W Physics Tutorial

Felix Friedrich, Arno Straessner – TU Dresden
Jana Kraus, Eckhard v. Törne – Universität Bonn

Introductory School to Particle Physics
DESY Hamburg
March 8-12, 2010

Outline

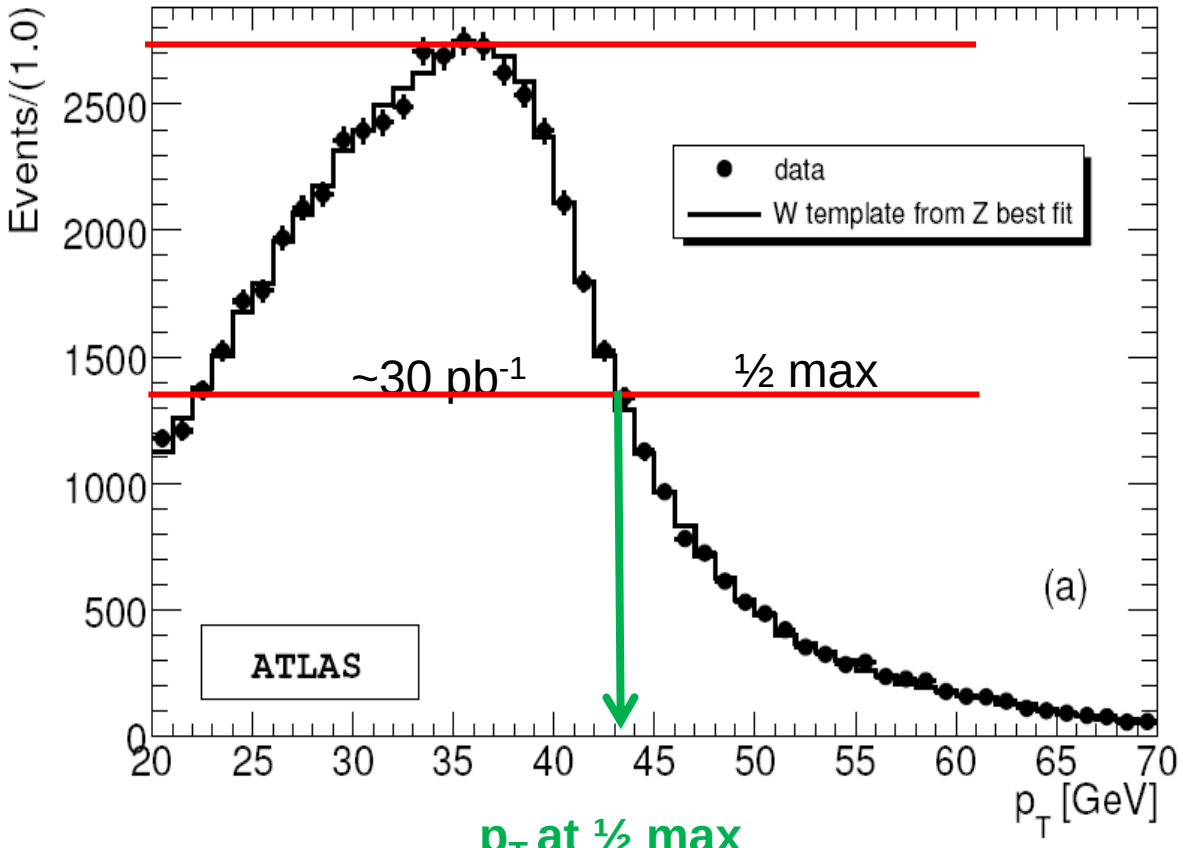
- In the tutorial of today you will learn:

- Part III:
  - how to measure of the W mass from lepton p_T spectrum
  - how the "template" Monte Carlo method works
  - how to estimate systematic uncertainties

- we will start using the electron energy calibration of yesterday
- you have to think how a good selection of W events should be set up
- the electron p_T shows a Jacobian peak at $M_W/2$
  - what is the effect of the W p_T? how can you suppress it?
  - one can determine M_W from the "half-maximum" value

- you will use MC templates with different W mass values
- then a calibration curve is created and a "data" measurement done
- eventually you should look at possible systematics from the W p_T estimated with the Z decay events

Half-maximum Method

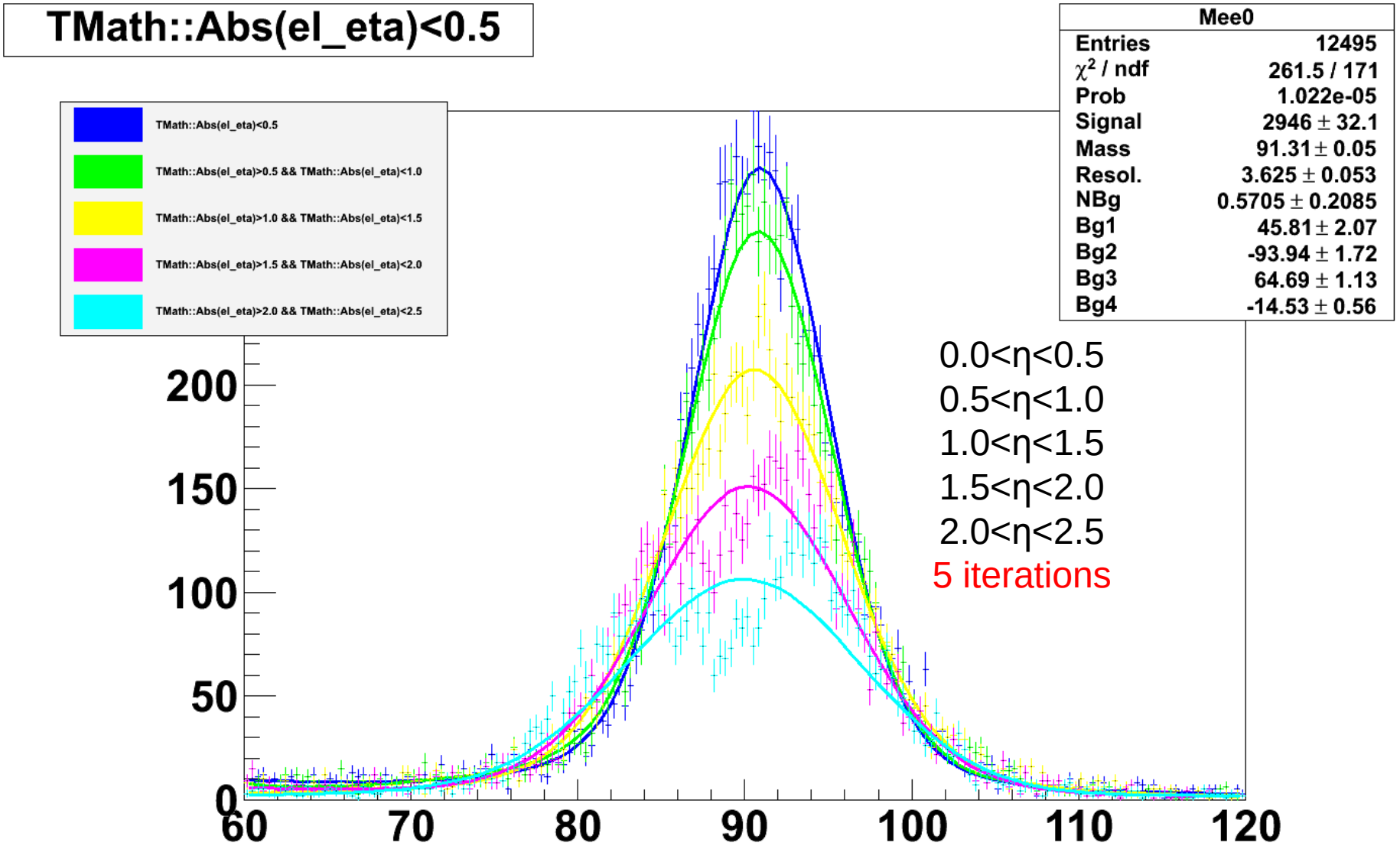


p_T at ½ max
→ W mass estimator

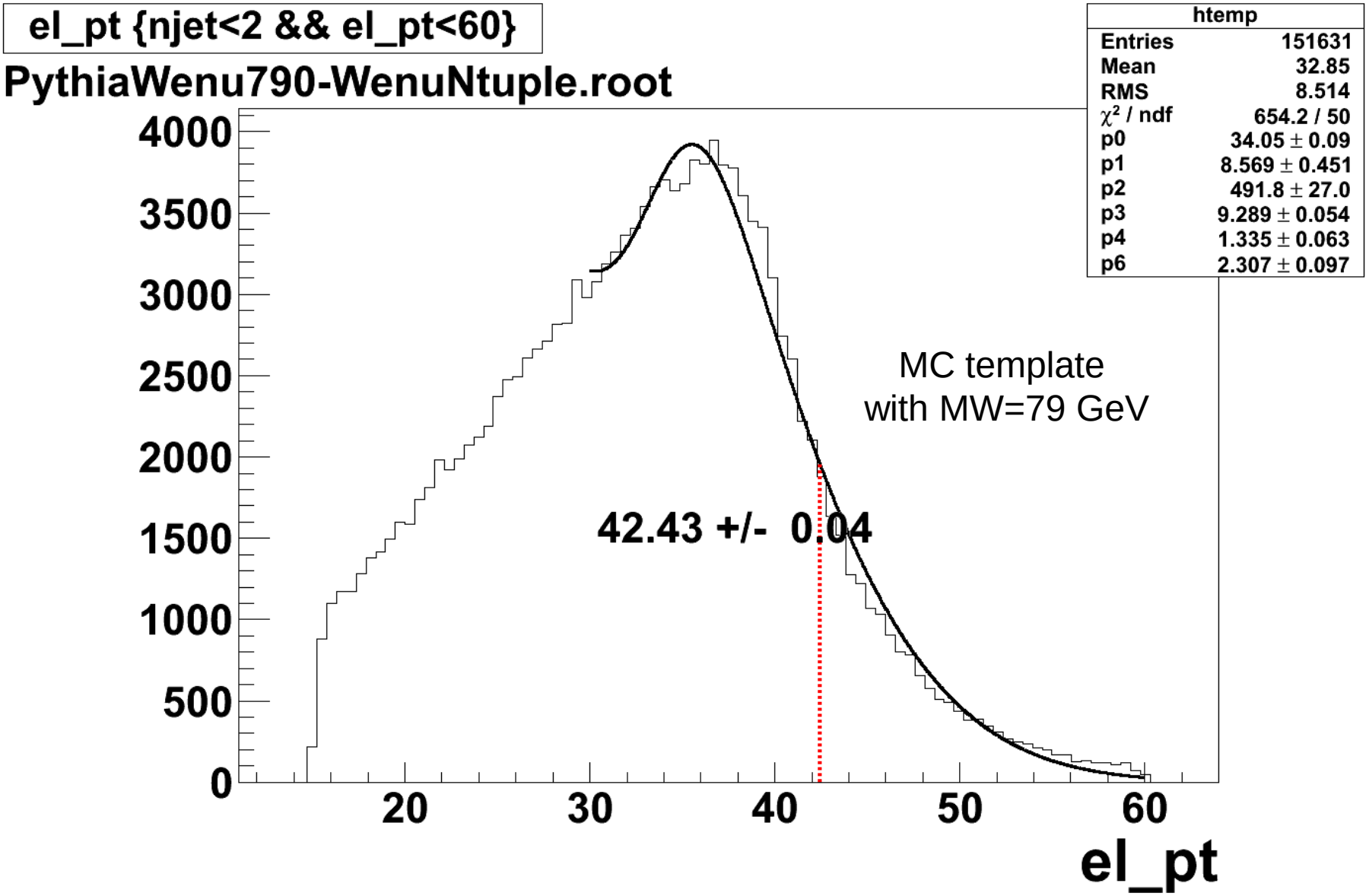
Solutions of part II

```
#include "math.h"

double ElecCalib(double e_raw, double pt, double eta, double phi,
                 double etiso, double eoverp, double drjet)
{
  // useable variables
  //  e_raw = raw energy
  //     pt = transverse momentum
  //    eta = pseudorapidity
  //    phi = azimuthal angle
  //  etiso = transverse energy
  // eoverp = E/p
  //  drjet = minimal delta R of jets


  double energy = e_raw;
  double mZ = 91.2;

  // ============ energy calibration ================================================================

  if      ( fabs(eta) < 0.5 ) energy = e_raw * mZ/89.26 * mZ/92.40 * mZ/92.07 * mZ/91.67 * mZ/91.47;
  else if ( fabs(eta) < 1.0 ) energy = e_raw * mZ/88.13 * mZ/91.68 * mZ/91.63 * mZ/91.32 * mZ/91.37;
  else if ( fabs(eta) < 1.5 ) energy = e_raw * mZ/86.45 * mZ/90.29 * mZ/90.97 * mZ/91.09 * mZ/91.15;
  else if ( fabs(eta) < 2.0 ) energy = e_raw * mZ/83.95 * mZ/88.12 * mZ/89.86 * mZ/90.57 * mZ/90.89;
  else if ( fabs(eta) < 2.5 ) energy = e_raw * mZ/80.39 * mZ/86.49 * mZ/89.40 * mZ/90.50 * mZ/90.86;

  // if      ( fabs(eta) < 1.5 ) energy = e_raw * mZ/mObserved;
  // else if ( fabs(eta) > 2.0 ) energy = e_raw * mZ/mObserved;

  // ================================================================================================


  return energy;
}
```

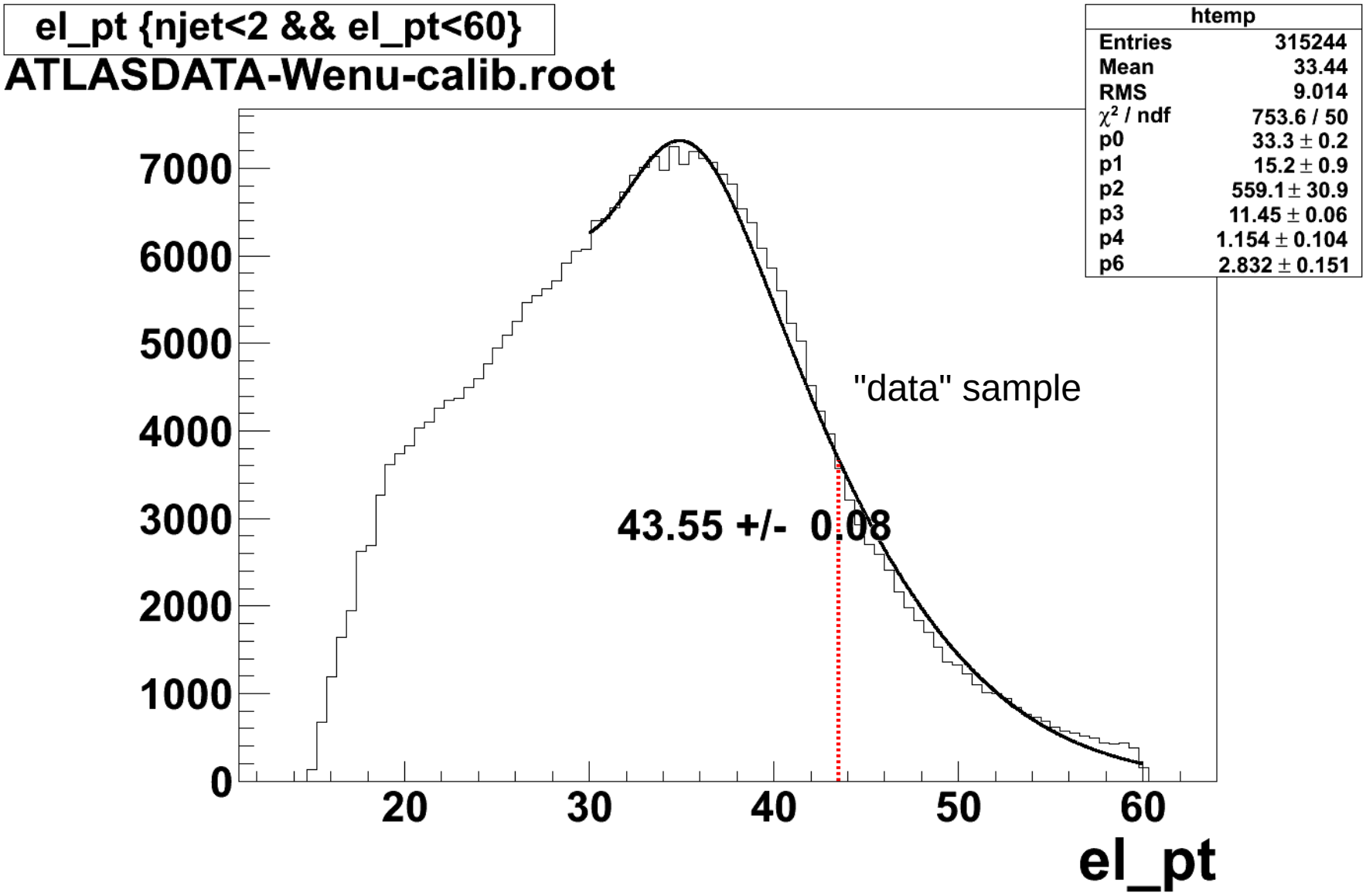
Solutions of part II



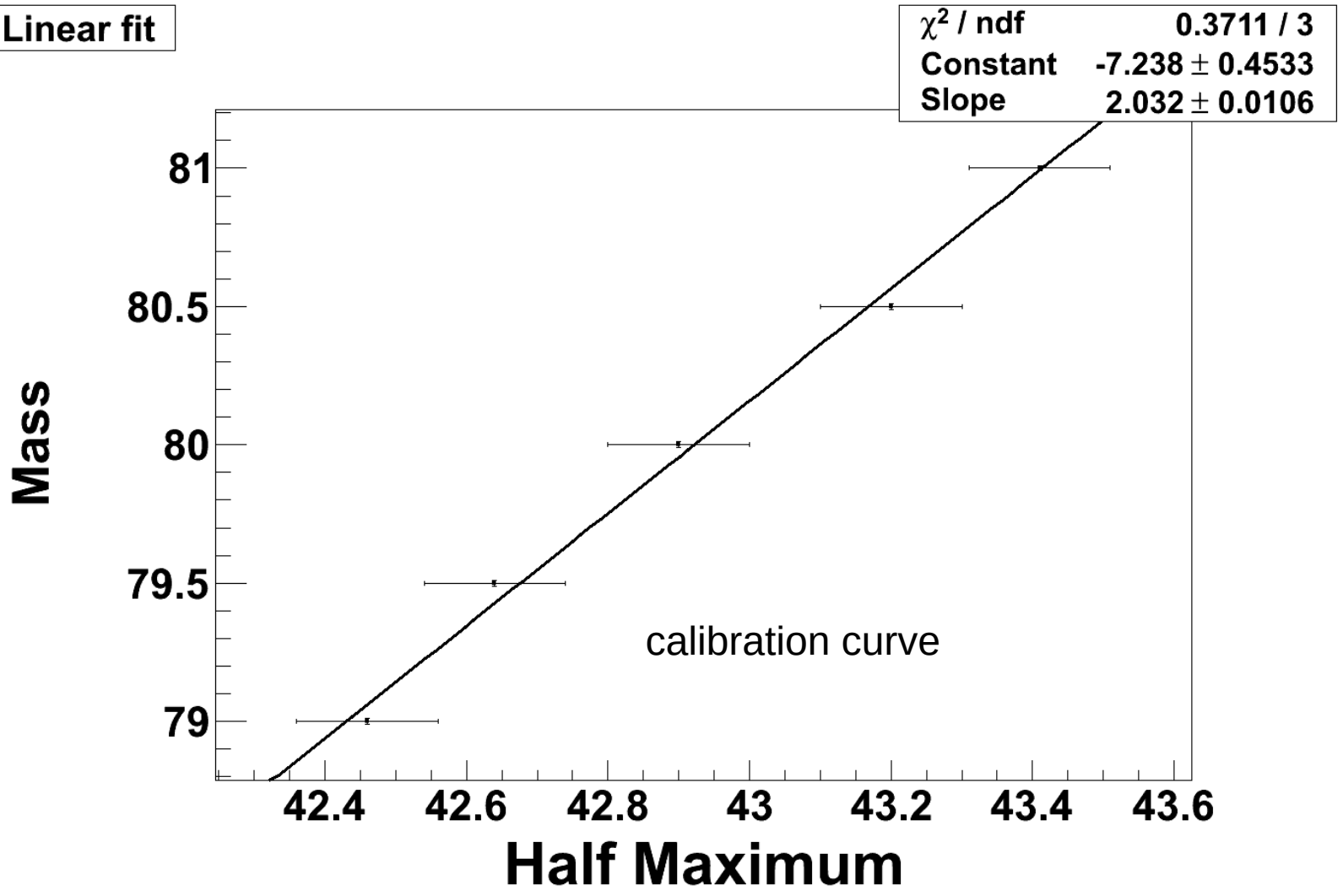
Solutions of part III



Solutions of part III



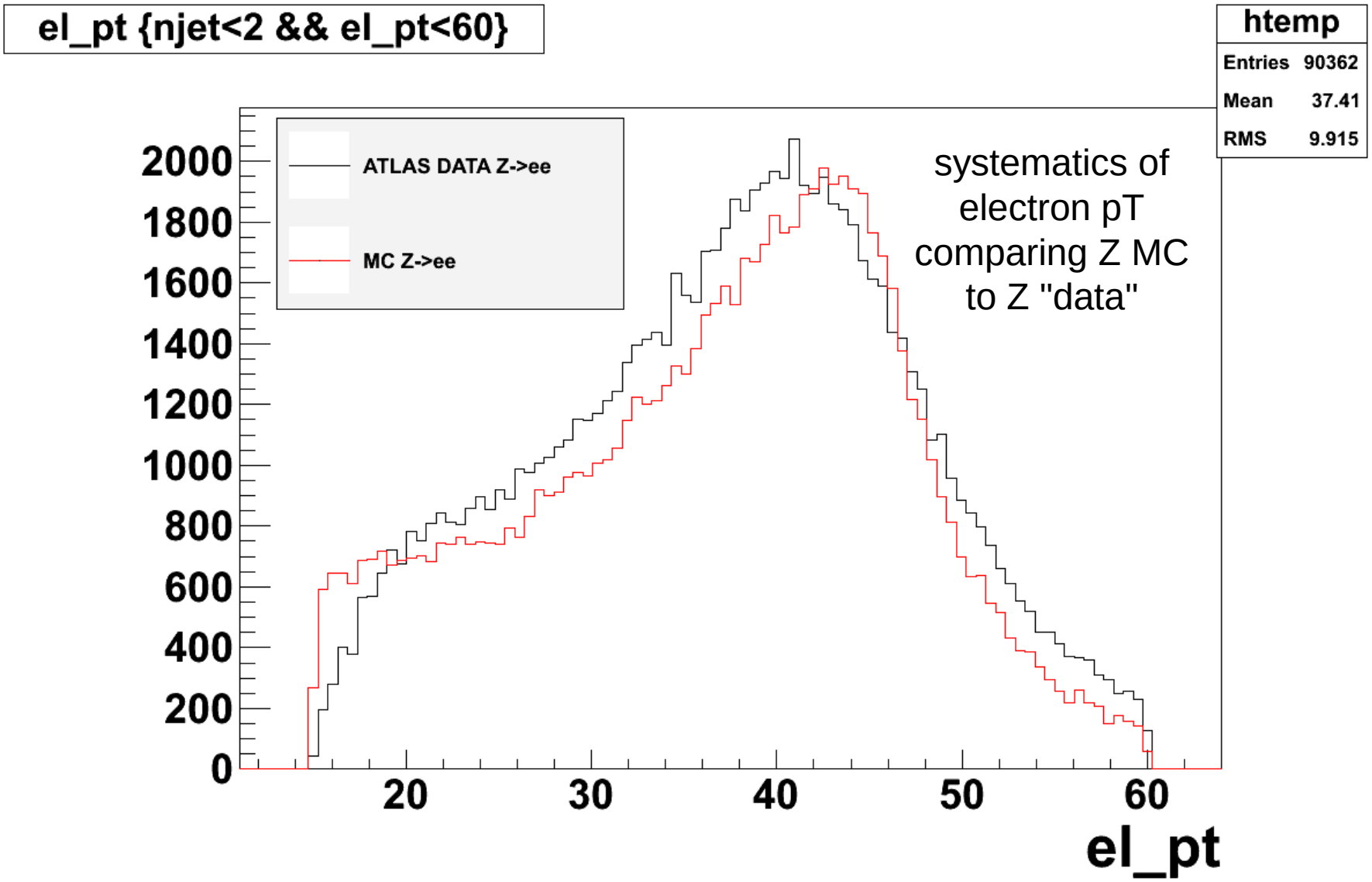
Solutions of part III



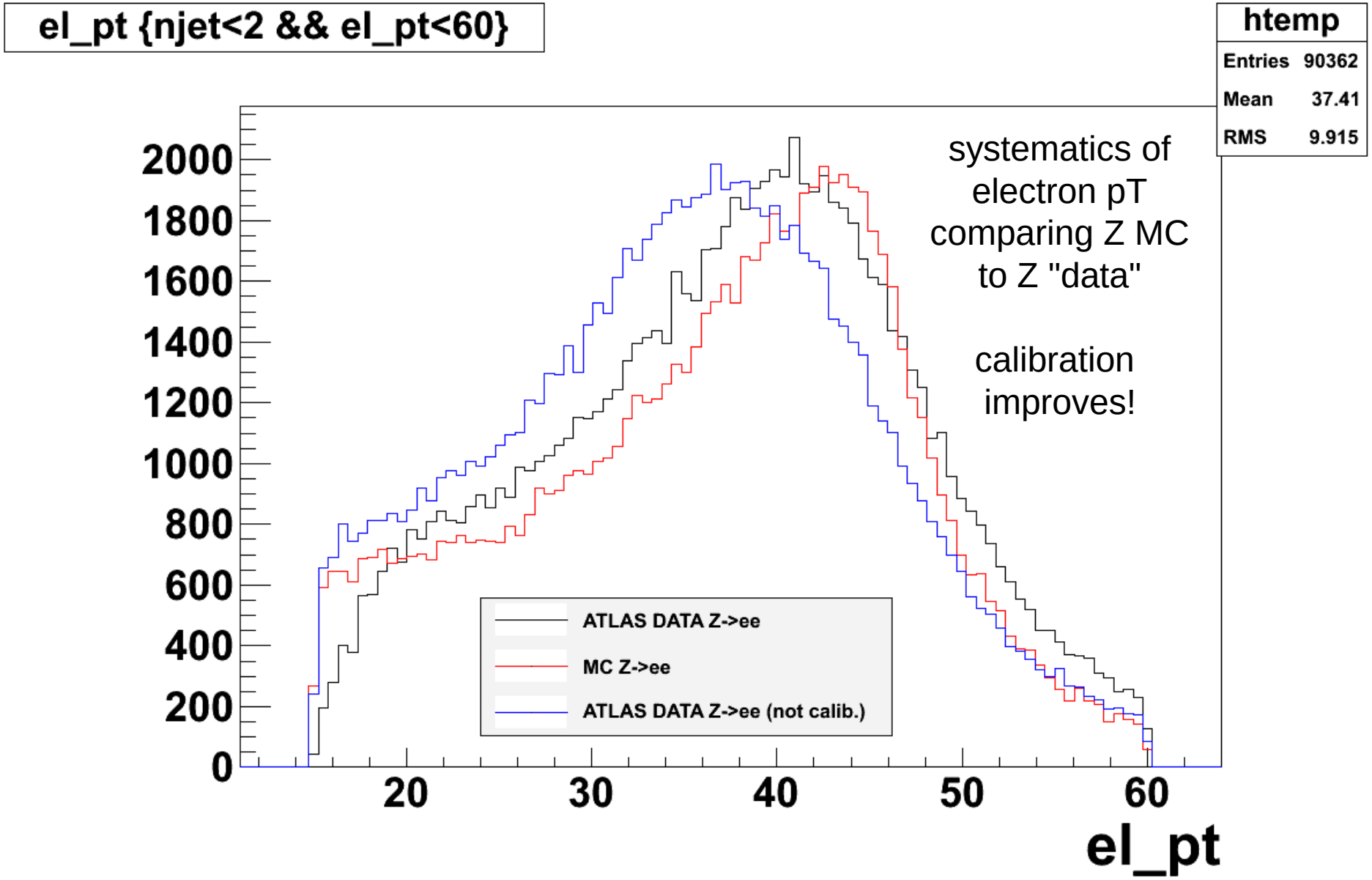
Solutions of part III

```
// Linear Fit using W/Z mass and the half maximum of the jacobian peak
//
//

#include "TGraphErrors.h"
#include "TCanvas.h"
#include "TF1.h"
#include "TMath.h"
#include "TAxis.h"


void linear_fit(Double_t HMdata=-1)
{
  // ============ put in here fit parameter =================

  // number of entries for fitting
  const int n=5;
  //const int n=0;

  // HalfMaximum
  Double_t x[n]={42.46, 42.64, 42.90, 43.20, 43.41};
  //Double_t x[n]={};

  // W and Z mass
  Double_t y[n]={79., 79.5, 80., 80.5, 81.};
  //Double_t y[n]={};

  // Error on HalfMaximum
  Double_t ex[n]={0.1, 0.1, 0.1, 0.1, 0.1};
  //Double_t ex[n]={};

  // Error on W mass is set to 0.1 GeV
  Double_t ey[n]; for (int i=0;i<n;++i) ey[i]=0.01;
  // ======================================================


  // make Canvas
  TCanvas *c1 = new TCanvas("c1","linear fit",200,10,600,400);
  c1->cd();

  // TGraphError
  TGraphErrors *gr = new TGraphErrors (n,x,y,ex,ey); // creates a graph with n points at positions x,y with errors ex,ey
  // Title, Axis and Markers
  gr->SetTitle("Linear fit");
  gr->GetXaxis()->SetTitle("Half Maximum");
```

Solutions of part III



Solutions of part III



Solutions of part III

```
// Linear Fit using W/Z mass and the half maximum of the jacobian peak
//
//

#include "TGraphErrors.h"
#include "TCanvas.h"
#include "TF1.h"
#include "TMath.h"
#include "TAxis.h"


void linear_fit(Double_t HMdata=-1)
{
  // ============ put in here fit parameter =================

  // number of entries for fitting
  const int n=5;
  //const int n=0;

  // HalfMaximum
  Double_t x[n]={42.46, 42.64, 42.90, 43.20, 43.41};
  //Double_t x[n]={};

  // W and Z mass
  Double_t y[n]={79., 79.5, 80., 80.5, 81.};
  //Double_t y[n]={};

  // Error on HalfMaximum
  Double_t ex[n]={0.3, 0.3, 0.3, 0.3, 0.3};
  //Double_t ex[n]={};

  // Error on W mass is set to 0.1 GeV
  Double_t ey[n]; for (int i=0;i<n;++i) ey[i]=0.1;
  // =======================================================
```

Solutions of part III

```
// make Canvas
TCanvas *c1 = new TCanvas("c1","linear fit",200,10,600,400);
c1->cd();

// TGraphError
TGraphErrors *gr = new TGraphErrors (n,x,y,ex,ey); // creates a graph with n points at positions x,y with errors ex,ey
// Title, Axis and Markers
gr->SetTitle("Linear fit");
gr->GetXaxis()->SetTitle("Half Maximum");
gr->GetYaxis()->SetTitle("Mass");
gr->GetXaxis()->CenterTitle();
gr->GetYaxis()->CenterTitle();
gr->SetMarkerStyle(21);
gr->SetMarkerSize(0.5);



// Fit Function
TF1 *fit = new TF1("fit", "pol1", 0.,10.);       // contructor of fit function (using a Polynomial of degree 1)
fit->SetParameters(0.0 ,1.0);                    // start values of parameters of Pol1 (first: constant term, second: l
fit->SetParNames("Constant","Slope");            // set names of parameters

// fit points using the fit function
gr->Fit("fit", "E");
gr->Draw("AP");       // draw pointsTest, errors and the fit
gr->Print();          // print results


// print fitted mass for given half maximum
if (HMdata!=-1) {
  TF1 *fitfunc = gr->GetFunction("fit");     // get fitted function from graph
  Double_t mass = fitfunc->Eval(HMdata);     // evaluate your half maximum value from the ATLAS dataset W->enu
                                             // this returns the mass for this half maximum

  cout << endl;
  cout << "============================================================" << endl << endl;
  cout << "for a half maximum of " << HMdata << " the fitted mass is " << endl << endl;
  cout << " --> "                  << mass   << " <-- "                 << endl << endl;
  cout << "============================================================" << endl << endl;
}
```