

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

Superviseur : Gérald Grenier (IP2I équipe CMS, FCC, SDHCAL)

Alexia HOCINE  
Étudiante en Master 2 en Physique SUBAtomique

Université de Claude Bernard Lyon 1

1<sup>er</sup> Juillet 2022



# Sommaire

## 1 Introduction

Processus étudiés

Projets de Déetecteurs

DéTECTEUR SDHCAL

## 2 Programmes

original

iLCSoft

FCC

## 3 Outils Numériques

Script

Test

## 4 Conclusion

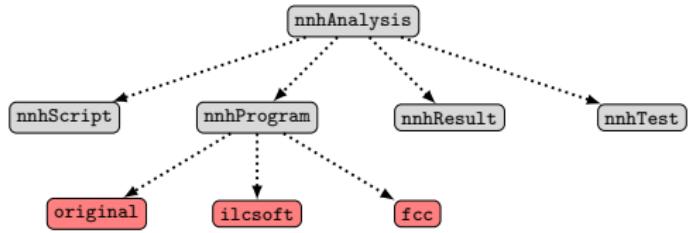


Figure 1 : Organisation des dossiers de mon Projet  
- <https://github.com/alexhxia/nnhAnalysis>

## Introduction



$e^+ e^-$  collisionneur leptonique  
 $\nu\nu h$  2 neutrinos-higgs

Canaux analysés

- ①  $h \longrightarrow WW \longrightarrow qqqq$
- ②  $h \longrightarrow b\bar{b}$

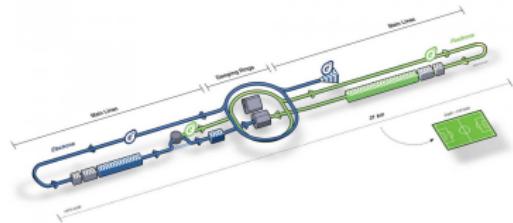
Mesures

- ① 4 jets
- ② 2 jets

# Projets iLC et FCC

## Des fabriques à bosons de Higgs

### ILC - International Linear Collider (Japon)



- Collisionneur linéaire
- 31 km (agrandissable)
- 500 GeV en centre de masse

### FCC - Future Circular Collider (CERN)



- Collisionneur circulaire
- 100 km
- 100 TeV dans le centre de masse

# Détecteur SDHCAL

## SDHCAL

- Collaboration internationale CALICE
- Semi-Digital Hadronic CALOrimeter
- Calorimètres à grande granularité
  - Mesure d'énergie
  - Mesure de trajectoire
- Très bonne performance des Algorithmes de Flux de Particules
- Candidat pour les futurs collisionneurs leptoniques
- Septembre nouvelle phase de tests en faisceaux

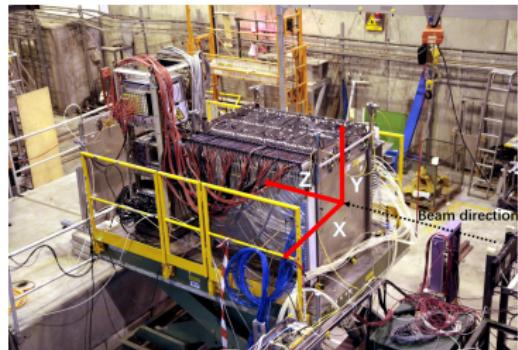


Figure 2 : Détecteur SDHCAL

original

<https://github.com/alexhxia/nnhAnalysis/tree/main/nnhProgram/original>

## Projet initial

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

miniDSTMaker

Télécharge du server  
lyogrid06 les fichiers  
DST de DESY : .lcio

processor

Transforme les fichiers  
.lcio en .root par type  
de processus

analysis

Entraîne une BDT, pour  
obtenir l'analyse  
statistique des  
événements

## Type de processus

```
/gridgroup/ilc/nnhAnalysisFiles/AHCAL
(base) [1]: AHCAL]$ 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
```

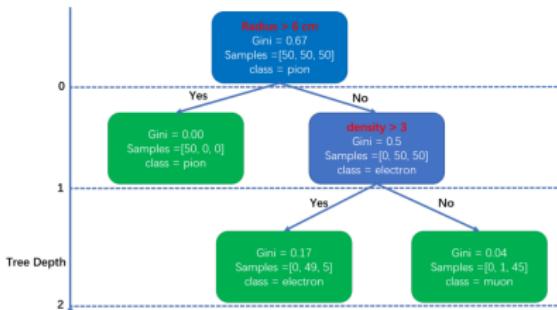
## Principe d'une BDT

### Boosted Decision Tree

- Arbres de décision boostés
- Machine Learning - apprentissage automatique
- Création d'un modèle, ici pour déterminer le type de processus

Pour 2 exécutions à partir des mêmes données

- fichiers de sortie ne sont pas identiques
- mais doivent être équivalents



## Améliorations apporter à processor et analysis

- Réécritures minimes (typographie, typage auto)
- Modification des noms de certaines fonctions
- Ajouts de commentaires (clarification des contrats)
- Nouvelle classe pour simplifier l'utilisation des codes PDG : PDGInfo.XX  
XX = {hh, cc}
- Réorganisation de la gestion des fichiers des sortis pour permettre l'exécution en parallèle (en cours)

## miniDSTMaker

- Non pertinent pour ce stage, puisque les données sont locales

## Ajout du programme convert (en cours)

- Tranforme les fichiers .lcio, exploitable par iLCSoft, en fichier exploitable par FCC.
- De la suite logiciel LCIO vers EDM4hep

## processor (en cours)

- Change toutes les utilisations de la suite logicielle d'iLCSoft vers key4HEP

## analysis

- Ne demande aucune nouvelle modification

## Outils de Numérique : Script

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

### Liste de nouveaux scripts

`nnh` programme général

- permet de choisir :

- combien de programme processus et analysis on souhaite

`nnhProcessor` lance un programme processus complet

`nnhAnalysis` lance un programme analysis complet

`prepareBDT` lance le programme

`prepareBDT`

`launchBDT` lance le programme `launchBDT`

## Outils de Numérique : Test

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

### Programme de tests : `testXxYy.py`

- Teste grâce à la fonction de Kolmogorov - développé par ROOT (CERN)
- Teste les fichiers de sorties :
  - des programmes Xx = {processus, analysis}
  - de type Yy = {Completed, Same}

•	Processus	Analysis
Completed	<code>testProcessorCompleted.py</code>	<code>testAnalysisCompleted.py</code>
Same	<code>testProcessorSame.py</code>	<code>testAnalysisSame.py</code>

`Completed` teste si tous les fichiers ont bien été générés

`Same` teste les différences entre 2 séries de fichiers

`processus` tous les fichiers sont sensés être identiques  
`analysis` certains les fichiers sont sensés :

- être identiques : `DATA.root`, données initiales
- d'autres différents : `model_[...].root`
- d'autres équivalents : `stats_[...].root`

# Conclusion

## Travaux réalisés

- Optimisation des codes pour iLCSoft processor
- Automatisation des programmes
- Programmation de codes de test

## Travaux en cours

- Optimisation des codes pour iLCSoft analysis
- Adaptation au projet FCC

## Compétences renforcées

- Programmation
  - C++
  - ROOT
  - Python
  - Script bash
- Physique
  - Physique des Particules
  - Modèle Standard
  - Statistiques

## Nouvelles compétences

- Utilisation de BDT

## Sources des Figures

## Bibliographie

# Suppléments

## Annexes