代码功能添加 scaling

RecNeutralTool.cc：

构造一个 TH2D；

按 (binXthe, binYene) 设置不同的 scaling 值；

保存为 .root 文件在代码中读取；

调整对应scale值

```
m_scaleResEne = 1.0;
m_scaleResThe = 1.0;
m_scaleResPhi = 1.0;
m_scaleRecEff = 1.0;
```

TH2D构造：

需要全模拟与快模拟数据（暂时不需要全模拟数据，每个bin的scaling设置为1.5或0.7，2等）

定义能量和角度的区间、分bin（产生一部分特定能量与角度的光子，与scaling为1的情况对比）

计算全模拟与快模拟分辨率的比Scaling=σ_FullSim/σ_FastSim

举例

θ_bin	E_bin	σ_FullSim	σ_FastSim	Scaling=σ_FullSim/σ_FastSim
40	72	0.028	0.025	1.12
20	50	0.019	0.021	0.90

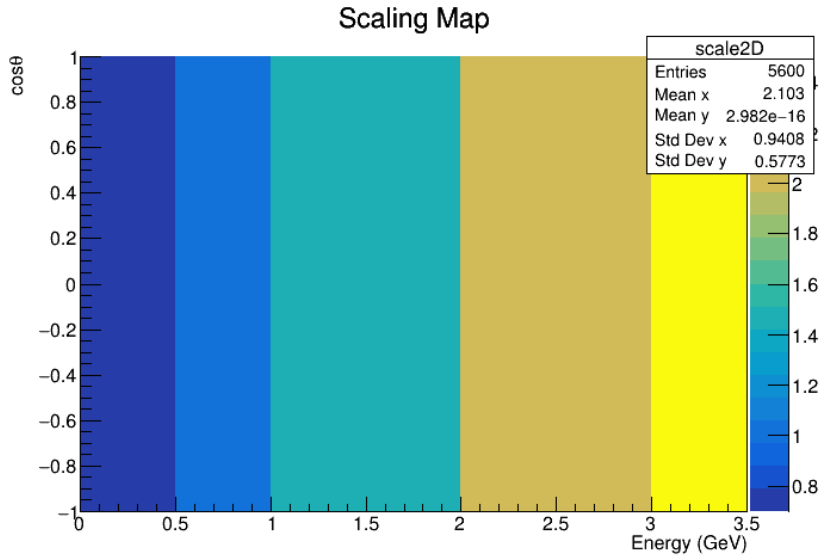
代码功能添加 scaling

RecNeutralTool.cc：

使用快模拟产生了10000个事例，光子动量0-3.5GeV，角度范围（-1，1），能量分70个bin，角度分80个bin

构造TH2D，先对m_scaleResEne = 1.0调整，scaling暂时赋值为：

```
if (E< 0.5) scaling= 0.7;
    else if (E < 1.0) scaling= 1.0;
    else if (E< 2.0) scaling= 1.5;
    else if (E< 3.0) scaling= 2;
    else scaling= 2.5;
```




```
#include "TH2D.h"
#include "TFile.h"
#include <iostream>

int make_scale_table() {
    int nBinE = 70;
    double E_min = 0.0;
    double E_max = 3.5;

    int nBinCos = 80;
    double cos_min = -1.0;
    double cos_max = 1.0;
```

保存为EneScaleTable.root 文件

在bool RecNeutralTool::recGamEmcShower(trutrk& m_truTrk, recgam& m_recGam){

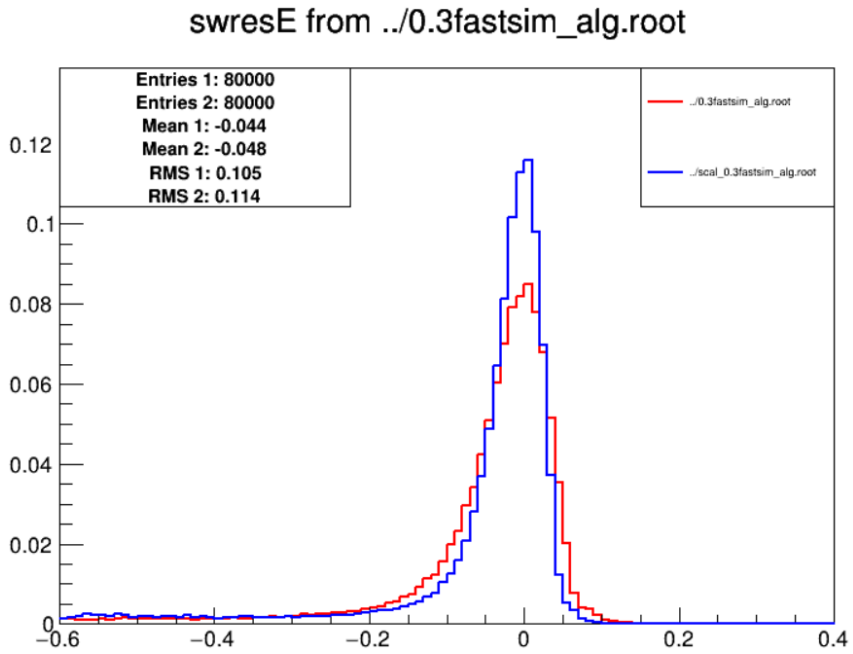
中读取EneScaleTable.root并改变m_scaleResEne 的值

代码功能添加 scaling

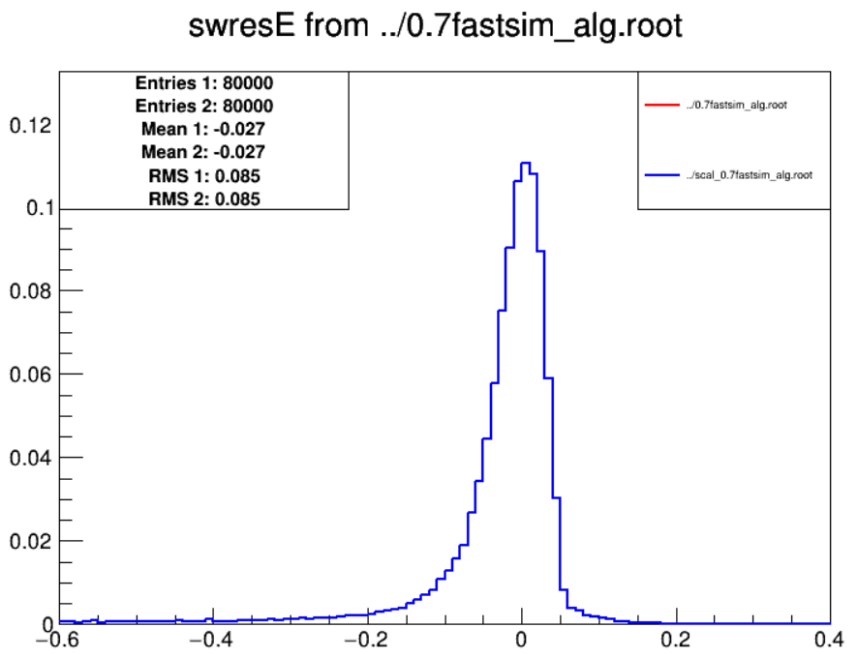
RecNeutralTool.cc：

生成0.3，0.7，1.5，2.5，3.3GeV的光子事例各80000个

跑分析算法FastTestAlg 生成 fastsim_alg.root 和 scal_fastsim_alg.root 对比能量分辨



Scaling = 0.7

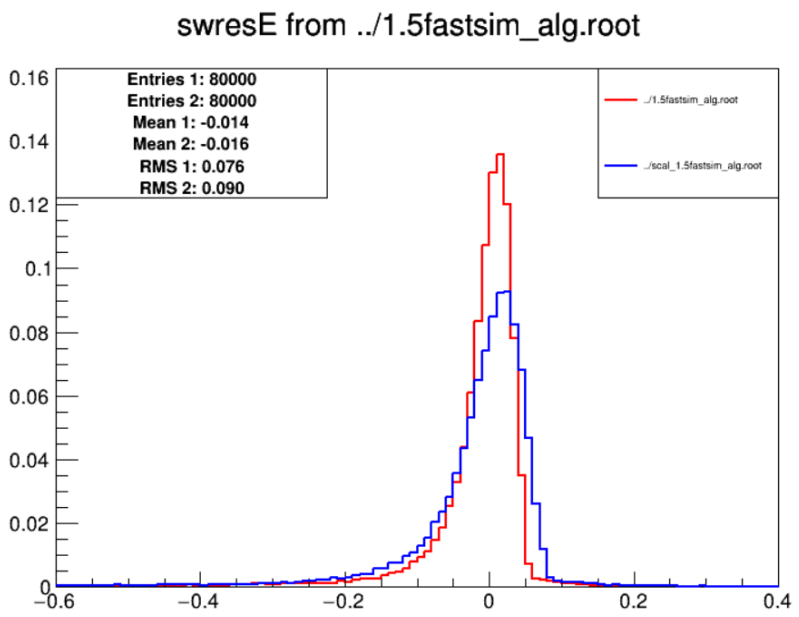


Scaling = 1

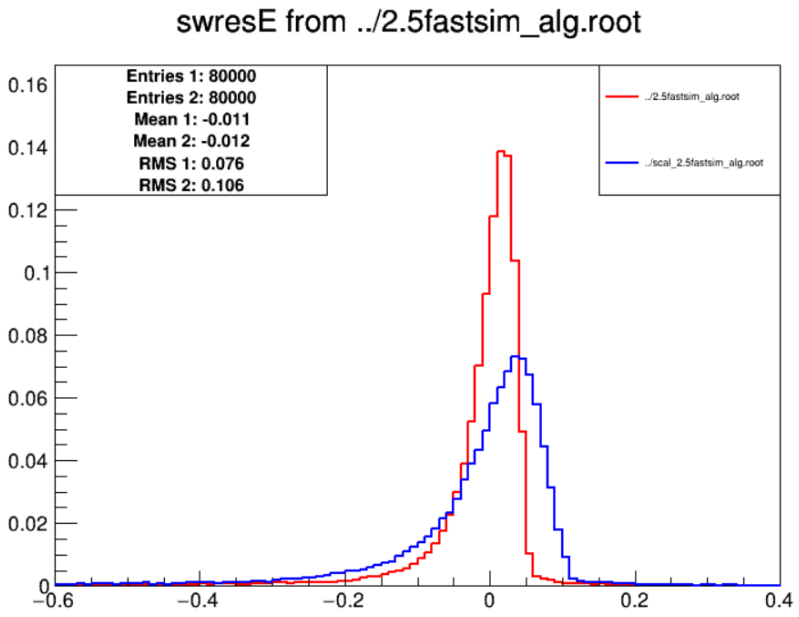
代码功能添加 scaling

RecNeutralTool.cc：

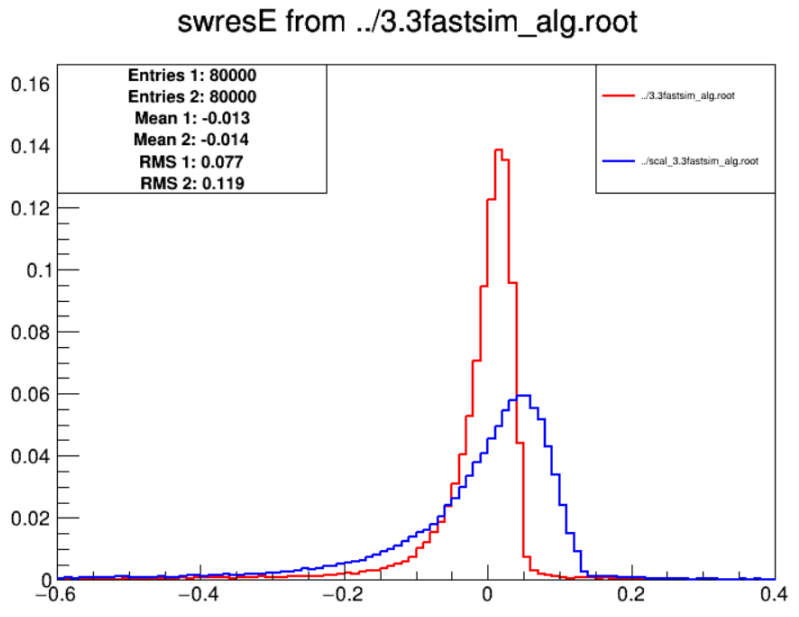
跑分析算法FastTestAlg 生成 fastsim_alg.root 和 scal_fastsim_alg.root 对比能量分辨



Scaling = 1.5

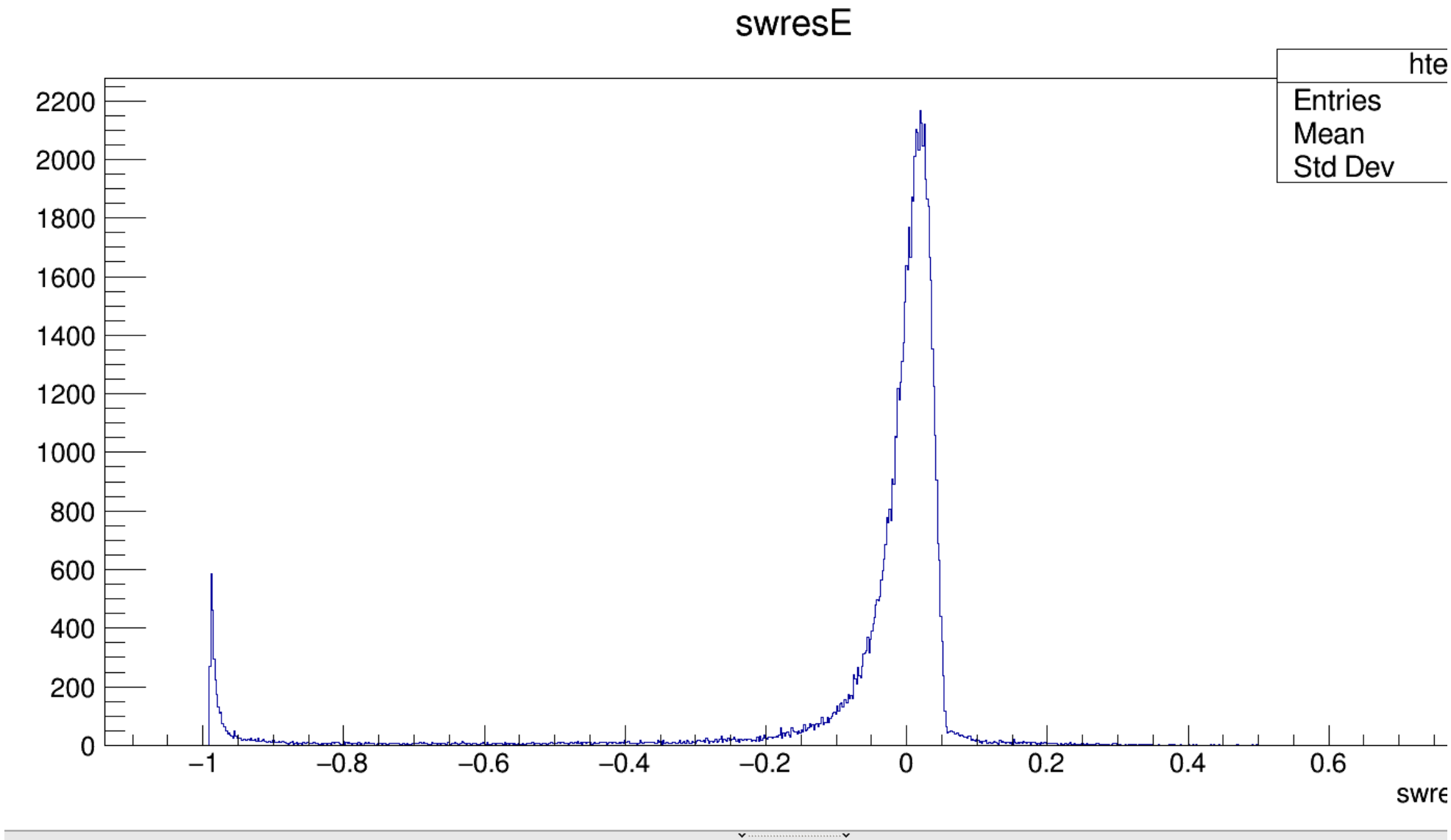


Scaling = 2



Scaling = 2.5

中心值偏移


swresE
hte
Entries
Mean
Std Dev
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
swre