```
//#include "FitApeak.hh"
#include "Fit2peaks.hh"
#include "TCanvas.h"
#include "TH1D.h"
#include "TFile.h"
#include "TTree.h"
#include "TString.h"
#include "TAxis.h"
#include "TStyle.h"
#include "TSystem.h"
#include <fstream>
#include <utility>

#include <iostream>
using std::cout;
using std::endl;

#include <vector>
using std::vector;

#include <string>
using std::string;

#include <sstream>
using std::stringstream;

int main(int argc,char* argv[])
{
        int runNumber;
        string inputCalName("");
        string outputCalName("");
        string reportName("");
        string resultsName("../output/mg25Results%i.txt");

        if(argc==1)
        {
                runNumber = 244;
                inputCalName = "../cal/calDefault.txt";
                outputCalName = "../cal/new/calDefault.txt";
                reportName = "../reports/test.pdf";
        }
        else if(argc==2)
        {
           stringstream ssRunNumber(argv[1]);
                ssRunNumber >> runNumber;
                inputCalName = Form("../cal/cal%i.txt",runNumber);
                outputCalName = Form("../cal/new/cal%i.txt",runNumber);
                reportName = Form("../reports/report%i.pdf",runNumber);
  }
  else if(argc==3)
  {
                stringstream ssRunNumber(argv[1]);
                ssRunNumber >> runNumber;
                inputCalName  = Form("%s",argv[2]);
                outputCalName = Form("../cal/new/cal%i.txt",runNumber);
                reportName = Form("../reports/report%i.pdf",runNumber);
  }


        // remove annoying "pdf/png created statements"
        gROOT->ProcessLine("gErrorIgnoreLevel = 1001;");
        gStyle->SetOptStat("ni");
```

```
        gStyle->SetOptFit(1111);

        //run10427.root is background run at the end of the low energy portion of the
2017 17O(a,ng) HAGRiD runs
        //
        const int Ndetectors = 13;
        const int bnum[Ndetectors] = {0,0,0,0,0,0,0,  1,1, 1,1,1,1 };//
        const int cnum[Ndetectors] = {0,1,2,3,4,5,6,  0,1, 3,4,5,6 };//


        TCanvas *c1 = new TCanvas("c1","c1",0,0,600,400);

        //runfile name pattern for COMPASS
        TString runfilename = Form("../rootfiles/run_%i/UNFILTERED/
compass_run_%i.root",runNumber,runNumber);


        cout << "Trying to find UNFILTERED Data... " << runfilename << endl;
        if(gSystem->AccessPathName(runfilename.Data() )  )
        {
                cout << "\tUNFILTERED Data not found!... " << runfilename << endl;
                runfilename = Form("../rootfiles/run_%i/FILTERED/
compass_run_%i.root",runNumber,runNumber);
                cout << "Trying to find FILTERED Data... " << runfilename << endl;
                if(gSystem->AccessPathName(runfilename.Data() )  )
                {
                        cout << "\tFILTERED Data not found!... " << runfilename <<
endl;
                        return -1;
                }
        }
        cout << "\t Processing data file " << runfilename << endl;
        TFile *runFile = new TFile(runfilename);
        TTree *runTree = static_cast<TTree*>(runFile->Get("Data"));
        TFile *bgfile  = new TFile("../cal/calibratedBackgroundHists.root");
        Fit2peaks *mg25analysis = new Fit2peaks(bgfile);
        mg25analysis->SetEgamma(0,1778.97);
        mg25analysis->SetEgamma(1,2838.3);
        c1->Print(Form("%s(",reportName.c_str()),".pdf");
        ifstream ecal(inputCalName.c_str());
        ofstream ecalNew(outputCalName.c_str());
        ofstream results[Ndetectors];

        //mg25analysis->Rebin(16);
        for(int i = 0; i<Ndetectors;i++)
        {
                results[i].open(Form(resultsName.c_str(),i),std::ofstream::app);
                if(!results[i].is_open())
                {
                        cout << "error opening results file: " <<
Form(resultsName.c_str(),i) << endl;
                }
                if(!ecal.is_open())
                {
                        cout << "error opening calibration file: "
                                << inputCalName << endl;
                        break;
                }
                TString id,check;
                check = Form("b%i_c%i",bnum[i],cnum[i]);
                double offset,gain,quad,resolution;
                ecal >> id >> offset >> gain >> quad >> resolution;
```

```
                if(id!=check)
                {
                        cout << "channel mismatch with caibration ("+id+"!="+check+")
"
                                << inputCalName << endl;
                        break;
                }
                //TO DO: grab this time from run.info file
                double tfg = runTree->GetMaximum("Timestamp")-runTree-
>GetMinimum("Timestamp");
                tfg/=1e12; // Timestamps are in picoseconds.
                double tbg = 3987.0;// Background run live time in seconds from the
timestamps in run 244

                mg25analysis->LoadBackground(bnum[i],cnum[i]);
                mg25analysis-
>CreateSpectrum(runTree,bnum[i],cnum[i],Form("h%i_%i",bnum[i],cnum[i]));

                mg25analysis->SetParameters(1000,resolution,offset,gain,quad,
1,0.002,500,1000,resolution*1.5,0.002);
                mg25analysis->FixParameter(5,tfg/tbg);
                c1->cd();
                c1->SetGridy();
                c1->SetLogy(1);
                mg25analysis->Reject(true);
                mg25analysis->GetSpectrum()->Draw("E0");
                mg25analysis->GetSpectrum()->GetXaxis()->SetRangeUser(0,0);
                mg25analysis->Fit("mq");
                mg25analysis->Subtract();
                mg25analysis->Reject(false);
                mg25analysis->GetBgFunction()->DrawClone("SAME");
                mg25analysis->Reject(true);

                //mg25analysis->RejectMore(true);
                mg25analysis->GetSpectrum()->GetXaxis()->SetRangeUser(250.0/gain,
5000.0/gain);
                c1->Update();
                c1->Print(reportName.c_str(),".pdf");
                mg25analysis->GetSpectrum()->GetXaxis()->SetRangeUser(1400.0/gain,
2200.0/gain);
                c1->Update();
                c1->Print(reportName.c_str(),".pdf");
                mg25analysis->GetSpectrum()->GetXaxis()->SetRangeUser(2500.0/gain,
3500.0/gain);
                c1->Update();
                c1->Print(reportName.c_str(),".pdf");
                mg25analysis->GetCalibrated()->Draw("E0");
                mg25analysis->GetCalibrated()->GetXaxis()->SetRangeUser(400,3000);
                c1->SetLogy(0);
                c1->Update();
                c1->Print(reportName.c_str(),"pdf");

                mg25analysis->GetCalibrated()->GetXaxis()-
>SetRangeUser(1779-150,1779+150);
                c1->Update();
                c1->Print(reportName.c_str(),"pdf");
                gain = mg25analysis->GetParameter(3);
                offset = mg25analysis->GetParameter(2);
                resolution = mg25analysis->GetParameter(1);
                double yield = mg25analysis->GetPeakArea(0);
                double error = mg25analysis->GetPeakUnc(0);
```

```
                ecalNew << id <<"\t" << offset <<"\t" << gain << "\t" << quad << "\t"
<< resolution << endl;
                results[i] << runNumber <<"\t" << yield << "\t" << error << "\t" <<
gain
                        << "\t" << offset << "\t" << resolution << endl;


        }
        ecal.close();
        c1->Clear();
        c1->Print(Form("%s)",reportName.c_str()),".pdf");


        delete mg25analysis;
        delete bgfile;
        delete runFile;
        return 0;
}
```