

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



## 1 Introduction

- Processus étudiés
- Détecteur SDHCAL
- Projets de Détecteurs

## 2 Programmes

- original
- iLCSoft
- FCC

## 3 Outils Numériques

- Script
- Test

## 4 Conclusion

- Premiers résultats
- État actuelle

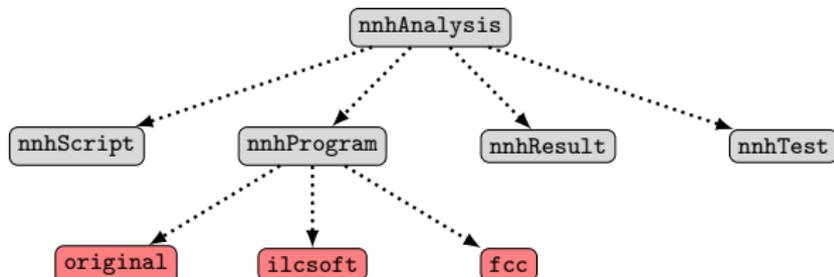


Figure 1 – Organisation des dossiers de mon Projet - <https://github.com/alexhxia/nnhAnalysis>

## Introduction

$$e^+e^- \longrightarrow \nu\nu h$$

$e^+e^-$  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

## Décteur 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
- En cours de développement : exploitation du temps à 100 pico-seconde

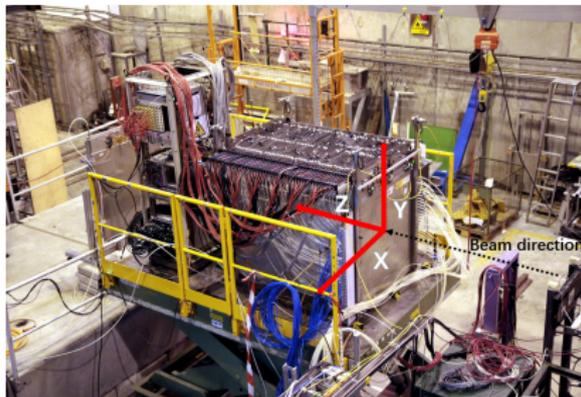
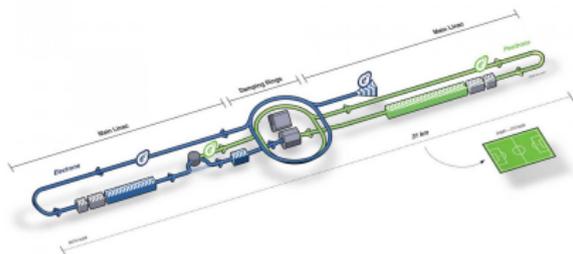


Figure 2 – Décteur SDHCAL

## 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
- Suites logicielles : iLCSoft avec Marlin

## FCC-ee - Future Circular Collider (CERN)



- Collisionneur circulaire
- 100 km
- 140 GeV dans le centre de masse
- Suites logicielles : Key4Hep avec Gaudi

## Sommaire

## ① Introduction

Processus étudiés  
Détecteur SDHCAL  
Projets de Détecteurs

## ② Programmes

original  
iLCSoft  
FCC

## ③ Outils Numériques

Script  
Test

## ④ Conclusion

Premiers résultats  
État actuelle

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

Entraine une BDT, pour  
obtenir l'analyse statistique  
des évènements

## Type de processus

```

/gridgroup/ilc/nnhAnalysisFiles/AHCAL
(base) [redacted] 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

```

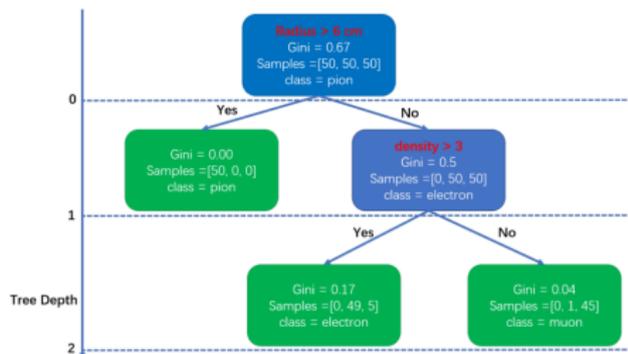
## Principe d'une BDT

## 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 minimales (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 sorties pour permettre l'exécution en parallèle (en cours)

## miniDSTMaker

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

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

## Ajout du programme convert (en cours)

- Transforme 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

## Sommaire

## ① Introduction

Processus étudiés  
Détecteur SDHCAL  
Projets de Détecteurs

## ② Programmes

original  
iLCSoft  
FCC

## ③ Outils Numériques

Script  
Test

## ④ Conclusion

Premiers résultats  
État actuelle

## 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 sortie :
  - des programmes Xx = {processus, analysis}
  - de type Yy = {Completed, Same}

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

**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 des fichiers sont sensés :

- être identiques : DATA.root, données initiales
- d'autres différents : model\_[...].root
- d'autres équivalents : stats\_[...].root

## Sommaire

## ① Introduction

Processus étudiés  
Détecteur SDHCAL  
Projets de Détecteurs

## ② Programmes

original  
iLCSoft  
FCC

## ③ Outils Numériques

Script  
Test

## ④ Conclusion

Premiers résultats  
État actuelle

## Premiers résultats

analysis

```

"0": {
  "stat": 312101,
  "sum": 20547665804624557,
  "statPreSel": 298409,
  "sumPreSel": 195646231174468994,
  "statSel": 209515,
  "sumSel": 13793752670288086,
  "name": "Signal",
  "effPreSel": 0.9561295848040957,
  "effSel": 0.671305091363886
},

```

```

"0": {
  "stat": 312298,
  "sum": 2054766735434532,
  "statPreSel": 298740,
  "sumPreSel": 195655617713928223,
  "statSel": 222924,
  "sumSel": 14667298316955566,
  "name": "Signal",
  "effPreSel": 0.9565863304562183,
  "effSel": 0.7138181704043305
},

```

```

"0": {
  "stat": 312064,
  "sum": 20547665238380432,
  "statPreSel": 298573,
  "sumPreSel": 19559358501434326,
  "statSel": 207665,
  "sumSel": 13673576354980469,
  "name": "Signal",
  "effPreSel": 0.9567684831030407,
  "effSel": 0.6654564494967518
},

```

**name** nom du type d'évènement

**stat** nombre total d'évènement mesuré

**sum** somme des poids de chaque évènement  
(signal sur bruit)

**statPreSel** nombre d'évènement pré-sélectionné

**sumPreSel** somme des poids des évènements  
pré-sélectionnés

**effPreSel** taux de pré-sélection

**statSel** nombre d'évènement sélectionné

**effSel** taux de sélection

## 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
- Étude de la variation des résultats dûs à la BDT

### Compétences renforcées

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

### Nouvelles compétences

- Utilisation de BDT