



中国科学技术大学
University of Science and Technology of China

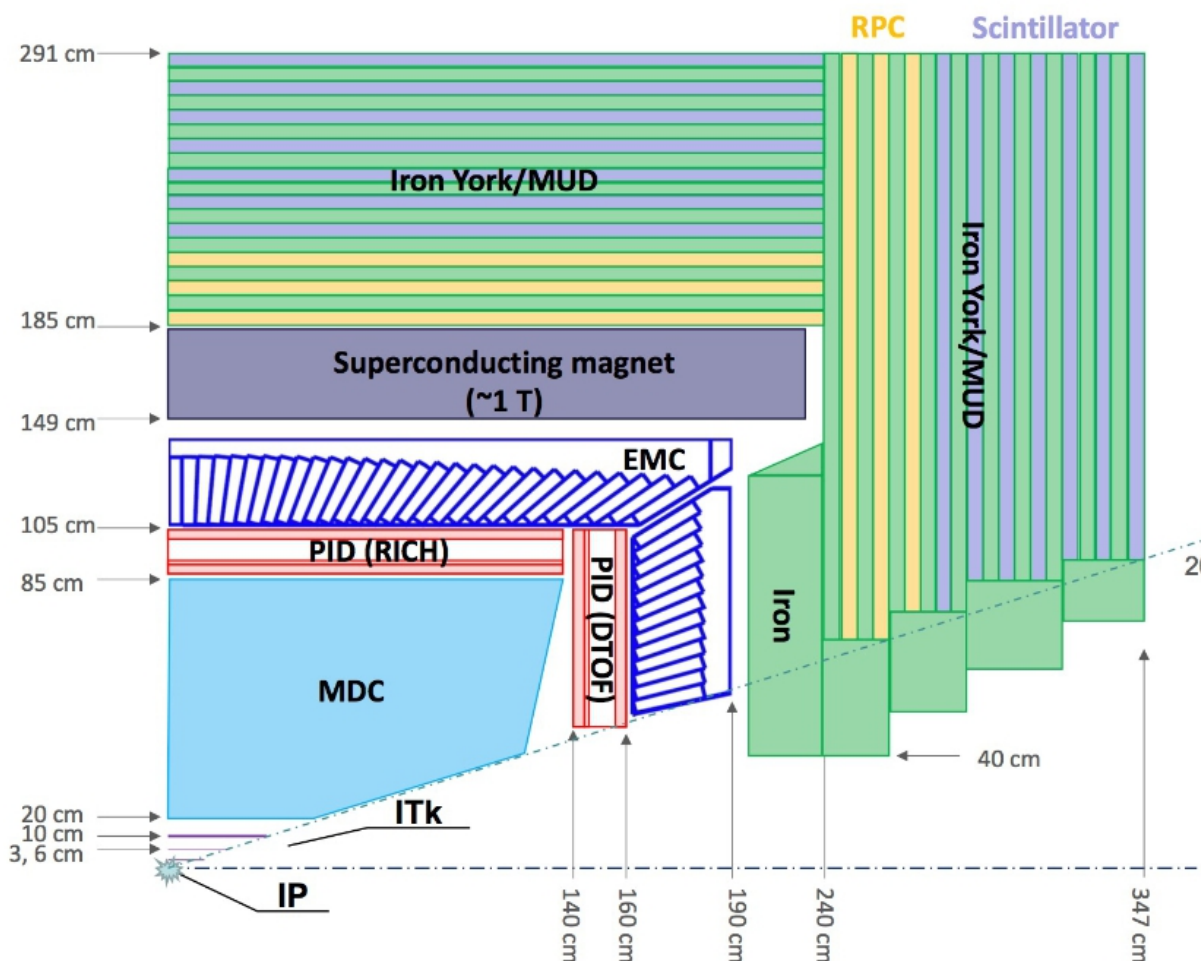
使用OSCAR进行模拟和物理分析

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi$ 为例

周杭

2025.02.22

STCF谱仪



ITK : Inner Tracker , 内径迹探测器

ITKW : 微结构气体探测器

ITKM : 硅像素探测器 , M-MAPS

两种方案

MDC : Main Drift Chamber , 漂移室 , 中心径迹探测器

端盖粒子鉴别探测器 (Particle Identification-Endcap , PIDE)

DTOF : DIRC-like TOF , 基于内部全反射切伦科夫光探测
(Detection of Internally Reflected Cherenkov , DIRC) 技术的
飞行时间探测器 (Time of Flight , TOF)

桶部粒子鉴别探测器 (Particle Identification-Barrel , PIDB)

RICH : 环形成像切伦科夫探测器 , Ring Imaging Cherenkov
detectors

BTOF : Barrel-TOF , 和DTOF相同技术

两种方案

ECAL (EMC) : 电磁量能器 , Electromagnetic Calorimeter

MUD : 缪子探测器 , Muon Detector

环境配置

科大服务器：

stcf01.ustc.edu.cn(202.141.163.202) , stcf02.ustc.edu.cn(202.141.163.203)

OSCAR外部库环境：source /software/STCF/OSCAR/2.6.0/bashrc.sh

OSCAR offline环境：source /software/STCF/OSCAR/2.6.0/offline/install/setup.sh

兰大服务器：

lzulogin01.hep.lzu.edu.cn , lzulogin02.hep.lzu.edu.cn

lzulogin03.hep.lzu.edu.cn , lzulogin04.hep.lzu.edu.cn

有4个节点，使用**lzulogin.hep.lzu.edu.cn**登录后会**自行分配**

OSCAR外部库环境：source /lzufs/user/STCF/oscar/centos7_amd64_gcc850/bashrc.sh

OSCAR offline环境：source /lzufs/user/STCF/oscar/centos7_amd64_gcc850/2.6.0/install/setup.sh

目前，科大和兰大服务器要配置并运行OSCAR均**需要使用容器**，进入容器指令：

```
export PATH=/cvmfs/common.ihep.ac.cn/software/hepjob/bin/:/cvmfs/container.ihep.ac.cn/bin/:$PATH
```

```
hep_container shell CentOS79 -g STCF
```

更多关于服务器的介绍和使用教程：<http://afsapply.ihep.ac.cn/cchelp/zh/>

自己编译offline（由于当前还不支持单独编译某个包，新增包后需重新编译，对大多数分析者来说这步**目前是必要的**）

获取offline源码：<https://git.ustc.edu.cn/>，需要申请git账号和OSCAR项目权限，也可从官方安装路径复制代码

配置外部库环境（**只需要**source外部库环境，如果同时source了已经编译好的offline环境再编译，可能会出问题）

```
cd your_path/offline
```

```
./build.sh
```

模拟

参考脚本路径：code_path/offline/Examples/RunCheck/FullChain/

其中sim_xxx.py为模拟脚本，可选粒子源为Generator或ParticleGun

rec_xxx.py为重建脚本，由于部分类别探测器有多个备选，因此有设置不同的脚本

配置环境：source OSCAR_externallib_PATH/bashrc.sh

source OSCAR_offline_PATH/install.sh

运行脚本：python xxx.py

```
import Sniper

task = Sniper.Task("task")
task.setLogLevel(4) #higher is less

import os
topdir = os.getenv('OFFLINETOP')

import StcfRndmGenSvc
sSvc=task.createSvc("StcfRndmGenSvc/hSvc")
sSvc.property("RndmSeed").set(28133)
```

创建任务。

setLogLevel：输出控制，数字越小输出越多

获取环境变量对应路径

随机数种子，每个作业应该不同
目前该处设置只对Generator生效。

整体设置


```
import KKMC
detalg0 = task.createAlg("KKMC")
detalg0.property("CMSEnergy").set(3.686)
detalg0.property("BeamEnergySpread").set(0.0013)
detalg0.property("NumberOfEventPrinted").set(0)

detalg0.property("GenerateResonance").set(True)
detalg0.property("GenerateContinuum").set(False)
detalg0.property("GenerateDownQuark").set(False)
detalg0.property("GenerateUpQuark").set(False)
detalg0.property("GenerateStrangeQuark").set(False)
detalg0.property("GenerateCharmQuark").set(True)
detalg0.property("GenerateMuonPair").set(False)
detalg0.property("GenerateTauPair").set(False)
detalg0.property("GenerateRho").set(False)
detalg0.property("GenerateOmega").set(False)
detalg0.property("GeneratePhi").set(False)
detalg0.property("GenerateJPsi").set(False) #3096
detalg0.property("GeneratePsiPrime").set(True) #3686
detalg0.property("GeneratePsi3770").set(False)
detalg0.property("GeneratePsi4030").set(False)
detalg0.property("GeneratePsi4160").set(False)
detalg0.property("GeneratePsi4415").set(False)
detalg0.property("GeneratePsi4260").set(False)
detalg0.property("ParticleDecayThroughEvtGen").set(True)
detalg0.property("RadiationCorrection").set(True)
```

产生子，目前可用的有：

KKMC, Babayaga, BBBrem, DIAG36, Phokhara

主要设置：质心能量和需要产生的粒子

```
import StcfEvtGen
detalg = task.createAlg("EvtDecay")
detalg.property("DecayDecDir").set("decay.dec") # "" means default
detalg.property("PdtTableDir").set("") # "" means default
detalg.property("userDecayTableName").set("decay.dec")
```

控制粒子衰变

DecayDecDir：衰变卡设置粒子衰变道和产生模式

如果不设置则使用默认值：\$STCFEVTGENROOT/share/DECAY.DEC

userDecayTableName：用户自定义衰变卡

会替换掉上一设置中的衰变模式

大多数产生子移植自BESIII软件包（BOSS），因此BES的大部分经验**仍然适用**，有BESIII工作经验的可以仿照之前设置使用

BOSS产生子文档：

<https://docbes3.ihep.ac.cn/~offlinesoftware/index.php/Generator>

```
#
Decay  psi(2S)
  分支比      末态粒子      衰变模型
1.0000      J/psi  pi+ pi-      JPIPI;
Enddecay
#
Decay  J/psi
  1.0000      mu+      mu-      PHOTOS  VLL;
Enddecay

End
```

衰变道

模拟

```
import PodioDataSvc
dsvc = task.createSvc("PodioDataSvc")

import PodioSvc
oSvc = task.createSvc("PodioOutputSvc/OutputSvc")
oSvc.property("OutputFile").set("Sim.root")
reduced_coll_list = ["EventHeaderCol", "MCParticleCol", "ITKHitCol",
oSvc.property("OutputCollections").set(reduced_coll_list)
```

数据IO服务

生成的模拟文件路径

想要保存的模拟数据集

(对少数情况如果完全用不上某些探测器
对应数据可以不存)

```
import GeometrySvc
myxmlsvc = task.createSvc("GeometrySvc")
myxmlsvc.property("GeoCompactFileName").set(topdir+"/GeoCompactFile.xml")

import MdcGeomSvc
Mdcgeom1 = task.createSvc("MdcGeomSvc")
Mdcgeom1.property("geomDataFilePath").set(topdir+"/CommonGeomDataFile.xml")
Mdcgeom1.property("geomDataSource").set(0)

import RICHGeoSvc
Richgeom = task.createSvc("RICHGeoSvc")

import DT0FGeoSvc
DT0Fgeom = task.createSvc("DT0FGeoSvc")
```

模拟使用的几何路径，根据使用的探测器情况可能有不同

子探测器几何服务，提供需要的几何信息比如探测器单元位置等

模拟

```
import G4Svc
g4svc = task.createSvc("G4Svc")
#g4svc.property("VisMac").set("vis.mac")
#g4svc.property("RunMac").set("run.mac")

import DetSimAlg
simalg = task.createAlg("DetSimAlg/DetSimAlg")
simalg.property("DetFactory").set("FullFactory")
```

模拟 (geant4) 相关服务和算法

```
import FullSim
genTool = simalg.createTool("GeneratorMgr")
genTool.property("Translation").set([0,0,0]) #generat
genTool.property("ParticleSource").set("Generator") #
#genTool.property("ParticleSource").set("ParticleGun")
genTool.property("Boost").set(True) #only work when P
```

后面还有一堆设置，通常遵从默认设置即可无需修改

如果选择ParticleSource为ParticleGun，需要设置run.mac

粒子来源

由于对撞不是完全正碰，
实验室系和质心系之间有boost
此处如果设置为true，
物理分析时**不要忘了这个boost**

```
# MultiTask for background mixing
from MultiStreamAnalysis import MultiStreamTask
AlgTask = MultiStreamTask("task")
AlgTask.property("TansSubColCache").set(True)
#AlgTask.disableAutoTriggering()
AlgTask.setLogLevel(4) #higher is less
```

如果重建和分析作业不在一个任务，
此设置**必须为true**

```
import PodioSvc
#Isvc = AlgTask.createSvc("PodioInputSvc/InputSvc")
#Isvc.property("InputFile").set("singleMuon.root")
AlgTask.addInputFile("Sim.root", "Sig") #signal hits
```

重建输入文件，即为模拟产生的文件
改xxx.root即可，“Sig”标签**不可更改**


```
#The following lines are used to add the background input
BKGDDir=os.getenv("OSCARDATA")+"/BkgDATA/offline_2.60_v7c3/ITKW_RICH_20241225"
BKGNameList=["Tous_e", "Tous_p", "Lumi", "Beamgas_e", "Beamgas_p"]
#AlgTask.addInputFile("Sim.root", "Bkg_Phys") #root file used here for test
for i in range(len(BKGNameList)):
    BKGFile=BKGDDir+"/"+BKGNameList[i]+".root"
    AlgTask.addInputFile(BKGFile, BKGNameList[i]) #beam bkg hits
```

添加本底（束流本底，亮度本底）输入文件

```
import BKG MixerSvc
BKGMix = AlgTask.createSvc("BKGMixerSvc")
BKGMix.property("BkgStreamList").set(BKGNameList) #
BKGMix.property("SubDetInfo").set(["ITK", "MDC", "P")
BKGMix.property("IsAddPhysBkg").set(False)
BKGMix.property("IsAddBeamBkg").set(False)
BKGMix.property("EnforceFixTimeWindow").set(False)
BKGMix.property("AvePhyNumInEachColl").set(1.6e-3)
BKGMix.property("BeamBkgStrength").set(1)
BKGMix.property("RndmSeed").set(123456)
```

添加本底文件后，仍然需要将IsAddBeamBkg设置为True才能加入本底

随机数种子不同作业应不一致

重建

```
outsvc = AlgTask.createSvc("PodioOutputSvc/OutputSvc")
outsvc.property("OutputFile").set("Rec.root")
#outsvc.property("TransferCollections").set(["MDCHitCol"])
#outsvc.property("TransferAllCollections").set(True)
```

输出文件，OSCAR标准数据输出
xxxCol

```
import RootWriter 服务名字“hSvc”，软件包中通过这个名字找到rootwriter
osvc = AlgTask.createSvc("RootWriter/hSvc")
osvc.property("Output").set({"Fkey" : "Rec_Info.root"})
```

输出文件，通用root文件
用户自定义，不依赖datamodel

注意保持一致

```
SniperPtr<RootWriter> hs(getParent(), "hSvc");
if (hs.valid())
```

```
import DDXMLSvc
xmlsvc = AlgTask.createSvc("DDXMLSvc")
xmlsvc.property("GeoCompactFileName").set(topdir+"/Geometry/FullGeometry/comp
```

重建所需几何，注意应和模拟的保持一致

```
task.setEvtMax(10)
```

作业事例数，如果设为-1表示循环输入文件中的所有事例

```
#####EventAssembler and SummaryWriter
import EventAssembler
EventAs = AlgTask.createAlg("EventAssembler")
#####EventAssembler and SummaryWriter END
```

EventAssembler：将对应同一重建粒子的不同探测器信息组装起来，构建ReconstructedParticle
由于PID系统和MUD重建均需要带电粒子重建信息作为输入，因此重建带电粒子信息自然关联起来
中性粒子由量能器构造

需要注意：带电粒子亦有可能在量能器上产生shower，通过带电径迹和量能器cluster重建位置匹配。
对某些特殊情况，EventAssembler提供的匹配未必够用，可能需要用户自行通过子探测器信息自行分析

分析

offline/Examples/RunCheck/Ana.py

```
import PodioSvc
Isvc = AlgTask.createSvc("PodioInputSvc/InputSvc")
Isvc.property("InputFile").set("Rec.root")

import RootWriter
osvc = AlgTask.createSvc("RootWriter/hSvc")
osvc.property("Output").set({"Fkey" : "Ana_Info.root"})
```

以重建文件作为输入

保存物理分析数据

```
import TrackInfoSvc
trInfo = AlgTask.createSvc("TrackInfoSvc")

import DeDxRec
dedxrecsvc = AlgTask.createSvc("DeDxRecSvc")

#Global
import GlobalPID
GlobalPID = AlgTask.createSvc("GlobalPIDSvc")
GlobalPID.property("SetModelPath").set(topdir+"
GlobalPID.property("SetMethod").set("XGBoost")
```

分析可能用到的服务（参数计算，PID等）

```
import PiPiJPsiAnalysis
AnaPiPiJPsi = AlgTask.createAlg("PiPiJPsiAnalysis")
AnaPiPiJPsi.property("CheckGlobalPID").set(False)
AnaPiPiJPsi.property("CheckAssembler").set(True)
AnaPiPiJPsi.property("CheckVertexAndKinematicFit").set(False)
AnaPiPiJPsi.property("LeptonType").set(-1)
```

分析算法包，作为例子之后介绍

脚本中导入的包按顺序执行，**注意导入顺序**
（有些包依赖于前置的包）

如何创建一个分析包

示例包：

offline/Analysis/RhoPiAnalysis

offline/Analysis/PiPiJPsiAnalysis

分析包应该为一个算法（Alg），包括initialize(), execute(), finalize()三部分

initialize()初始化参数设置，导入需要的服务

execute()逐事例处理

主算法代码：offline/Analysis/PiPiJPsiAnalysis/src

可参照示例包构建自己的分析软件包，

如果新添加了软件包，需要修改对应的CMakeLists.txt

如果复制已有包修改，注意**修改包名**

需要修改对应CMakeLists.txt包名

```
2 # Find three party libraries dependency
3 find_package(podio REQUIRED)
4 find_package(ROOT REQUIRED)
5
6 link_directories($ENV{OSCAR_EXTLIB_DD4hep_HOME}/lib)
7
8 include_directories(${CMAKE_CURRENT_SOURCE_DIR}/include)
9
10 oscar_add_pkg(AnalysisTest
11   DEPENDS PodioDataSvc DataModel PodioRootIO TrackInfoSvc MdcGeomSvc DeDxRec GlobalPID
12 )
13
14 # Install python running scripts
15 install(DIRECTORY share DESTINATION ${CMAKE_INSTALL_PREFIX})
16
```

此处为该程序包需要调用或依赖的包

并在上级目录的CMakeLists.txt里添加这个文件夹

```
1 set(dirlist RhoPiAnalysis VertexFit MultiStreamAnalysis GlobalPID AnalysisTest)
2 foreach(dir ${dirlist})
3   add_subdirectory(${dir})
4 endforeach()
```

然后按照前述步骤编译安装(如果已经配置环境，无需重新source)

目前暂不支持独立只编译某个包，因此如果需要新增或修改分析算法包都需要把整个offline重新编译一遍

分析所需通用功能

● 获取服务

```
GlobalPIDSvc* m_GlobalPIDSvc;
```

头文件定义变量

```
SniperPtr<GlobalPIDSvc> _globalPIDsvc(getParent(), "GlobalPIDSvc");  
if (_globalPIDsvc.valid())  
{  
    LogInfo << "the GlobalPIDSvc instance is retrieved" << std::endl;  
}  
else  
{  
    LogError << "Failed to get the GlobalPIDSvc instance!" << std::endl;  
    return false;  
}  
m_GlobalPIDSvc = _globalPIDsvc.data();
```

通过名字找到该服务（需要在脚本里创建）并赋值

分析所需通用功能

● 数据写出：RootWriter

```
import RootWriter
osvc = AlgTask.createSvc("RootWriter/hSvc")
osvc.property("Output").set({"Fkey" : "Ana_Info.root"})
```

在脚本中构建RootWriter

```
SniperPtr<RootWriter> hs(getParent(), "hSvc");
if (hs.valid())
{
    {
        recInfo = hs->bookTree(*getParent(), "Fkey/PiPiJPsiAnalysis", "title");
        recInfo->Branch("eventID", &EventID);
        recInfo->Branch("NTrack", &NTrack);
    }
}
```

名字标签，对应一致

TTree名字

算法包的initialize()阶段，
读取RootWriter，构建TTree

```
TTree* recInfo;
int EventID = 0;
int NTrack;
```

头文件定义所需要保存的数据
用法与常规的root数据填写完全一致

```
recInfo->Fill();
```

数据赋值后Fill()

分析所需通用功能

● 读取数据collection

```
auto m_mcParticle_new = getRWColl(MCParticleCollection, "MCParticleColl");
auto m_recParticle = getRWColl(ReconstructedParticleCollection, "ReconstructedParticleCol");
auto m_track = getRWColl(TrackerRecTrackCollection, "TrackerRecTrackCol");
auto m_dEdX = getRWColl(dEdXRecTrackCollection, "DEDXRecTrackCol");
auto m_extTrack = getRWColl(RecExtTrackCollection, "RecExtTrackCol");
auto m_dtofPid = getRWColl(DTOFPidCollection, "DT0FPidCol");
auto m_richPid = getRWColl(RecRICHLikelihoodCollection, "RecRICHLikelihoodCol");
auto m_shower = getRWColl(RecECALShowerCollection, "RecECALShowerCol");
auto m_mudTrack = getRWColl(MUDTrackCollection, "MUDTrackCol");
auto m_mudCluster = getRWColl(MUDClusterCollection, "MUDClusterCol");
```

getROColl, readOnly 从内存或文件中读取数据, 不存在返回空指针

注意, **数据可能不存在**, 此时调用函数 (例如 m_track->size()) 会导致break

getRWColl, readWrite 从内存读取数据, 不存在则创建

此两函数根据第二个参数的名字寻找Col是否存在, 对于getROColl, 一定要注意这个**名字是否正确**

应该在execute() (或其调用的子函数) 中获取collection

获取到的collection保存了这一个事例中的所有对应信息

可以使用的DataModel: offline/DataModel, 需include对应的头文件

分析所需通用功能

● 读取truth信息

```
m_mcParticle = getROColl(MCParticleCollection, "MCParticleColl");
```

读取MCParticleCollection

```
for (size_t j=0; j<m_mcParticle->size(); ++j) {  
    auto mcpair = m_mcParticle->at(j);
```

循环该事例保存的所有MCParticle，保存了所有原初粒子及其衰变产生的粒子（包括中间态）

```
auto start_point = mcpair.getVertex();
```

读取粒子数据，getXXX()，可获取到的数据参见 `offline/DataModel/DataModel/MCParticle.h`

```
for(size_t j=0; j<mcpair.parents_size(); j++)  
{  
    auto parent = mcpair.getParents(j);  
}
```

获取某个粒子的母粒子

```
for(size_t j=0; j<mcpair.daughters_size(); j++)  
{  
    auto daughter = mcpair.getDaughters(j);  
}
```

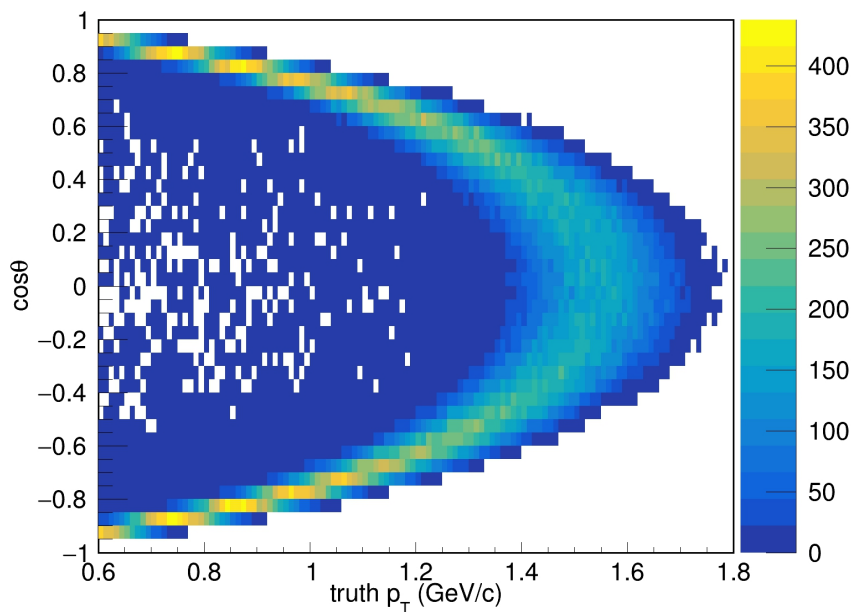
获取某个粒子的子粒子

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

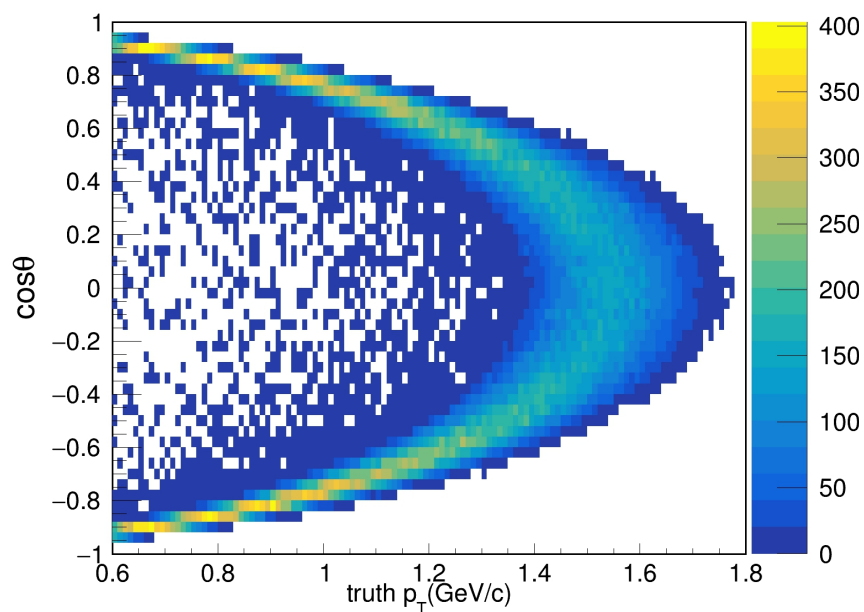
软件包位置：offline/Analysis/PiPiJPsiAnalysis

主程序：offline/Analysis/PiPiJPsiAnalysis/src/PiPiJPsiAnalysis.cc

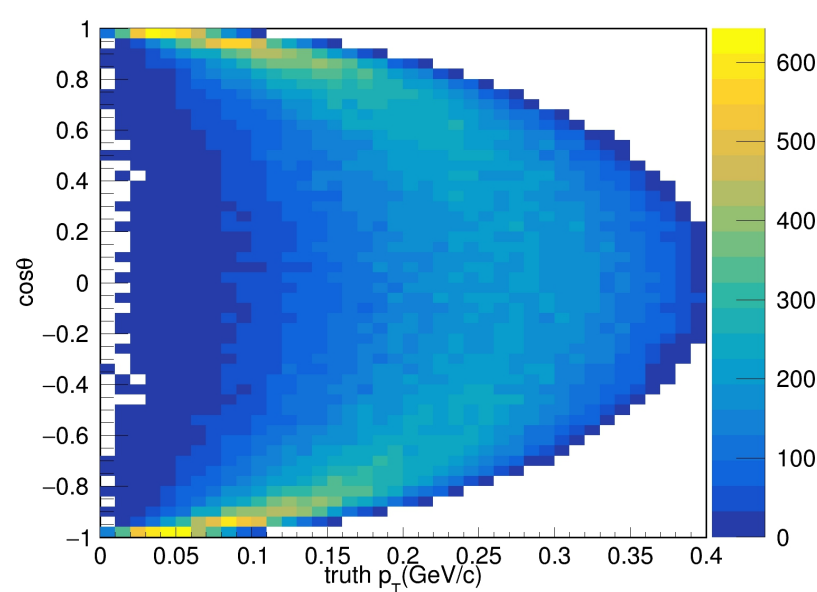
头文件：offline/Analysis/PiPiJPsiAnalysis/include/PiPiJPsiAnalysis.h



MC truth e^-



MC truth μ^-



MC truth π^+

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi (\rightarrow L^+L^-)$ 为例

Event selection:

Charged tracks

- $|d_0| < 1.0 \text{ cm} \ \&\& \ |d_z| < 10.0 \text{ cm};$

- $N_{positive} = 2 \ \&\& \ N_{negative} = 2.$

PID

- $p_\pi < 1.0 \text{ GeV}/c, \ p_l > 1.0 \text{ GeV}/c;$

- $N_{\pi^-} = 1 \ \&\& \ N_{\pi^+} = 1 \ \&\& \ N_{l^+} = 1 \ \&\& \ N_{l^-} = 1.$

- Electron: $E_{\text{EMC}} > 1.1 \text{ GeV}$

- Muon: $E_{\text{EMC}} < 0.35 \text{ GeV}$

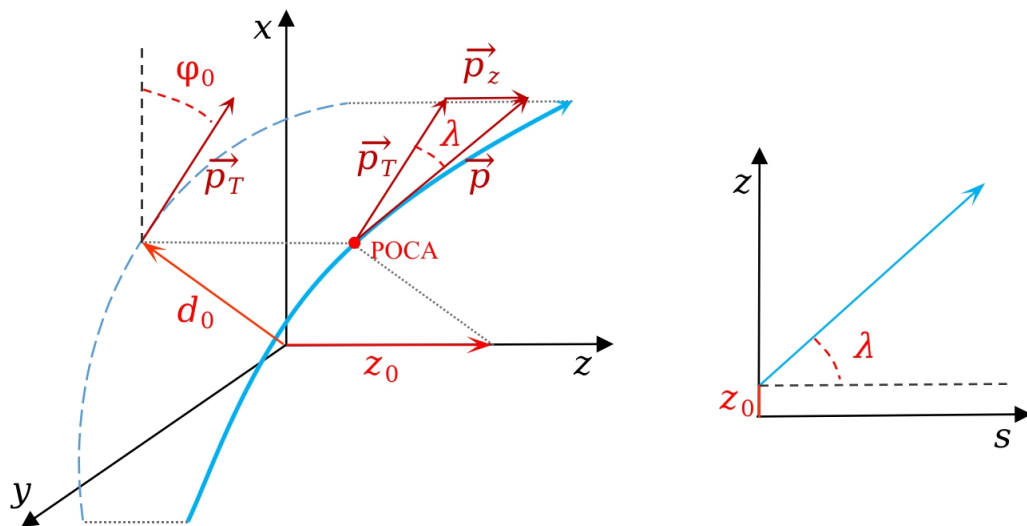
Kinematic Fit

- four-momentum constraint, $\chi_{4c}^2 < 60$

- $3.08 \text{ GeV}/c^2 < M(l^+ l^-) < 3.12 \text{ GeV}/c^2$

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

■ $|d_0| < 1.0 \text{ cm}$ && $|d_z| < 10.0 \text{ cm}$;



参考点：默认为原点 $(0, 0, 0)$
POCA(point of closest approach)：
径迹在xy平面离参考点最近的点

螺旋线五参数：

d_0 带符号距离，POCA到参考点距离

Φ_0 为初始极角

R 径迹圆半径

(或 $Kappa$, $1/p_T$)

z_0 POCA点的 z

$\tan L \approx p_T/p_z$

在此事例中，末态粒子在原点产生：

径迹真实的 d_0 , $z_0 \sim 0$

但重建有误差， d_0 , z_0 的选择条件越严格，
可能被排除的事例越多

这个选择条件和**重建精度**

挑选物理事例的特征（**顶点位置**等）

和可能混进的**本底水平**相关

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

■ $|d_0| < 1.0 \text{ cm} \ \&\& \ |d_z| < 10.0 \text{ cm};$

```
auto m_recParticle = getROColl(ReconstructedParticleCollection, "ReconstructedParticleCol");
if (!m_recParticle)
    return;
int nRecParSize = m_recParticle->size();
```

获取ReconstructedParticle

```
for (int i = 0; i < nRecParSize; i++)
{
    if (m_recParticle->at(i).getTrack().isAvailable())    判断是否是带电粒子
    {
        double d0 = m_recParticle->at(i).getTrack().getTrackHypos(1).getD0();
        double z0 = m_recParticle->at(i).getTrack().getTrackHypos(1).getZ0();
        double pT = m_recParticle->at(i).getTrack().getTrackHypos(1).getMomPOCA().Pt();
```

获取对应径迹信息

getTrack()获取的是TrackerRecTrack对象 (offline/DataModel/DataModel/TrackerRecTrack.h)
每个TrackerRecTrack关联到五种粒子假设的径迹参数 (offline/DataModel/DataModel/TrackerHypoTrack.h)
编号从0-4分别为e , mu , pi , ka , p。例：getTrackHypos(1)即为获取mu假设下的TrackerHypoTrack
获取径迹动量和位置： (Vector3d类 offline/DataModel/DataModel/Vector3d.h)

```
/// Access the first fitted point
const Vector3d& getPosFirst() const;

/// Access the Momentum at first fitted point
const Vector3d& getMomFirst() const;
```

第一个径迹点
附近参数

```
/// Access the
const Vector3d& getPosPOCA() const;

/// Access the
const Vector3d& getMomPOCA() const;
```

POCA附近参数

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

■ $N_{positive} = 2 \ \&\& \ N_{negative} = 2.$

```
if (fabs(d0) < m_d0cut && fabs(z0) < m_z0cut)
{
    if(m_recParticle->at(i).getTrack().getCharge()==-1){
        Qsum--;
    }
    if(m_recParticle->at(i).getTrack().getCharge()==1){
        Qsum++;
    }
    nCharge++;
    selectTracks.push_back(i);
}
```

判断电荷

注意：重建出的径迹数可能大于所研究事例的末态径迹数（尤其是在有本底的情况下）
因此严格要求重建径迹数等于需要径迹数**可能导致明显效率丢失**，具体的选择条件要根据所研究的物理道特点决定

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

■ $p_{\pi} < 1.0 \text{ GeV}/c$, $p_l > 1.0 \text{ GeV}/c$;

■ $N_{\pi^-} = 1 \&\& N_{\pi^+} = 1 \&\& N_{l^+} = 1 \&\& N_{l^-} = 1$.

```
if (m_recParticle->at(i).getTrack().getTrackHypos(0).getMomFirst().Mag() > 1)
{
    nGoodFlag++;
}
else if (m_recParticle->at(i).getTrack().getTrackHypos(2).getMomFirst().Mag() < 1)
{
    nGoodFlag++;
}
```

注意：在不知道粒子是什么时可以选取一种假设，但确认粒子种类后应读取对应假设的径迹参数
比如此处不确定粒子是e还是mu，使用了e假设

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

■ Electron: $E_{\text{EMC}} > 1.1 \text{ GeV}$

■ Muon: $E_{\text{EMC}} < 0.35 \text{ GeV}$

```
if (m_recParticle->at(i).getTrack().getCharge() == -1)
{
    if (m_recParticle->at(i).getTrack().getTrackHypos(1).getMomFirst().Mag() > 1)
    {
        if (m_recParticle->at(i).getShower().isAvailable()) 获取对应的量能器能量沉积
        {
            if(m_recParticle->at(i).getShower().getEnergy()>1.1){
                lepnum++;
                LepMEcal=m_recParticle->at(i).getShower().getEnergy();
            }
            if(m_recParticle->at(i).getShower().getEnergy()<0.35){
                lepnum--;
                LepMEcal=m_recParticle->at(i).getShower().getEnergy();
            }
        }
    }
}
```


以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

Kinematic Fit

- four-momentum constraint, $\chi_{4c}^2 < 60$
- $3.08 \text{ GeV}/c^2 < M(l^+l^-) < 3.12 \text{ GeV}/c^2$

```
KalmanKinematicFit *kmfit = KalmanKinematicFit::instance();  
  
//-----4c-----  
double chisq4c = 9999.;  
// const double etot = 3.686;  
kmfit->init();
```

初始化运动学拟合

```
// kmfit->AddTrack(0, wpip);  
// kmfit->AddTrack(0, mass_pi, charge, parPos, parMom, cov, helix5);  
kmfit->AddTrack(0, mass_pi, hpip, pip);
```

添加径迹信息，
第一个参数为添加track的编号
添加4条径迹，编号为0-3

函数原型：(offline/Analysis/VertexFit/src/TrackPool.cxx)

```
void TrackPool::AddTrack(const int number, WTrackParameter wtrk) {  
void TrackPool::AddTrack(const int number, const double mass, int charge,  
Vector3d pos, Vector3d mom, CovMatrix6 cov, double helix[5])  
void TrackPool::AddTrack(const int number, const double mass,  
const TrackerHypoTrack htrk, const TrackerRecTrack trk) {
```

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例

Kinematic Fit

- four-momentum constraint, $\chi_{4c}^2 < 60$
- $3.08 \text{ GeV}/c^2 < M(l^+ l^-) < 3.12 \text{ GeV}/c^2$

```
/*  
kmfit->AddResonance(0,3.0969,2,3);  
kmfit->AddFourMomentum(1, m_Etot4C);  
*/  
kmfit->AddFourMomentum(0, m_Etot4C);  
bool okfit = kmfit->Fit();  
  
chisq4c = kmfit->chisq();
```

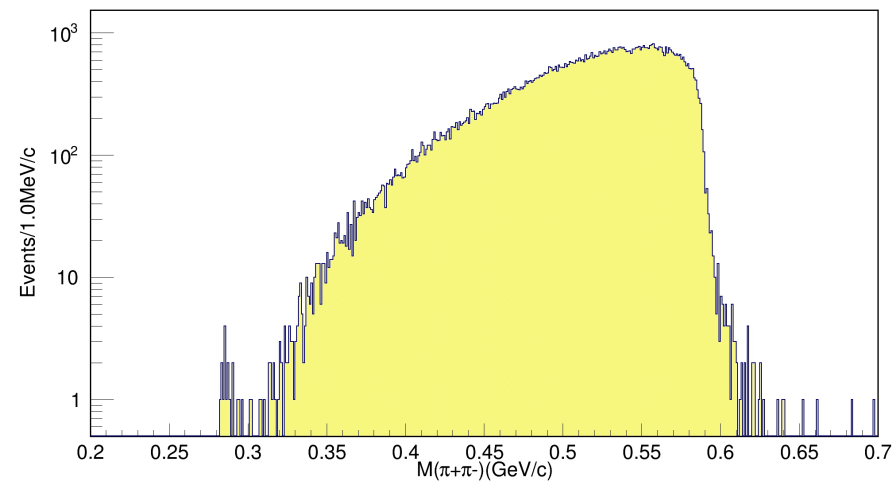
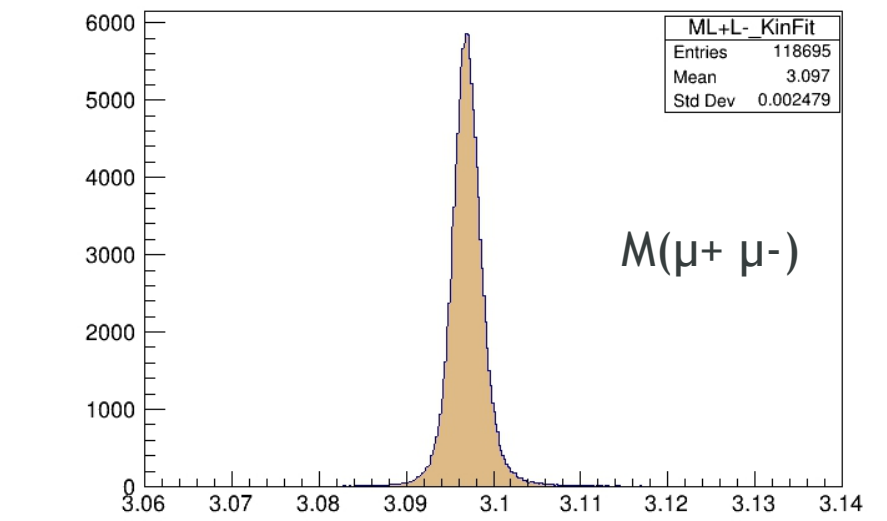
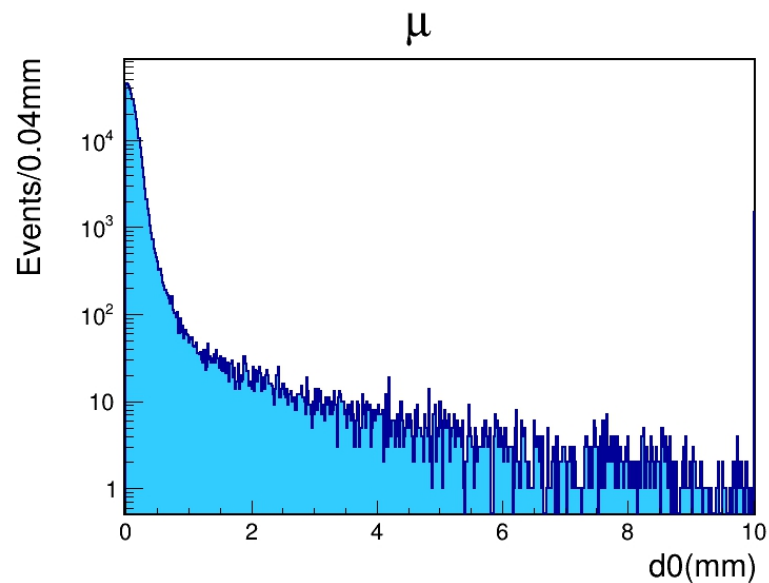
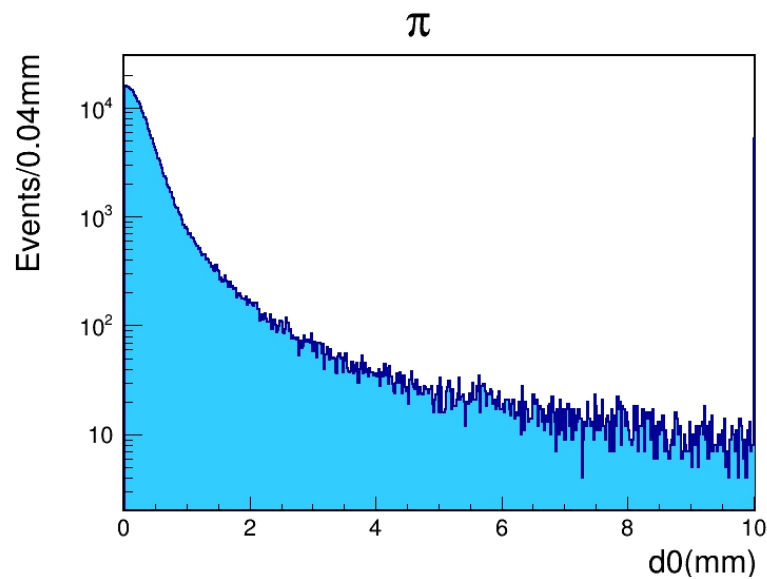
添加约束，进行运动学拟合

与AddTrack时的id对应

```
HepLorentzVector pipp = kmfit->pfit(0);  
HepLorentzVector pimp = kmfit->pfit(1);  
HepLorentzVector leppp = kmfit->pfit(2);  
HepLorentzVector lepmp = kmfit->pfit(3);  
  
MPiPi2 = (pipp + pimp).m();  
psi_mass = (pipp + pimp + lepmp + leppp).m();  
jpsi_mass = (leppp + lepmp).m();
```

获取运动学拟合后的参数

以 $\psi(3686) \rightarrow \pi^+\pi^- J/\psi(\rightarrow L^+L^-)$ 为例





BACKUP

不通过ReconstructedParticle获取重建信息

```
auto m_mcParticle_new = getRWColl(MCParticleCollection, "MCParticleColl");
auto m_recParticle = getRWColl(ReconstructedParticleCollection, "ReconstructedParticleCol");
auto m_track = getRWColl(TrackerRecTrackCollection, "TrackerRecTrackCol");
auto m_dEdX = getRWColl(dEdXRecTrackCollection, "DEDXRecTrackCol");
auto m_extTrack = getRWColl(RecExtTrackCollection, "RecExtTrackCol");
auto m_dtofPid = getRWColl(DTOFPidCollection, "DTOFPidCol");
auto m_richPid = getRWColl(RecRICHLikelihoodCollection, "RecRICHLikelihoodCol");
auto m_shower = getRWColl(RecECALShowerCollection, "RecECALShowerCol");
auto m_mudTrack = getRWColl(MUDTrackCollection, "MUDTrackCol");
auto m_mudCluster = getRWColl(MUDClusterCollection, "MUDClusterCol");
```

```
if (m_shower) {
    for (size_t j=0; j<m_shower->size(); ++j) {
        auto shower = m_shower->at(j);
    }
}
```

```
if (m_richPid) {
    for (size_t j=0; j<m_richPid->size(); ++j) {
        auto richpid = m_richPid->at(j);
    }
}
```

先获取对应的collection，然后循环读取元素

offline/Analysis/PiPiJPsiAnalysis/src/PiPiJPsiAnalysis.cc
EventAsInfo()函数内

OSCAR manual

[https://indico.pnp.ustc.edu.cn/event/2436/contributions/16612/attachments/5885/9090/
Get_start_with_analysis_in_OSCAR_v1.pdf](https://indico.pnp.ustc.edu.cn/event/2436/contributions/16612/attachments/5885/9090/Get_start_with_analysis_in_OSCAR_v1.pdf)

批量提交作业例子

offline/Examples/JobSubmission/hepjob

（由于服务器升级和环境配置等问题，该示例可能不能直接运行，可根据实际情况修改）

- 创建模板脚本，其中一些参数设为可调（输入输出文件名，随机数种子等）
- 生成多个可执行脚本，具有相似的名称但后缀ID不同
- 用hep_sub提交作业

```
hep_sub "run_${jobname}_${ProclD}.sh" -n ${maxjob} -wt short
```

```
/// Access the  
const float& getD0() const;  
  
/// Access the  
const float& getPhi0() const;  
  
/// Access the  
const float& getKap() const;  
  
/// Access the  
const float& getTanlamda() const;  
  
/// Access the  
const float& getZ0() const;
```