimiser les codes existants, c'est-à-dire le programme **nnhAnalysis** de Guillaume GARILLOT qui utilise **LCIO**, **Marlin**. Puis, je vais transformer son programme pour qu'il puisse correspondre aux nouvelles normes des collisionneurs leptoniques, **Key4HEP** et **Gaudi**.

Chapitre 2

Projet ILC

2.1 Présentation du Projet ILC

Le projet ILC (International Linear Collider) est un collisionneur linéaire, électron-positron, de 31 km conçu pour atteindre une énergie de centre de masse de 500 GeV [3].

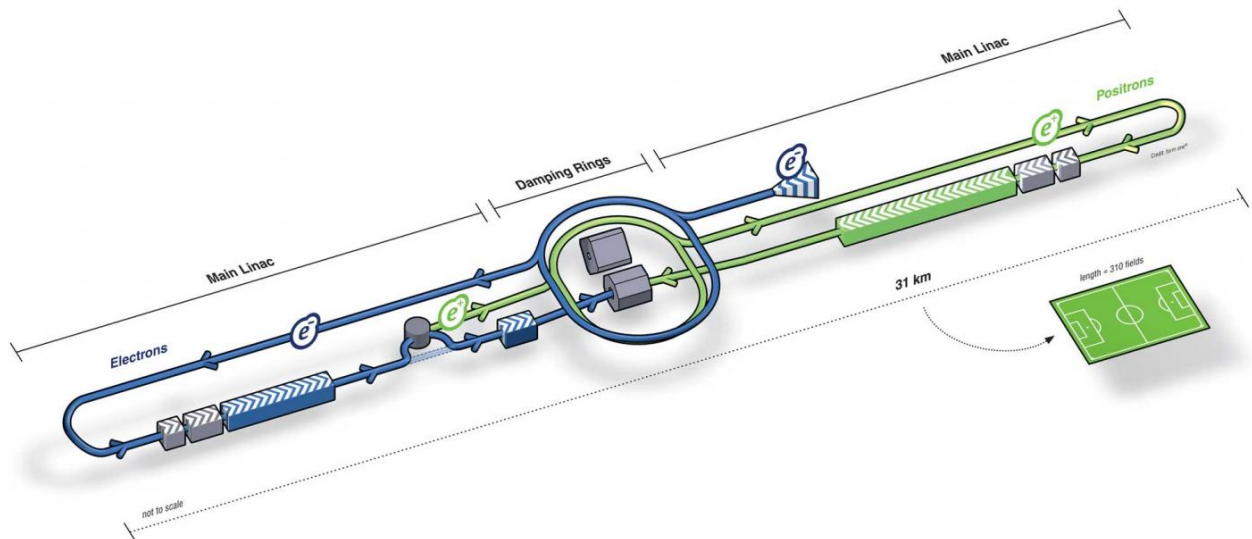


FIGURE 2.1 – Schéma ILC[3]

L'objectif de l'ILC est de produire beaucoup de boson de Higgs, notamment pour découvrir s'il y en a d'autres générations du boson de Higgs. Et plus globalement pour rechercher de la nouvelle physique, par de nouveaux écarts avec le Modèle Standard.

À l'heure actuelle, ce projet attend les autorisations pour lancer sa construction, probablement dans les montagnes au nord du Japon. Et le détecteur SDHCAL correspond parfaitement aux spécifications nécessaires pour un accélérateur linéaire de ce type. C'est pourquoi, afin de consolider sa candidature pour l'appel d'offre, l'IP2I développe aussi des programmes d'analyse de ce détecteur.

2.2 Programme : original

Au début de mon stage, j'ai récupéré les codes de Guillaume GARILLOT, qui les a développés en 2021 au cours de son post-doctorat à l'IP2I.

Son projet se divise en 3 programmes indépendants [FIGURE B.1] :

miniDSTMarker qui permet de récupérer les données¹ sur le serveur distant où elles sont stockées.

1. aujourd'hui simulées mais plus tard obtenues dans le détecteur

processor permet de tirer, des données brutes précédentes, des arbres ROOT s.

analysis utilise les méthodes des arbres binaires boostés pour effectuer une analyse statistique des arbres ROOT issus du programme **processor**.

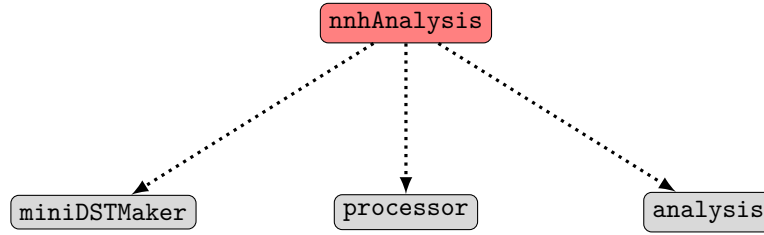


FIGURE 2.2 – Organisation des dossiers de mon Projet - <https://github.com/ggarillot/nnhAnalysis/tree/refactor>

En résumer, ce projet permet l'analyse des collisions survenues dans ce détecteur pour le projet ILC.

2.2.1 Données initiales

Malheureusement, pour le temps de ce stage, je n'ai pas obtenu l'accès au serveur, donc je n'ai pas pu utilisé **miniDSTMaker**. Je l'ai donc pas testé, ni pu développer ma propre version. C'est pourquoi, j'ai décidé de ne pas l'inclure dans mon propre code.

Pour que je puisse travailler, on a mis à ma disposition certains de quelqu'un de ces fichiers, ceux des collisions à 250 GeV. Soient 66 dossiers [FIGURE 2.3] de fichiers LCIO, où chaque nom correspond au code du type de processus [FIGURE ??].

```

/gridgroup/ilc/nnhAnalysisFiles/AHCA
(base) [ /gridgroup/ilc/nnhAnalysisFiles/AHCA ]$ ls
402001 402007 402013 500006 500066 500078 500090 500101 500107 500115 500122
402002 402008 402014 500008 500068 500080 500092 500102 500108 500116 500124
402003 402009 402173 500010 500070 500082 500094 500103 500110 500117 500125
402004 402010 402176 500012 500072 500084 500096 500104 500112 500118 500126
402005 402011 402182 500062 500074 500086 500098 500105 500113 500119 500127
402006 402012 402185 500064 500076 500088 500100 500106 500114 500120 500128
  
```

FIGURE 2.3 – Les noms des dossiers correspondent aux numéros de processus

Type de processus	Code des processus
2 leptoniques	500006, 500008
2 hadroniques	500010, 500012
4 hadroniques	500062, 500064, 500066, 500068, 500070, 500072
4 semi-leptoniques	500074, 500076, 500078, 500080, 500082, 500084, 500101, 500102, 500103, 500104, 500105, 500106, 500107, 500108, 500110, 500112
4 leptoniques	500086, 500088, 500090, 500092, 500094, 500096, 500098, 500100, 500113, 500114, 500115, 500116, 500117, 500118, 500119, 500120, 500122, 500124, 500125, 500126, 500127, 500128
signal	402007, 402173, 402176
autres higgs	402001, 402002, 402003, 402004, 402005, 402006, 402008, 402009, 402010, 402011, 402012, 402013, 402014, 402182, 402185, 402173, 402176

FIGURE 2.4 – Signification des codes des processus, si le signal recherché est de type $b\bar{b}$, si le signal est WW^*

2.2.2 Conversion des fichiers initiaux en fichiers ROOT

Grâce au programme **processor** on va pouvoir convertir les fichiers initiaux SLCIO en fichiers ROOT standards, afin de pouvoir les analyser.

On obtient ainsi pour chaque dossier de fichier de données SLCIO un fichier ROOT en sortie, c'est-à-dire que l'on obtiendra un arbre ROOT par type de processus.

Ce programme doit être robuste et donc à partir des mêmes fichiers d’entrées toujours générer des fichiers ROOT strictement identiques.

2.2.3 Analyses des collisions

Étape 1

Avant de commencer l’analyse des fichiers ROOT générés précédemment, on va terminer ce que le programme `processor` avait commencé, et fusionner l’intégralité de ces fichiers en un seul gros fichier `DATA.root`, grâce à la commande `hadd`. Cette commande a été développée par le CERN et elle fusionne tous les histogrammes de différents fichiers en un seul[2].

Donc là encore, la répétition de ce programme doit toujours générer des fichiers `DATA.root` strictement identique.

Étape 2

Pour la deuxième étape de ce programme `analysis`, on va trier nos données en 4 fichiers distincts. Mais au lieu de les séparer par leur numéro de processus, qui est un critère numérique donc éloigné de la réalité des résultats d’une véritable expérience, on va les diviser suivant la polarisation initiale des particules incidentes, c’est-à-dire l’électron et le positron à -0,8 et 0,3 ou les 2 nulles. Et par le type de particules produites par un boson de Higgs, c’est-à-dire les canaux $b\bar{b}$ et WW^* , puisque ce sont les plus probables².

Decay mode	Branching ratio (%)	relative uncertainty (%)
$H \rightarrow b\bar{b}$	57.7	+3.2, -3.3
$H \rightarrow c\bar{c}$	2.91	+12, -12
$H \rightarrow \tau^-\tau^+$	6.32	+5.7, -5.7
$H \rightarrow \mu^-\mu^+$	2.19×10^{-2}	+6.0, -5.9
$H \rightarrow WW^*$	21.5	+4.3, -4.2
$H \rightarrow ZZ^*$	2.64	+4.3, -4.2
$H \rightarrow \gamma\gamma$	0.228	+5.0, -4.9
$H \rightarrow Z\gamma$	0.153	+9.0, -8.8
$H \rightarrow gg$	8.57	+10, -10

Table 7.2: The Higgs boson decay modes and their corresponding branching ratios under the SM prediction of Higgs boson mass of 125 GeV. The uncertainty contributed by both theoretical and parametric sources [20].

FIGURE 2.5 – Les modes de désintégrations du boson de Higgs à 125 GeV [7]

Donc on obtient ces 4 fichiers suivants :

Polarisation	Canal	
(e, p)	$b\bar{b}$	WW^*
$(0, 0)$	<code>split_bb_e+0_p+0.root</code>	<code>split_ww_e+0_p+0.root</code>
$(-0.8, +0.3)$	<code>split_bb_e-0.8_p+0.3.root</code>	<code>split_ww_e-0.8_p+0.3.root</code>

FIGURE 2.6 – Tous les événements sont séparés en 4 fichiers suivant la polarisation des particules incidentes et des types des particules produites par le boson de Higgs.

Dans chacun de ces fichiers les arbres ROOT, `TTree`, auront les variables suivantes :

isSignal : booléen qui indique si on considère l’évènement comme du signal ou du bruit de fond.

channelType : entier qui représente le type de canaux de l’évènement [FIGURE 2.4] :

- 2 fermions leptoniques ou hadroniques
- 4 fermions leptoniques ou semi-leptoniques ou hadroniques

2. Rien n’empêchera d’ajouter d’autres canaux même si le code manque encore de un peu de modularité.

— une autre type de résultat avec un boson de Higgs

isTrain booléen qui confirme si l'évènement a été entraîné ou testé par la BDT.

preSelected booléen, si l'évènement a été pré-sélectionné par la BDT.

weight flottant qui est le poids de l'évènement en fb^{-1} de la luminosité intégrée.

Pour trier les données du fichier `DATA.root`, on va utiliser des BDT pour *Boosted Decision Tree*, en français, des arbres de décision boostés.

Les arbres sont des structures de données courantes en informatique, ils sont utilisés notamment pour trier les données car ils ont une rapidité optimale en temps. Ici on utilise les arbres pour déterminer la nature des particules en répondant à des questions booléennes (Similaire à l'exemple [FIGURE 2.7]. Et comme il n'y a que 2 issus à chaque nœud, on parle d'arbre binaire.

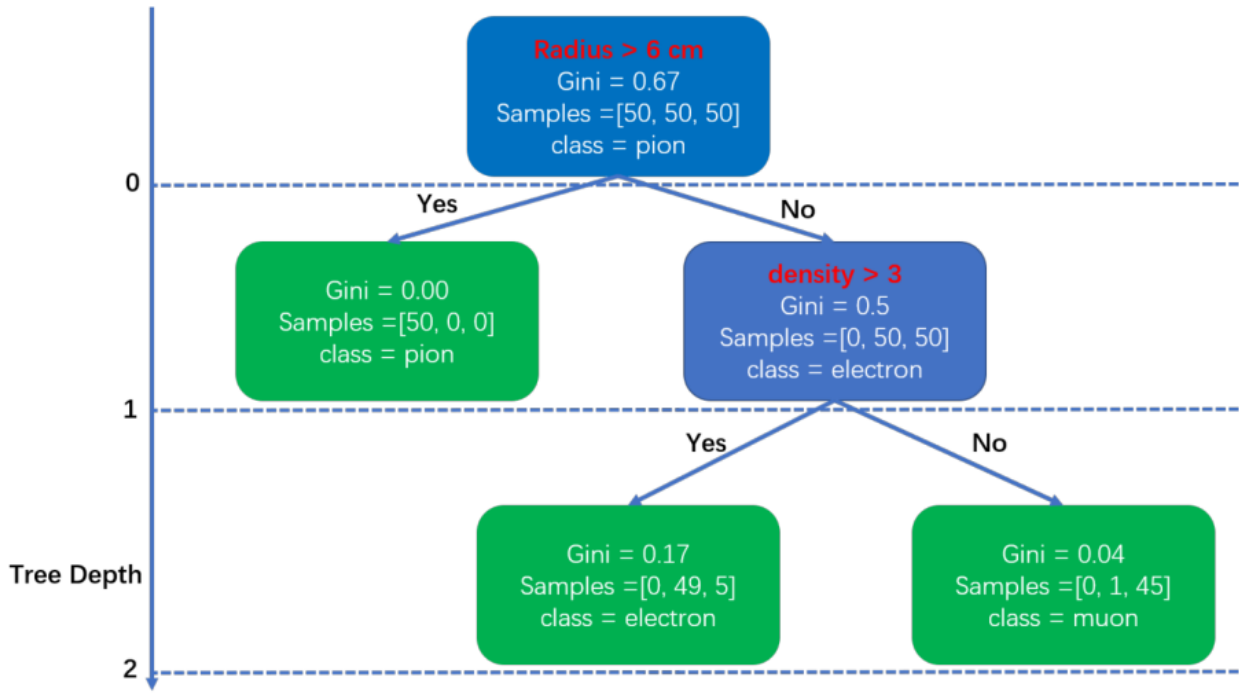


FIGURE 2.7 – Exemple de BDT [7]. Ici, elle permet de déterminer de le type d'une particule

Une fois encore, des méthodes ont déjà été développées dans l'API du CERN[1]. Ces méthodes utilisent des paramètres générés aléatoirement afin d'augmenter la rapidité des calculs. C'est pourquoi on parle d'arbre boosté.

Mais cette fois-ci les fichiers créés sont équivalents mais pas identiques. Car les BDT utilisent la génération de nombre aléatoire, ce qui engendre des variations dans les entraînements.

Étape 3

Et pour terminer, on peut enfin effectuer l'analyse à proprement parlé de nos données à partir des fichiers `split_XX.root` précédents [FIGURE ??].

Cette étude se fait sur les fichiers avec une polarisation non nulle pour les particules incidentes. Car ?????

Polarisation	Canal	
	$b\bar{b}$	WW^*
$(-0.8, +0.3)$	<code>model_bb_e-0.8_p+0.3.joblib</code>	<code>model_ww_e-0.8_p+0.3.joblib</code>

FIGURE 2.8 – Fichiers du model de la BDT.

Et même si cette analyse sera toujours la même quelque soit les fichiers `split_XX.root`, ces derniers étant légèrement différents d'un entraînement à l'autre les résultats statistiques [FIGURE ??] seront, là encore, légèrement différents mais doivent rester équivalents.

Polarisation	Canal	
(e, p)	$b\bar{b}$	WW^*
$(0, 0)$	<code>scores_bb_e-0.8_p+0.3.root</code>	<code>scores_ww_e-0.8_p+0.3.root</code>

FIGURE 2.9 – Les fichiers scores ont la réponse de la BDT : 2 variables pour chaque évènement, un booléen pour savoir s'il est sélectionné et la valeur retournée par la BDT.

Polarisation	Canal	
(e, p)	$b\bar{b}$	WW^*
$(-0.8, +0.3)$	<code>bestSelection_bb_e-0.8_p+0.3.root</code>	<code>bestSelection_ww_e-0.8_p+0.3.root</code>

FIGURE 2.10 – Ce fichier ROOT contient tous les arbres TTree sélectionnés par la BDT comme un évènement de `nnh`

Polarisation	Canal	
(e, p)	$b\bar{b}$	WW^*
$(-0.8, +0.3)$	<code>stats_bb_e-0.8_p+0.3.joblib</code>	<code>stats_ww_e-0.8_p+0.3.joblib</code>

FIGURE 2.11 – Fichiers des résultats statistiques pour les polarisations des particules incidentes à 0,8 pour l'électron et à 0,3 pour le positron.

2.3 Programme : `ilcsoft`

À présent, je vais vous présenter les modifications que j'ai apporté au programme `original`.

Dans un premier temps, j'ai fais de la cosmétique : j'ai typé les variables, appliqué les bonnes règles de typographie et commenté les différents fichiers, classes, interfaces, fonctions, et variables globales. Ce qui m'a permis de bien comprendre les programmes et de le rendre plus lisible pour les futurs développeurs de ce code.

Ensuite je les ai modifié pour qu'il puisse s'exécuter en parallèle. En effet, `processor` et `analysis` mettent quelques heures à s'exécuter. Et pour tester leurs robustesses et pouvoir faire des statistiques, il est nécessaire de le faire de très nombreuses fois. Donc cette nouvelle fonctionnalité la version `ilcsoft` permet un gain de temps considérable. Et plus tard, cela permettra aussi d'exécuter avec différents jeux de données (niveaux d'énergies, canaux...).

2.3.1 Données

Donc bien sûr, il s'agit des mêmes fichiers `SLCIO` que pour le programme `original` [FIGURE 2.3]

2.3.2 Programme : `processor`

Le but est toujours la conversion de chaque dossier de fichiers `SLCIO` en un seul fichier `ROOT`.

Pour aider à la lisibilité du programme, j'ai aussi développé une nouvelle classe qui permet de simplifier sa lecture. Ma classe `PDGInfo` permet de manipuler plus facilement les outils du PDG, grâce à une correspondance entre les codes entiers des particules et leurs noms, ainsi que quelques méthodes et tests booléennes³.

2.3.3 Programme : `analysis`

La encore le fonctionnement global reste inchangé, par rapport à `original`

3. <https://github.com/alexhxia/nnhAnalysis/blob/main/nnhProgram/ilcsoft/processor/include/PDGInfo.hh>

Chapitre 3

Programme FCC

3.1 Présentation du Projet FCC

Le FCC (Futur Collisionneur Circulaire) est le projet du CERN pour remplacer leur collisionneur actuelle, le LHC (Large Hadronic Collider). Dont la fin de l'exploitation est prévu en 2040 [5]. On prévoit un anneau de 100 km, contre 27 km pour le LEP et le LHC (comme montrer Figure 3.1). Ce qui devrait nous permettra d'atteindre une énergie de 100 TeV contre 13 TeV actuellement pour le LHC.

L'objectif est la recherche d'une nouvelle physique par la mise en évidence de déviation avec le modèle standard. Et plus particulièrement, en augmentant la statistique sur le boson de Higgs, découvert avec le collisionneur actuelle, afin de mieux comprendre sa physique.

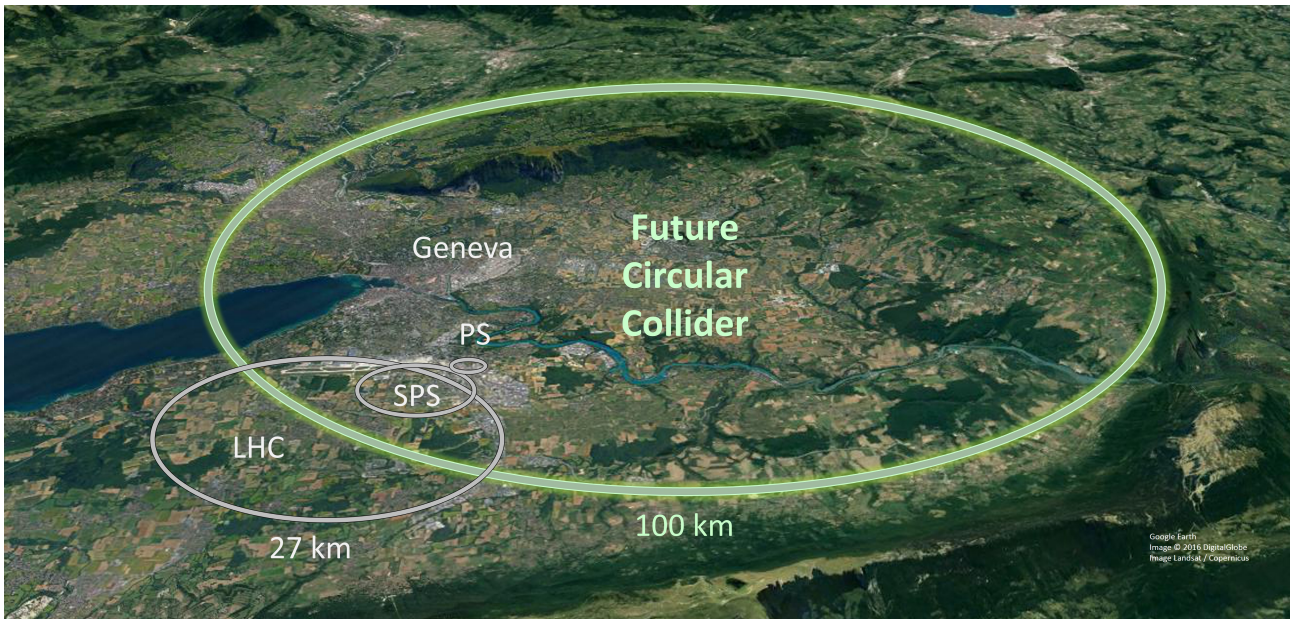


FIGURE 3.1 – <https://cds.cern.ch/images/OPEN-PHO-ACCEL-2019-001-2>

Une fois encore, le détecteur SDHCAL candidate pour ce projet et pourrait être y être installer.

3.2 Développement Numérique

Mon objectif est d'adapter les codes développés précédemment par Guillaume GARILLOT du projet ILC au projet FCC qui n'utilise pas les mêmes suites logiciels.

3.2.1 convert

Dans un premier temps, je dois convertir les fichiers SLCIO en fichiers ROOT en utilisant le programme libre k4MarlinWrapper du projet de key4hep¹. Il s'agit d'une collaboration entre chercheurs du CERN dont l'objec-

1. <https://github.com/key4hep/k4MarlinWrapper>

tif est de développer des outils pour le projet FCC, y compris des utils de conversion entre `LCIO` (`iLCSoft`) et `EDM4hep` (FCC).

Cette partie a été laborieuse, car le programme `k4MarlinWrapper` est en cours de développement. J'ai du faire remonter de très nombreux problèmes en parallèle de mon travail. En effet, à chaque mise à jour, le programme ne fonctionnait plus, ce qui m'a fait perdre beaucoup de temps en essayant de comprendre le problème, le signaler et le corriger quand je le comprenais, ou patienter pour que quelqu'un d'autre le corrige.

3.2.2 processor

Comme les fichiers d'entrées sont à présent des fichiers `ROOT`, il faut adapter cette partie pour que les fichiers de sortie soit les mêmes ou au moins équivalent.

3.2.3 analysis

Cette partie reste inchangé par rapport à `ilcsoft`.

Chapitre 4

Outils Numériques

Pour utiliser et comparer les résultats obtenus facilement, j'ai développé de très nombreux programmes supplémentaires afin de pouvoir automatiser leur utilisation, mais aussi de les exécuter sur un serveur distant.

4.1 Répertoire script

Dans ce dossier¹, j'ai développé des outils d'automatisation de l'exécution des programmes. En effet, avant pour lancer un **processor** ou une **analysis** il y avait de très nombreuses commandes à taper au terminal et parfois incompatible entre elles. J'ai donc développé 6 programmes principaux d'automatisation des programmes :

nnh programme générale qui permet l'exécution de tout le programme (**convert**, **processor** et **analysis**) et plusieurs fois.

nnhConvert permet de convertir les fichiers SLCIO en fichiers ROOT.

nnhProcessor d'exécuter tout le programme **processor**.

nnhAnalysis exécute tout le programme **analysis**.

prepareBDT exécute la préparation de la BDT.

launchBDT lance la BDT².

Chacun de ces programmes peuvent être lancé ensemble mais aussi séparément. Ce qui permet de gagner énormément en temps d'utilisation mais aussi de pouvoir personnaliser selon les besoins de l'utilisateur.

4.2 Répertoire test

Pour tester les résultats obtenus, j'ai développé 4 programmes en **python**. Et de tester s'ils sont compatibles entre eux.

•	Processus	Analysis
Completed	testProcessorCompleted.py	testAnalysisCompleted.py
Same	testProcessorSame.py	testAnalysisSame.py

FIGURE 4.1 – Tableau récapitulatif des fonctions de tests

4.2.1 Programmes testXxCompleted.py

L'objectif de ce type de programme est de tester si tous les fichiers qui aurait du être créer, l'ont bien été. Ces programmes sont très rapides, ne prennent que en quelques secondes.

Ce qui permet, en cas de problèmes de relancer juste la partie qui n'est pas terminé et pas tout le programme.

Programmes testProcessorCompleted.py

Un **processor** est complet si tous les dossiers du répertoire d'entrée ([FIGURE 2.3]) ont bien généré un fichier ROOT.

1. <https://github.com/alexhxia/nnhAnalysis/tree/main/script>

2. Le programme **launchBDT** est incompatible avec les autres, ce qui nécessite de le lancer dans un terminal enfant.

Programmes `testAnalysisCompleted.py`

Une `analysis` est complète si elle contient 13 fichiers, car :

`hadd` : 1 fichier `data.root`

`prepareBDT` : 4 fichiers (2 polarisations \times 2 canaux) `split_XX.root`

`launchBDT` : 2 fichiers (2 canaux) `model_XX.joblib`, `scores_XX.root`, `bestSelection_XX.root`, `stats_XX.json`

4.2.2 Programmes `testXxSame.py`

Ce type de programme³ va prendre quelques dizaines de minutes à une heure pour s'exécuter, car il va comparer tous les arbres `ROOT` des différents fichiers pour s'assurer que 2 fichiers sont identiques ou au moins compatibles. Cette comparaison se ferait avec la fonction de Kolmogorov présente, là encore, dans l'API du CERN, qui compare 2 histogrammes et retourne un flottant sur leur compatibilité. Dans le chapitre `original`, j'ai explicité que les programmes qui donnaient des résultats toujours identiques et ceux qui devaient avoir des résultats équivalents.

Programmes `testProcessorSame.py`

Toutes les exécutions de `processor` doivent donner des résultats identiques, surtout au sein du même projet. Et j'ai pu le confirmer en comparant plusieurs résultats obtenus de différentes exécutions.

Et lorsque j'ai apporté des corrections au programme, quand je suis passée de `original` à `ilcsoft`, cela m'a permis de vérifier que mon code obtenait bien le même résultat. De même, de `ilcsoft` à `fcc`.

Programmes `testAnalysisSame.py`

La comparaison entre 2 exécutions du programme `analysis` est plus compliquée, puisque la BDT utilise des nombres aléatoires, ce qui ne permet pas d'obtenir des résultats identiques.

Certains fichiers doivent rester identiques comme les fichiers `data.root` ou `split_XX.root`. Certains sont complètement différents comme les fichiers `model_XX.joblib` ou `bestSelection_XX.root`, car ils sont liés à l'entraînement de la BDT. Mais à la fin les fichiers `stat_XX.json` doivent être statistiquement compatibles.

4.2.3 Répertoire `result`

Dans ce répertoire, je place les fichiers de sorti des programmes `test`. En effet, quand on exécute un programme de test, l'utilisateur a la possibilité d'obtenir les résultats sous la forme d'un fichier `JSON`. Ce qui permet d'en garder une trace et de ne pas refaire un test déjà effectué.

4.2.4 Mes résultats

Je peux donc confirmer que mes programmes s'exécutent complètement et correctement.

3. <https://github.com/alexhxia/nnhAnalysis/tree/main/test>

Chapitre 5

Résultats Physiques

5.1 Physique du Higgs

5.1.1 Section efficace

Fusion WW^* [6] Comme on est à très haute énergie, on peut approximer que $\sqrt{s} \gg 2$

$$\sigma_{WW \text{ fusion}} \longrightarrow \frac{g_{HWW}^2 G_F^2}{32 \pi^3} \left[\left(1 + \frac{m_H^2}{s} \right) \log \left(\frac{s}{m_H^2} \right) - 2 \left(1 - \frac{m_H^2}{s} \right) \right] \quad (5.1)$$

g_{HWW} : couplage du boson de Higgs et du boson W

G_F : constante de couplage de Fermi

m_H : masse du boson de Higgs

s

5.1.2 Nombre de Higgs attendu

5.1.3 Propriétés à Préciser

Chapitre 6

Conclusion

6.1 Travail effectué

6.2 Résultats attendus

6.3 Au delà

Annexe A

Résumé du travail effectué

A.1 Bibliographie

<https://tel.archives-ouvertes.fr/tel-03405418>

- Étude du calorimètre hadronique semi-digital et étude du canal physique

$$e^-e^+ \longrightarrow \nu\nu h \ (H \longrightarrow WW \longrightarrow qqqq)$$

au collisionneur circulaire électron positon (CEPC)

- Bing Liu, IP2I
- 2020

<https://tel.archives-ouvertes.fr/tel-02141420>

- Étude des gerbes hadroniques dans un calorimètre à grande granularité et étude du canal

$$e^-e^+ \longrightarrow HZ \ (Z \longrightarrow qq)$$

dans les futurs collisionneurs leptoniques

- Guillaume Garillot, IPNL
- 2019

A.2 Tutoriels

LCIO <https://github.com/iLCSoft/LCIO>

ILDConfig <https://github.com/iLCSoft/ILDConfig>

Marlin <https://github.com/iLCSoft/Marlin>

key4hep <https://github.com/key4hep/k4MarlinWrapper>

A.3 Code initial

<https://github.com/ggarillot/nnhAnalysis/tree/refactor>

A.4 Code final

<https://github.com/alexhxia/nnhAnalysis>

Annexe B

Organisation du Projet

B.1 Organisation initiale

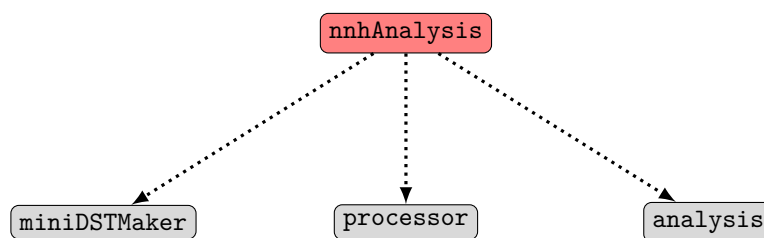


FIGURE B.1 – Organisation des dossiers de mon Projet - <https://github.com/ggarillot/nnhAnalysis/tree/refactor>

B.2 Organisation finale

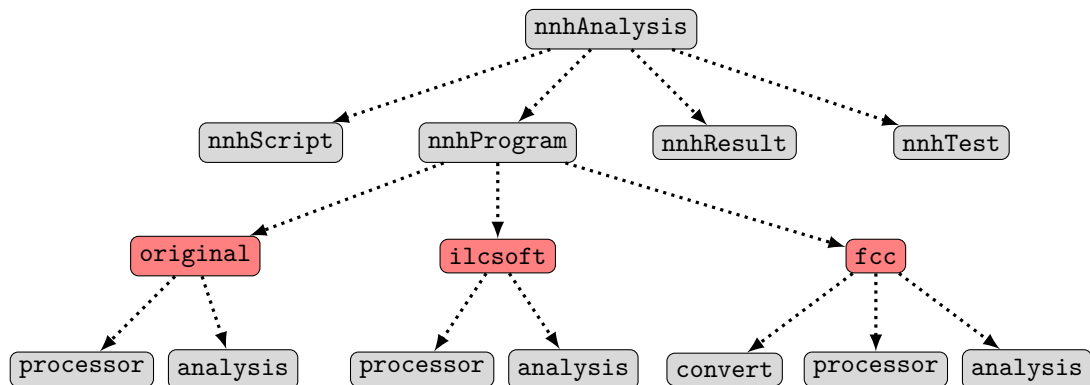


FIGURE B.2 – Organisation des dossiers de mon Projet - <https://github.com/alexhxia/nnhAnalysis>

B.3 Le dossier NNH_HOME

Pour s'exécuter, le projet a besoin de la variable d'environnement NNH_HOME qui est le chemin du programme que vous souhaitez exécuter, mis en avant en rouge dans les Figure ?? et Figure B.2.

Donc dans le projet initial, il s'agissait de NNH_HOME=\nnhAnalysis et dans le nouveau projet :

- NNH_HOME = \nnhAnalysis\nnhProgram\original
- NNH_HOME = \nnhAnalysis\nnhProgram\ilcsoft
- NNH_HOME = \nnhAnalysis\nnhProgram\fcc

Annexe C

Fichiers ROOT de sorties du programme processor

On a vu précédemment que `processor` converti une série de fichier `SLCIO` en fichiers `ROOT`.
Il considère deux canaux de désintégration du Higgs :

$$h \longrightarrow b\bar{b} \tag{C.1}$$

$$h \longrightarrow WW^* \longrightarrow qqqq \tag{C.2}$$

La première équation (C.1) sera mesuré par le détecteur sous la forme de 2 jets reconstruit et identifié comme 2 quark `b`.

Alors que la seconde équation (C.2) sera reconstruit comme 4 jets, 2 par boson `W`¹.

1. `W` sur couche de masse, `Wstar` hors couche de masse

Table des figures

1.1	Diagramme de Feynmann de collision $e^+e^- \rightarrow \nu\nu h$	5
2.1	Schéma ILC[3]	6
2.2	Organisation des dossiers de mon Projet - https://github.com/ggarillot/nnhAnalysis/tree/refactor	7
2.3	Les noms des dossiers correspondent aux numéros de processus	7
2.4	Signification des codes des processus, si le signal recherché est de type $b\bar{b}$, si le signal est WW^*	7
2.5	Les modes de désintégrations du boson de Higgs à 125 GeV [7]	8
2.6	Tous les évènements sont séparés en 4 fichiers suivant la polarisation des particules incidentes et des types des particules produites par le boson de Higgs.	8
2.7	Exemple de BDT [7]. Ici, elle permet de déterminer de le type d'une particule	9
2.8	Fichiers du model de la BDT.	9
2.9	Les fichiers scores ont la réponse de la BDT : 2 variables pour chaque évènement, un booléen pour savoir s'il est sélectionné et la valeur retournée par la BDT.	10
2.10	Ce fichier ROOT contient tous les arbres TTree sélectionnés par la BDT comme un évènement de nnh	10
2.11	Fichiers des résultats statistiques pour les polarisations des particules incidentes à 0,8 pour l'électron et à 0,3 pour le positron.	10
3.1	https://cds.cern.ch/images/OPEN-PHO-ACCEL-2019-001-2	11
4.1	Tableau récapitulatif des fonctions de tests	13
B.1	Organisation des dossiers de mon Projet - https://github.com/ggarillot/nnhAnalysis/tree/refactor	18
B.2	Organisation des dossiers de mon Projet - https://github.com/alexhxia/nnhAnalysis	18

Bibliographie

- [1] Rene Brun. treefriend.c file reference.
- [2] Rene Brun, Dirk Geppert, Sven A. Schmidt, and Toby Burnett. hadd.cxx file reference.
- [3] CERN. International linear collider ready for construction, jun 2013. <https://home.web.cern.ch/news/news/accelerators/international-linear-collider-ready-construction>.
- [4] CERN. The large electron-positron collider, juin 2022. <https://home.cern/science/accelerators/large-electron-positron-collider>.
- [5] CERN. Le futur collisionneur circulaire, jun 2022. <https://home.cern/fr/science/accelerators/future-circular-collider>.
- [6] Klaus Desch and Meyer Niels. Study of higgs boson production through ww-fusion at tesla, Février 2001.
- [7] Bing Liu. *Etude du calorimètre hadronique semi-digital et étude du canal physique $e^+e^- \rightarrow H \rightarrow WW \rightarrow qq\bar{q}\bar{q}$ au collisionneur circulaire electron positon (CEPC)*. Theses, Université de Lyon ; Shanghai Jiao Tong University, November 2020.