



Национальный исследовательский ядерный университет  
«МИФИ»

Кафедра физики элементарных частиц №40



Научная исследовательская работа студента на тему:

# Исследование рождения легких векторных мезонов в ультрапериферических столкновениях тяжелых ионов

Научный руководитель:  
Тимошенко С.Л.

Работа  
студента 4-ого курса  
Захарова Арсения  
Михайловича  
ИЯФит

г. Москва 2021

# Что такое STARlight?

STARlight - это Монте-Карло генератор, моделирующий двухфотонное и фотон-померонное взаимодействие между релятивистскими ядрами и протонами.

Данная программа была написана специально для образования частиц в ультрапериферических взаимодействиях при энергиях RHIC для эксперимента STAR. Программный пакет учитывает возможность перехода ядра из основного состояния в возбужденное при обмене дополнительным фотоном. В программе также разыгрывается распад частиц по двухчастичному каналу, с учетом мод распада данной частицы.

# Поставленные задачи

Выполнено:

Задача 1:

Ознакомление с программным пакетом. Симуляция 10.000 распадов  $\rho^0 \longrightarrow \pi^+\pi^-$  с построением характерных гистограмм;

Задача 2:

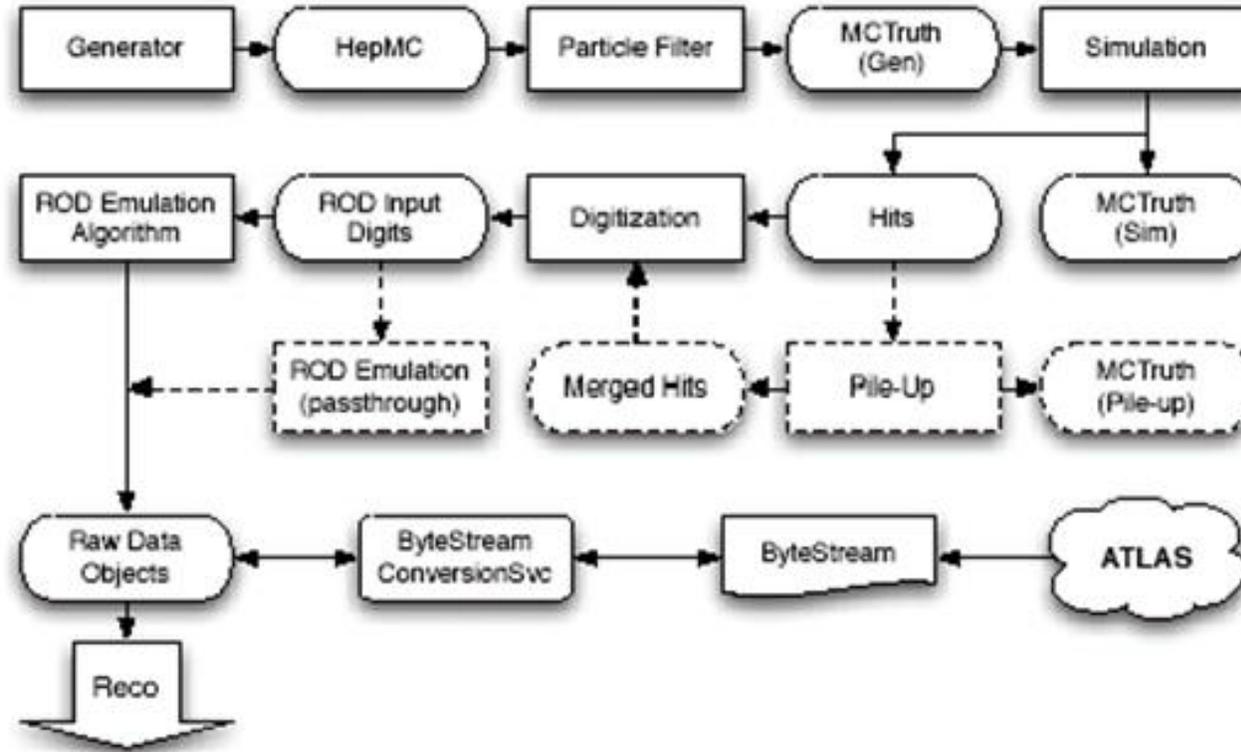
Введение нового канала распада  $\varphi \longrightarrow K_S^0 K_L^0$ , построение распределений;

Текущий этап:

Полное моделирование двухэтапного распада  $\varphi \longrightarrow K_L^0 K_S^0 \longrightarrow \pi^+\pi^-\pi^0 K_L^0$

С помощью ATLAS софта; Последующий анализ данных.

# Полное моделирование



Упрощенная схема полного моделирования

Основные блоки: генерация физического сигнала, процесс моделирования, оцифровка, реконструкция, создание AOD файла и анализ полученных данных. Хиты полученные в результате моделирования могут быть непосредственно обработаны алгоритмом преобразования в цифровую форму и преобразованы в формат «Объекты Сырых Данных» (RDOs). **Модификация кода -> standalone**

Генерация – MC, standalone выходной файл – ASCII. Для симуляции, оцифровки и реконструкции необходим EVNT.pool.root (HepEVNT).

Методы решения: 1. Написание/поиск конфига перевода;

2. Git -> копия Athena -> Подключение локальных библиотек -> Генерация (**Atlassian Tutorial**)

# Что такое Git?

Git — современная система управления версиями. Это развитый проект с активной поддержкой и открытым исходным кодом. Git применяется для управления версиями в рамках колоссального количества проектов по разработке ПО, как коммерческих, так и с открытым исходным кодом. Система используется множеством профессиональных разработчиков программного обеспечения.

Workflow Overview – возможность скопировать Athena/Generators/Starlight\_i -> подключение локальных библиотек -> генерация 334 канала на внутреннем StarLight -> выходной файл EVNT.pool.root

The screenshot shows the GitHub repository page for 'athena' under the 'atlas' organization. It includes the repository name, project ID (53790), star count (153), and statistics: 88,184 commits, 26 branches, 2,045 tags, 110.6 MB files, 673.8 MB storage, and 230 releases. A merge commit is highlighted: 'Merge branch 'GeantTruthThininnng\_re-entrant-cleanup' into 'master'' by Johannes Elmsheuser, 9 hours ago. Below this is a table of recent commits.

Name	Last commit	Last update
.devcontainer	vscode devcontainer: move motd display to ....	4 months ago
.vscode	add vscode setting for gitlab extension	7 months ago
AsgExternal/Asg_Test	Update ASG test inputs	11 months ago
AtlasGeometryCommon	Disable unit test post-processing where not ...	3 weeks ago
AtlasTest	TestTools: fix link to cmake documentation	6 days ago
Build	Removed the excess "--" from the script.	4 months ago
Calorimeter	CaloDepth Tool use enum to avoid too many...	3 days ago
Commission	rename uncalibrated TopoCluster container ...	1 month ago
Control	Merge branch 'jetConfigForReco' into 'master'	5 days ago

The screenshot shows the 'Workflow Overview' page from the ATLAS Software Documentation. It features a navigation sidebar with sections like 'Tutorial Home', 'Basics', 'Detailed Tutorial', 'Code Review', 'Reference', and 'Misc'. The main content area includes an introduction, a reminder of one-time steps, and a section for setting up the basic environment. A code block shows the commands for setting up the development environment.

## Workflow Overview

Last update: 12 Jan 2021 [History] [Edit]

### Introduction

This page gives you a quick overview of the ATLAS code development workflow. It is assumed that you worked through the [main git development tutorial](#) at least once. Please refer back to that if you are not clear on any points.

The workflow that ATLAS has adopted is basically [GitLab Flow](#), which offers enough flexibility to manage ATLAS use cases, but is structured enough to avoid a mess.

You may also find it handy to refer to our [git cheat sheet](#) as a quick reference to the most common command and terms.

### Reminder: one time steps

You only need to do these steps once, but for completeness we remind you to:

1. Check you have done your [git environment setup](#).
2. Check you have [made a fork](#) of the main ATLAS repository.

⚠ One point to reemphasise is **make sure that `atlasbot` is a developer in your fork** or continuous integration results are not published properly.

### Setup your basic environment

Assuming that you will work on an lxplus-like machine you want to start the development workflow by setting up decently modern version of git:

```
ssh lxplus
setupATLAS
lsetup git python
```

### Clone

Once you have your [fork](#) of the code in GitLab you need to [make a local copy](#) to work with and modify.

Cloning to AFS is slow, so if you have an alternative then we recommend that (e.g., a private local disk area, or even `$TMPDIR` for very short developments).

# Проблемы с симуляцией

## Ошибка, связанная с написанием канала

Написан на основе существующих -> распад на частицу и античастицу -> pdgIdCode =  $\pm 130$  – ошибка на первой же стадии, -130 не существует

До

```
EVENT: 1 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 0 0.124297 0.0562912 -14.1068 1 0 0 -130
TRACK: 10 -0.0649924 -0.033287 -18.2177 1 1 0 130
EVENT: 2 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 10 0.00863557 0.0637147 11.5942 2 0 0 130
TRACK: 0 -0.102671 -0.0125661 8.55472 2 1 0 -130
EVENT: 3 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 0 0.0948404 0.0129457 -0.253868 3 0 0 -130
TRACK: 10 -0.0993462 0.00168524 -0.232482 3 1 0 130
EVENT: 4 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 0 0.00606019 -0.0517015 -3.02633 4 0 0 -130
TRACK: 10 -0.0263082 0.0337192 -1.82674 4 1 0 130
EVENT: 5 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 0 0.0350755 0.0909863 -9.4007 5 0 0 -130
TRACK: 10 -0.00774562 -0.0731236 -7.17847 5 1 0 130
EVENT: 6 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 0 0.0962332 -0.0623385 1.26656 6 0 0 -130
TRACK: 10 -0.138833 0.0997447 1.22634 6 1 0 130
```

**STRANGE  
MESONS**

$K_L^0$	130
$K_S^0$	310
$K^0$	311
$K^+$	321

После

```
EVENT: 1 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 10 0.11779 0.0748806 -18.4373 1 0 0 130
TRACK: 16 -0.0584851 -0.0518764 -13.8872 1 1 0 310
EVENT: 2 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 10 -0.0941778 0.119586 10.2507 2 0 0 130
TRACK: 16 0.000142336 -0.0684372 9.89822 2 1 0 310
EVENT: 3 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 10 0.0313727 -0.123665 -35.8798 3 0 0 130
TRACK: 16 -0.0163072 0.0810673 -32.7521 3 1 0 310
EVENT: 4 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 16 0.0736513 0.0971558 -11.6445 4 0 0 310
TRACK: 10 -0.0767498 -0.0354239 -13.2126 4 1 0 130
EVENT: 5 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 16 0.0580088 0.0845076 9.59819 5 0 0 310
TRACK: 10 -0.0548169 -0.140689 10.0215 5 1 0 130
EVENT: 6 2 1
VERTEX: 0 0 0 0 1 0 0 2
TRACK: 10 -0.00438089 0.0982987 0.705422 6 0 0 130
TRACK: 16 0.0441684 -0.0545008 0.587536 6 1 0 310
```

Добавлен метод Rndom()

xtest < 0.5 -> ipId track1 = 130

ipId track2 = 310

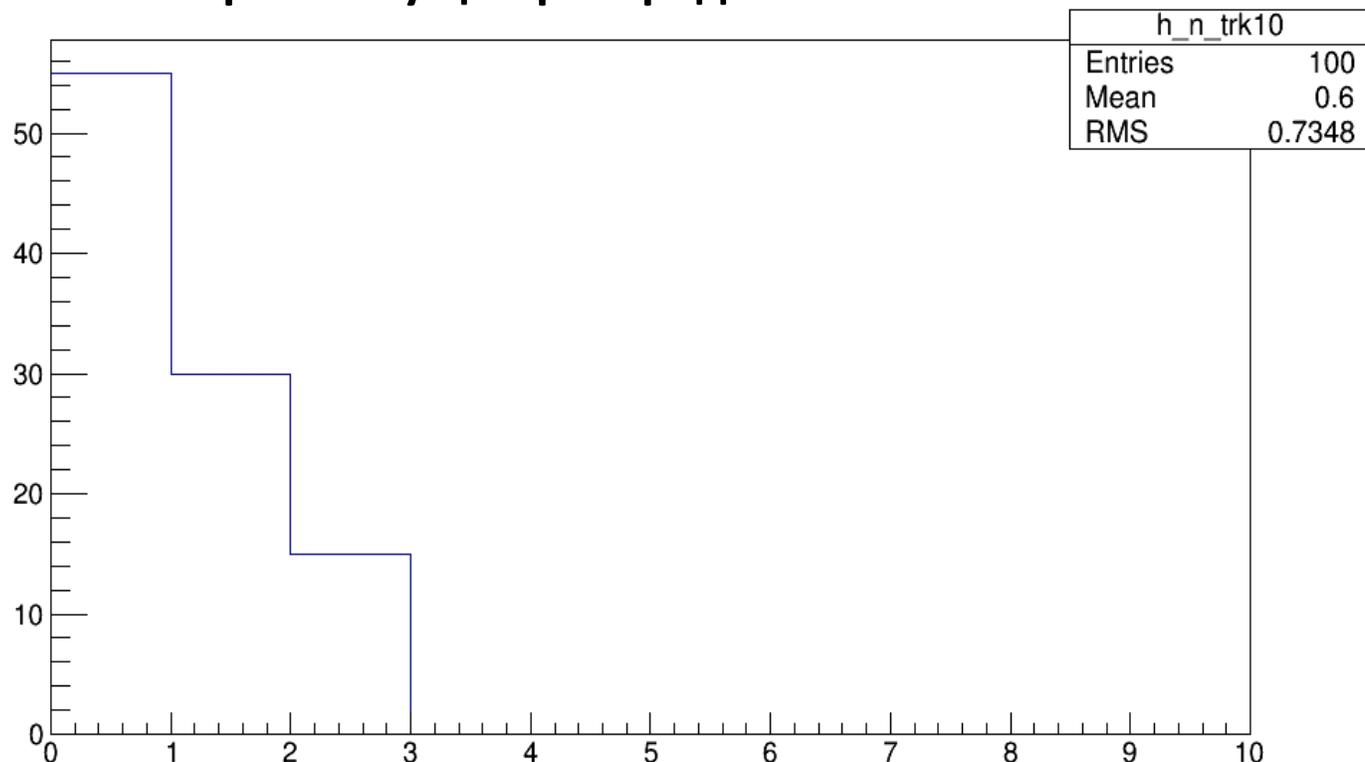
else -> ipId track1 = 310

ipId track2 = 130

# Проблемы реконструкции

Время обработки команды на lxplus ограничено -> реконструкция занимает большое количество времени -> ограничение по количеству событий на данном этапе (возможно около  $0 < n_{\text{Events}} (= 100) < 1000$ ) -> малая статистика. Данные сырые, возможно в будущем использования GRID.

## Некоторые текущие распределения

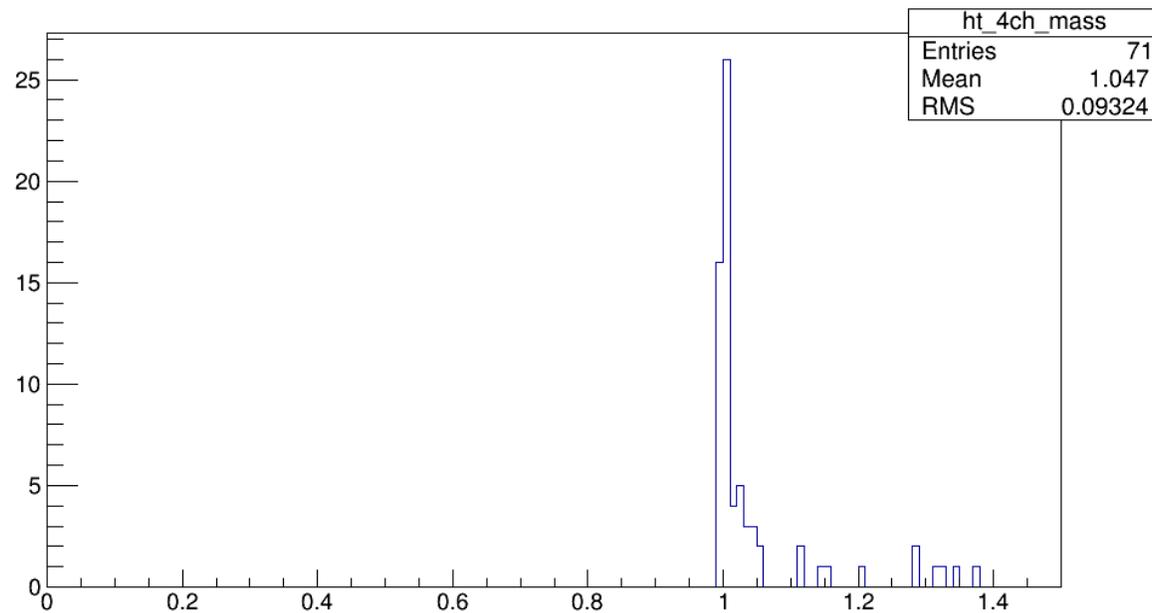


Около 50 событий без треков

30 событий с 1 треком и 14 с 2 треками

# Некоторые текущие распределения

Распределение по инвариантной массе

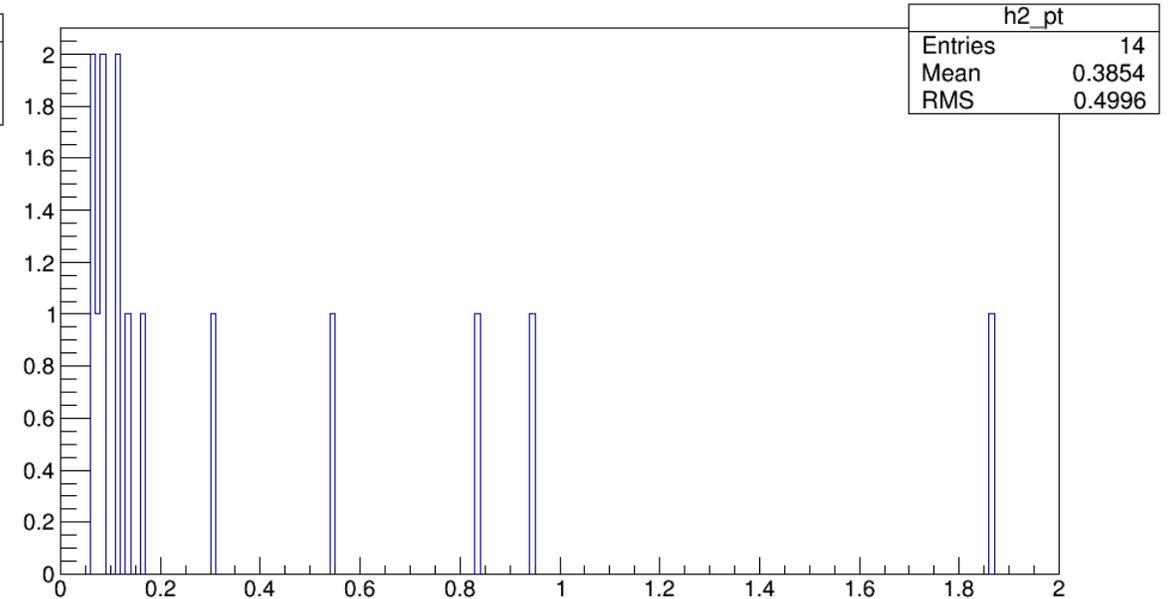


Распределение по инв. массе (пик в  $\approx 1$  GeV)

$$M_{K_L} = M_{K_S} = 0.497614 \text{ GeV}/c^2$$

$$\rightarrow 2 * 0.497614 \approx 1 \text{ GeV}$$

По поперечному импульсу



Пики достаточно небольшие -> объясняется тем, что при ультрапериферических взаимодействиях поперечный импульс образованной частицы очень мал

# Заключение

Сейчас:

- Теоретические сведения о программном пакете STARLight уже были освещены;
- Включен новый канал распада;
- Процесс полного моделирования практически освоен, решена проблема формата выходного файла и ошибки кода;

В будущем:

- Увеличение статистики, попытки загрузить в GRID;
- Сравнение полученных данных с моделью, построенной с помощью ParticleGun;
- Сравнение итоговых данных с полученными в ходе экспериментов ATLAS

# Команды

setupATLAS

1. Генерация: Asetup 21.6.20,AthGeneration

Gen\_tf.py --ecmEnergy=5020 --jobConfig=421120 --maxEvents=10 --outputEVNTFile = test\_starlight\_.EVNT.pool.root

2. Симуляция: asetup Athena,21.0.93,here

Sim\_tf.py --inputEvgenFile 'EVNT.pool.root' --outputHITSFile 'HITS.pool.root' --AMIconfig s3469

3. Оцифровка, Реконструкция: asetup Athena,21.0.97,here

Reco\_tf.py --inputHITSFile 'HITS.pool.root' --outputAODFile=AOD.pool.root --outputESDFile=ESD.pool.root --AMIconfig r11509

```
include("Starlight_i/Starlight_Common.py")

genSeq.Starlight.Initialize = \
["beam1Z 82", "beam1A 208", #Z,A of projectile
 "beam2Z 82", "beam2A 208", #Z,A of target
 # TODO: Calculate this from runArgs.ecmEnergy
 "beam1Gamma 2705", #Gamma of the colliding ion1, for sqrt(nn)=5.02 TeV
 "beam2Gamma 2705", #Gamma of the colliding ion2, for sqrt(nn)=5.02 TeV
 "maxW 4", #Max value of w
 "minW 0.6", #Min value of w
 "nmbWBins 200", #Bins n w
 "maxRapidity 3", #max y
 "nmbRapidityBins 200", #Bins n y
 "accCutPt 0", #Cut in pT? 0 = (no, 1 = yes)
 "minPt 0", #Minimum pT in GeV
 "maxPt 10.0", #Maximum pT in GeV
 "accCutEta 0", #Cut in pseudorapidity? (0 = no, 1 = yes)
 "minEta -2.7", #Minimum pseudorapidity
 "maxEta 2.7", #Maximum pseudorapidity
 "productionMode 2", #(1=2-phot,2=vmeson(narrow),3=vmeson(wide))
 "nmbEventsTot 1", #Number of events
 "prodParticleId 334", #Channel of interest
 "beamBreakupMode 5", #Controls the nuclear breakup
 "interferenceEnabled 0", #Interference (0 = off, 1 = on)
 "interferenceStrength 1.", #% of intefernce (0.0 - 0.1)
 "coherentProduction 1", #Coherent=1,Incoherent=0
 "incoherentFactor 1.", #percentage of incoherence
 "maxPtInterference 0.24", #Maximum pt considered, when interference is turned on
 "nmbPtBinsInterference 120", #Number of pt bins when interference is turned on
 "xsecMethod 1", #Set to 0 to use old method for calculating gamma-gamma luminosity
 "nThreads 1", #Number of threads used for calculating luminosity (when using the new method)
 "pythFullRec 1" #Write full pythia information to output (vertex, parents, daughter etc)
]
```