Next Steps

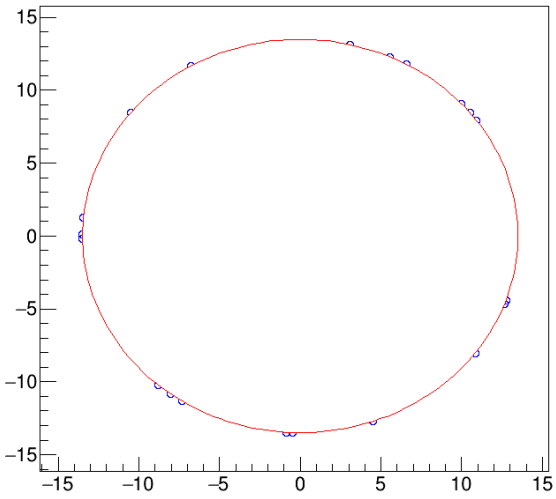
Tasks:

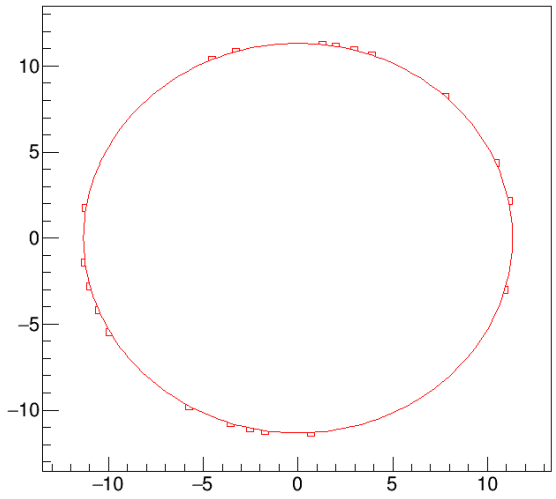
- ☐ Sklearn fit
- ☐ Problem with momentum

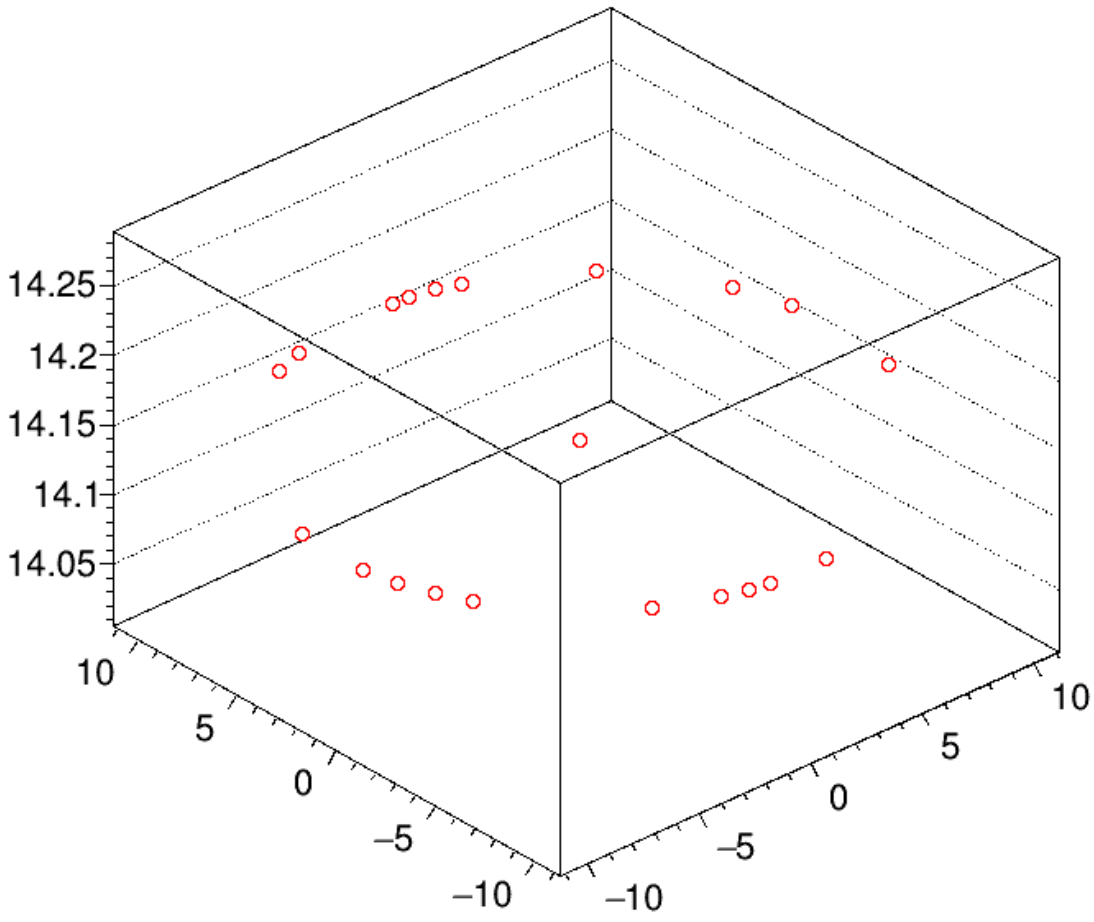
01.12 Updates

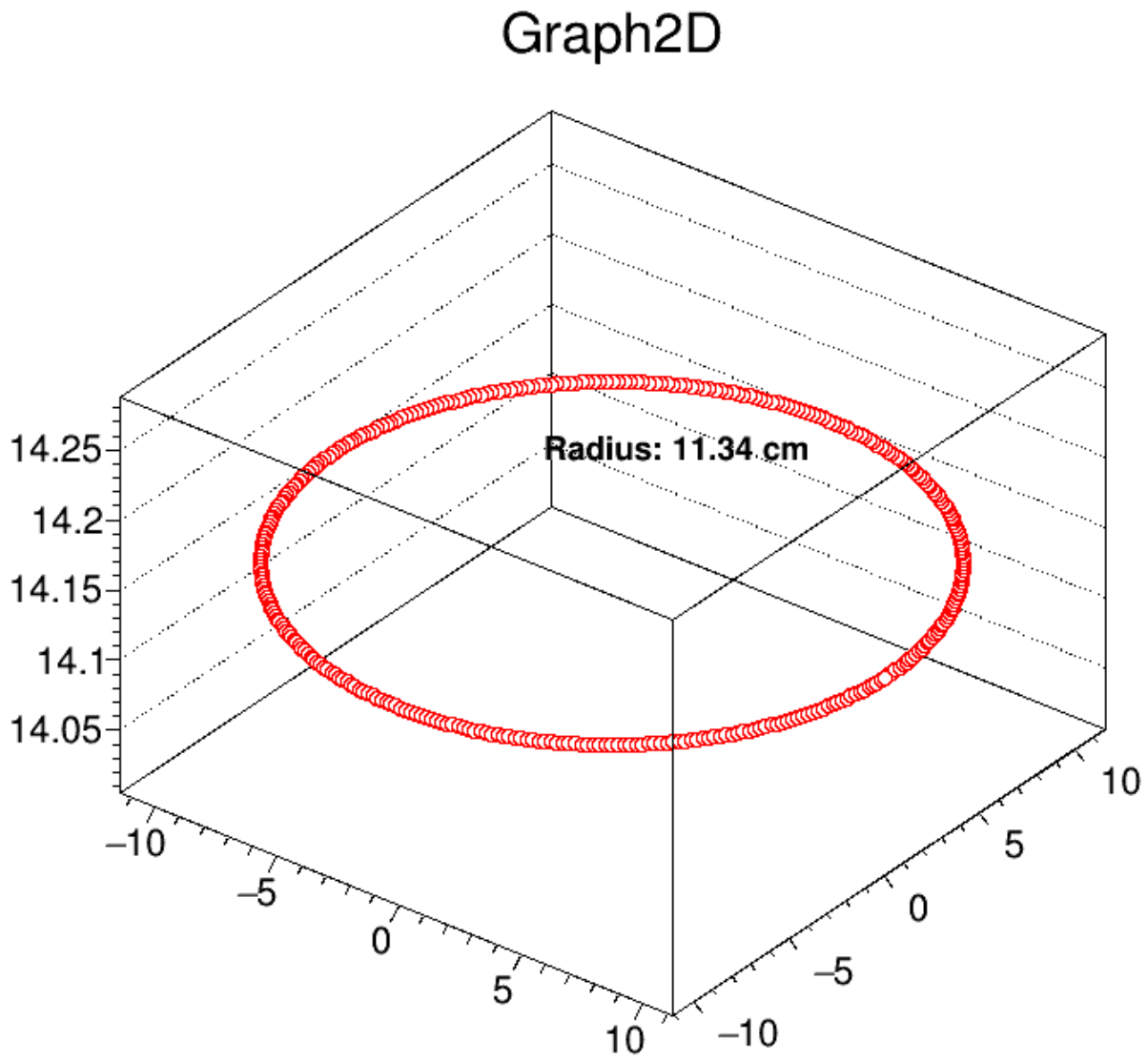
Tasks:

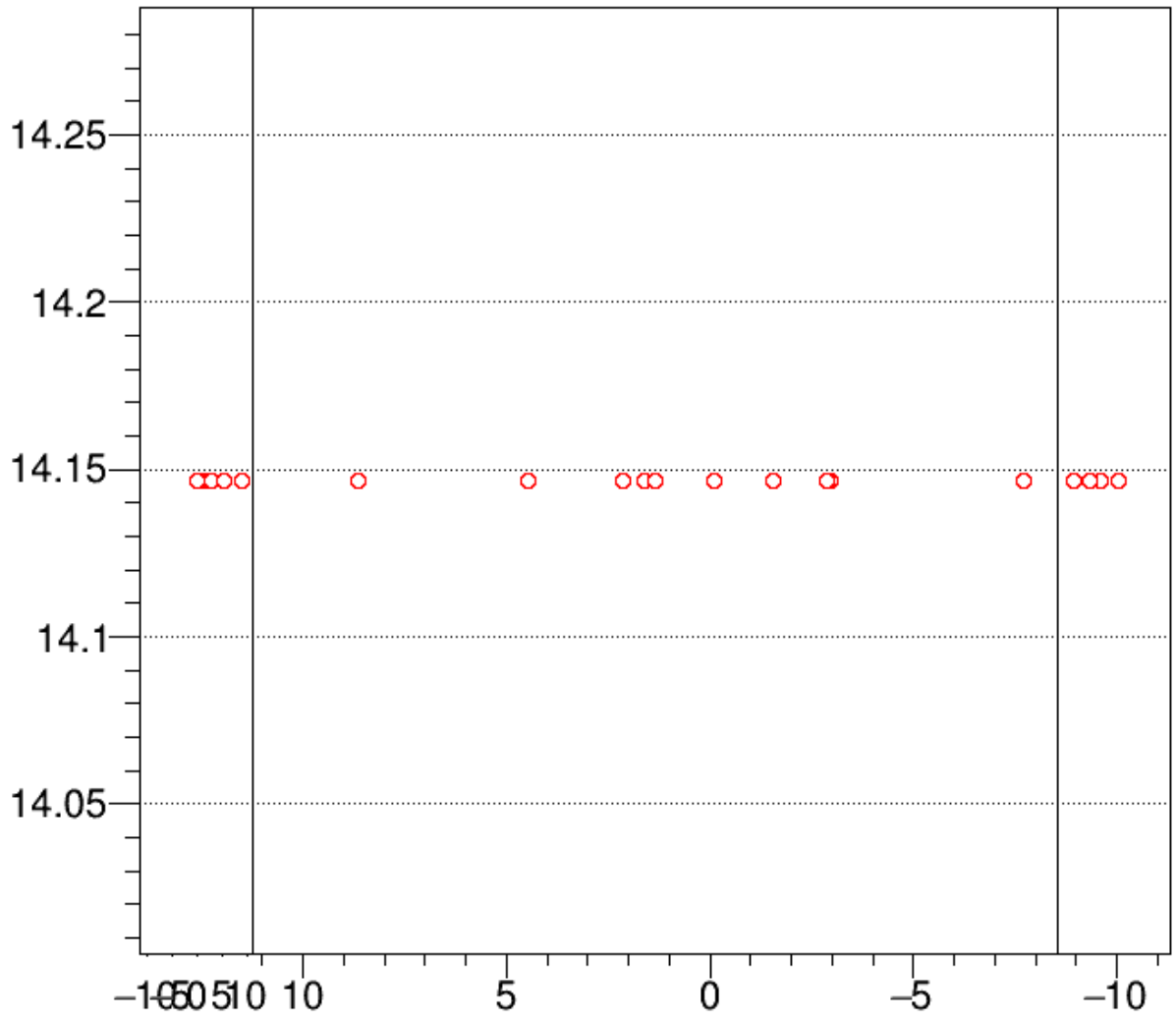
- ☑ ~~Find cherenkov angle (assume 45 cm, between cherenkov source and detector plane)~~
- ☑ ~~Find particle momentum (when we will know refractive index)~~
- ☑ ~~Check out ch.angle problem~~
- ☑ ~~Root fitting for momentum blue/red (must be the same)~~

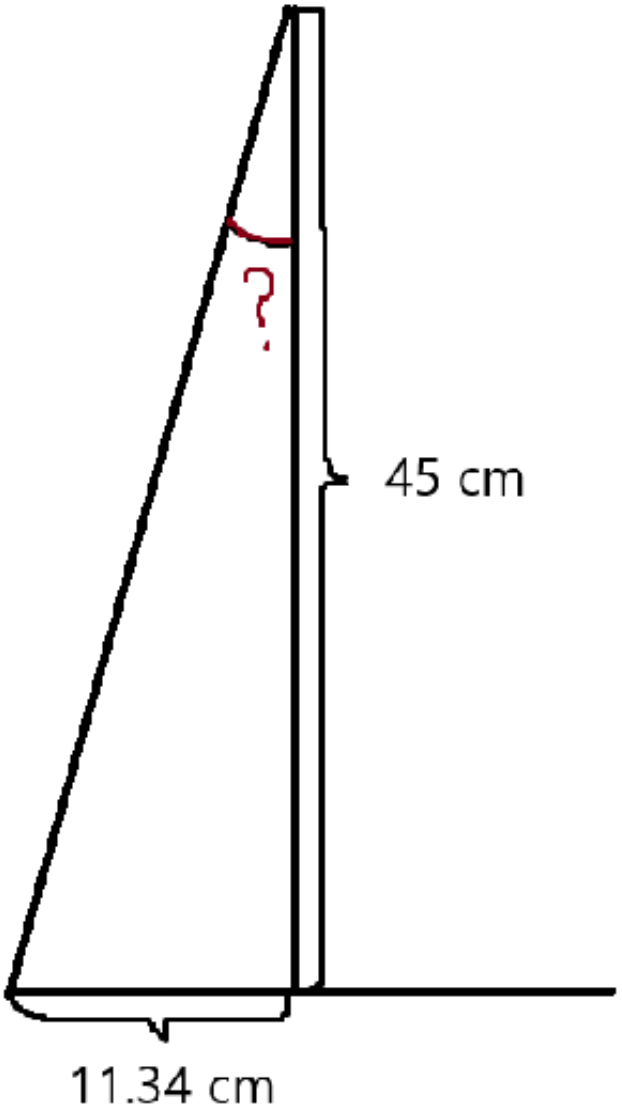

15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15


10
5
0
−5
−10
−10
−5
0
5
10


14.25
14.2
14.15
14.1
14.05
10
5
0
−5
−10
−10
−5
0
5
10


Graph2D
Radius: 11.34 cm
14.25
14.2
14.15
14.1
14.05
−10
−5
0
5
10
10
5
0
−5
−10


14.25
14.2
14.15
14.1
14.05
10
5
0
−5
−10



arctg(cherenkov ring radius/distance to the detector plane)

Refractive Index: 1.33:

Fit Results for Graph_0: x0=1.19467533582203e-05, y0=9.363214754972836e-06, **R**=13.48633579106043
Cherenkov Angle : 0.2912 radians, 16.68 degrees

Velocity: 2.35e+10 cm/s
Momentum : 1.16e+13 MeV/c

Fit Results for Graph_1:
x0=-0.111122938818861 55, **y0**=0.2211682215179589
R=11.342242845486684
Cherenkov Angle : 0.2469 radians, 14.15 degrees
Velocity: 2.33e+10 cm/s
Momentum : 3.25e+12 MeV/c

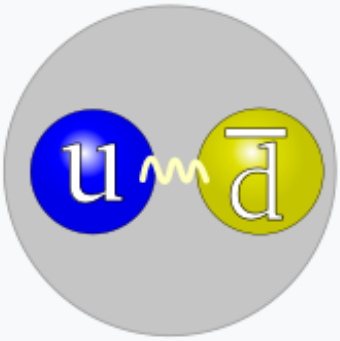
Refractive Index: 1.05:

Fit Results for Graph_0:
x0=1.19467533582203e-05, **y0**=9.363214754972836e-06
R=13.48633579106043
Cherenkov Angle : 0.2912 radians, 16.68 degrees
Velocity: 2.98e+10 cm/s
Momentum : 1.47e+13 MeV/c

Fit Results for Graph_1: x0=-0.111122938818861 55,
y0=0.2211682215179589, **R**=11.342242845486684
Cherenkov Angle : 0.2469 radians, 14.15 degrees
Velocity: 2.95e+10 cm/s
Momentum : 4.11e+12 MeV/c

Kaon/Pion ?

Pion



The quark structure of the positively charged pion.

Composition	π^+: $u\bar{d}$ π^0: $u\bar{u}$ or $d\bar{d}$ π^-: $d\bar{u}$
Statistics	Bosonic
Family	Mesons
Interactions	Strong, weak, electromagnetic, and gravity
Symbol	π^+, π^0, and π^-
Antiparticle	π^+: π^- π^0: self
Theorized	Hideki Yukawa (1935)
Discovered	$\pi^\pm$: César Lattes, Giuseppe Occhialini (1947), Cecil Powell π^0: 1950
Types	3
Mass	$\pi^\pm$: 139.570 39(18) MeV/c^2[1] π^0: 134.9768(5) MeV/c^2[1]

Kaon

Composition	K^+: $u\bar{s}$ K^0: $d\bar{s}$ K^-: $s\bar{u}$
Statistics	Bosonic
Family	Mesons
Interactions	Strong, weak, electromagnetic, gravitational
Symbol	K^+, K^0, K^-
Antiparticle	K^+: K^- K^0: $\bar{K}^0$ K^-: K^+
Discovered	1947
Types	4
Mass	$K^\pm$: 493.677 ± 0.016 MeV/c^2 K^0: 497.611 ± 0.013 MeV/c^2

The momentum of Graph_0 is significantly larger than that of Graph_1. Since momentum is directly proportional to both velocity and mass, and the velocities are relatively close, the larger momentum in Graph_0 suggests a

larger mass. Therefore, Graph_0 is more likely to be associated with kaons, which have a larger mass compared to pions. Similarly, with Graph_1

In summary:

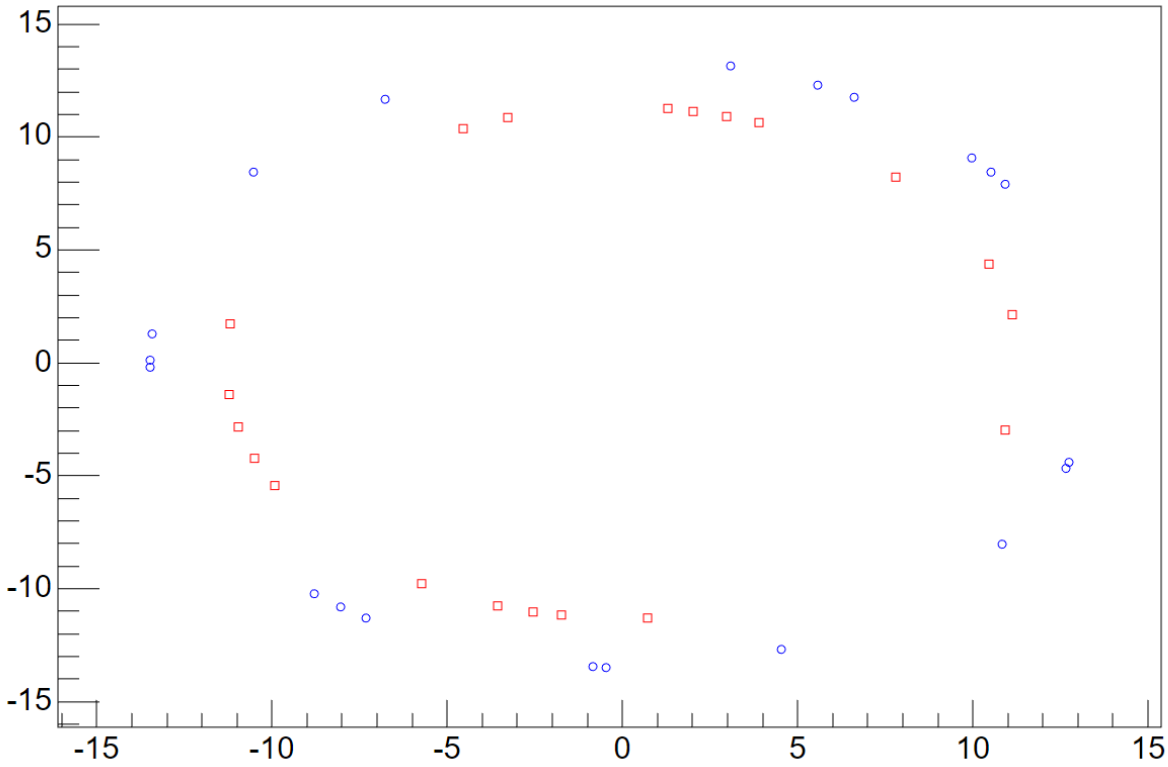
Graph_0 is more likely associated with **kaons**.
Graph_1 is more likely associated with **pions**.

12.11 Updates

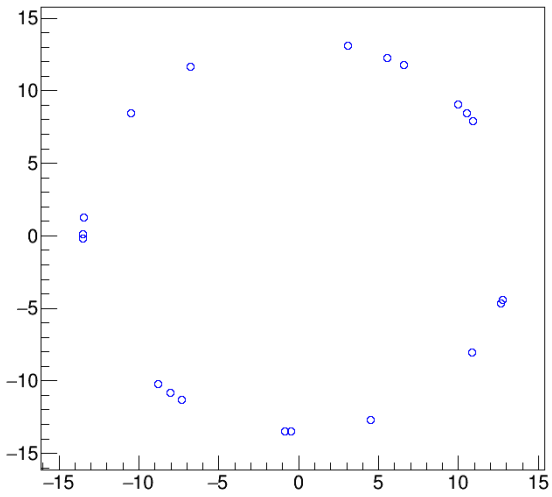
Tasks:

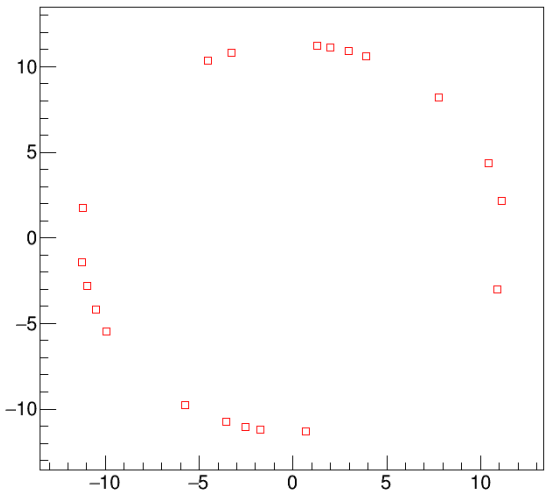
- ☑ ~~make a graph with sklearn library~~
- ☑ ~~reproduce plot from .root file~~
- ☑ ~~find radius~~

Original plot:

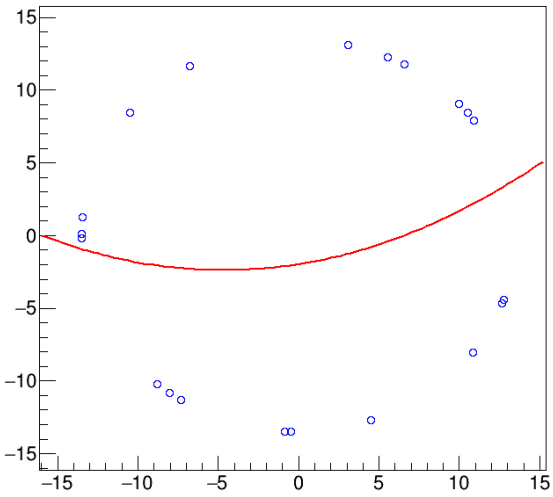


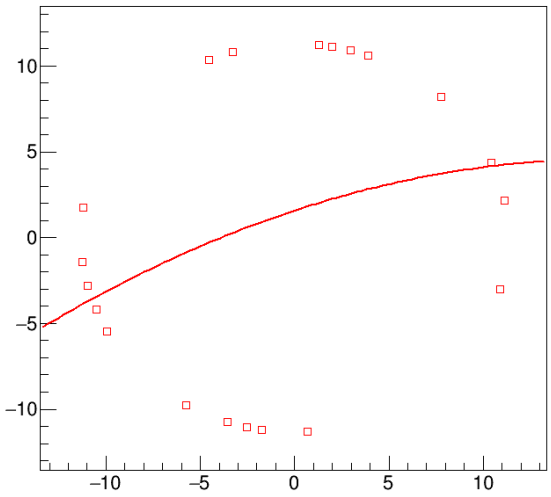
Resized and divided into two plots :





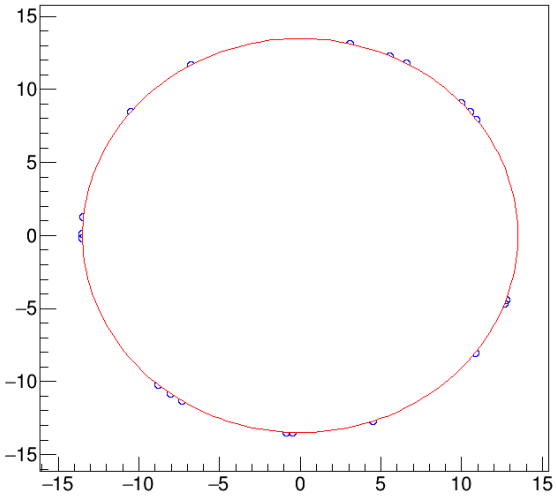
An attempt to use the usual ROOT fitting :

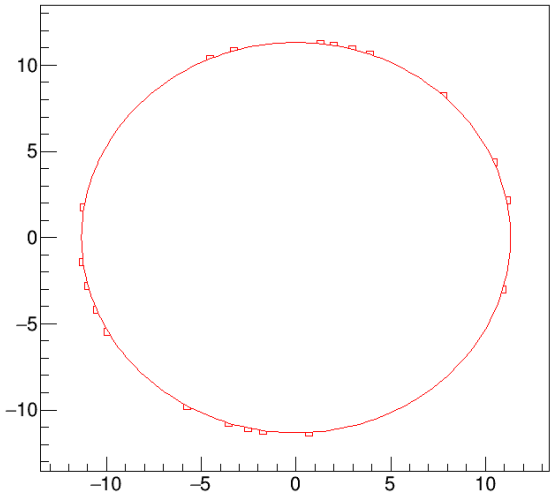




fit for circle

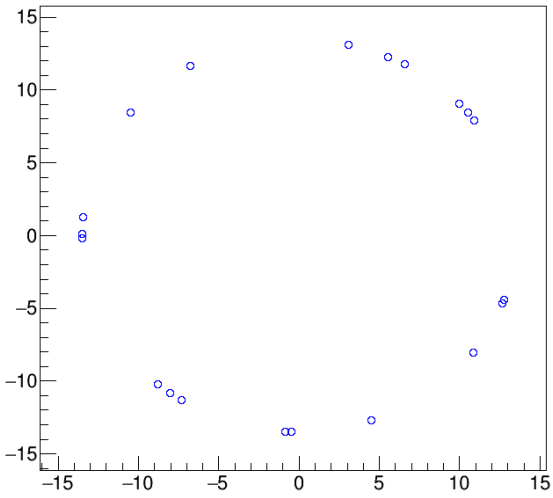
https://root.cern/doc/v608/fitCircle_8C.html

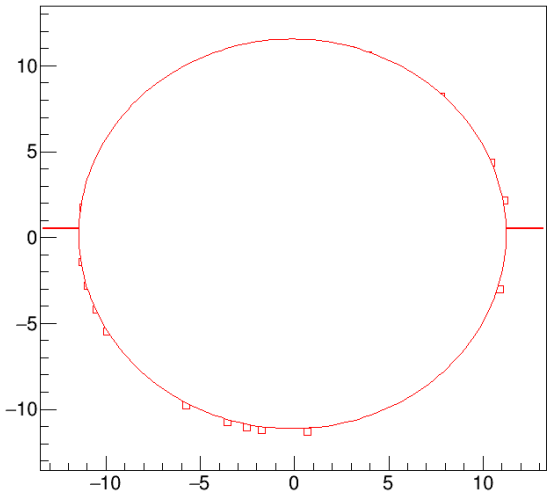




```
****************************************
Minimizer is Minuit / Migrad
MinFCN                      =  9.51822e-08
NDf                         =           0
Edm                         =  1.90364e-07
NCalls                      =          66
x0                          = -5.98829e-05   +/-   0.324678
y0                          =  8.63756e-05   +/-   0.31625
R                           =      13.4863   +/-   0.223772

****************************************
Minimizer is Minuit / Migrad
MinFCN                      =  1.64511e-07
NDf                         =           0
Edm                         =  3.29019e-07
NCalls                      =          70
x0                          =  3.16537e-05   +/-   0.363602
y0                          = -5.23469e-05   +/-   0.314088
R                           =      11.3227   +/-   0.228209
```

Recoded to python (have errors with visual part, but layouts parameters good):






```
Fit Results for Graph_0: x0=-7.517222209616141e-07, y0=-1.6335566746291795e-06, R=13.486335513582468
Fit Results for Graph_1: x0=-0.11103709360207031, y0=0.2211162693726172, R=11.342211054661147
Press Enter to exit
```

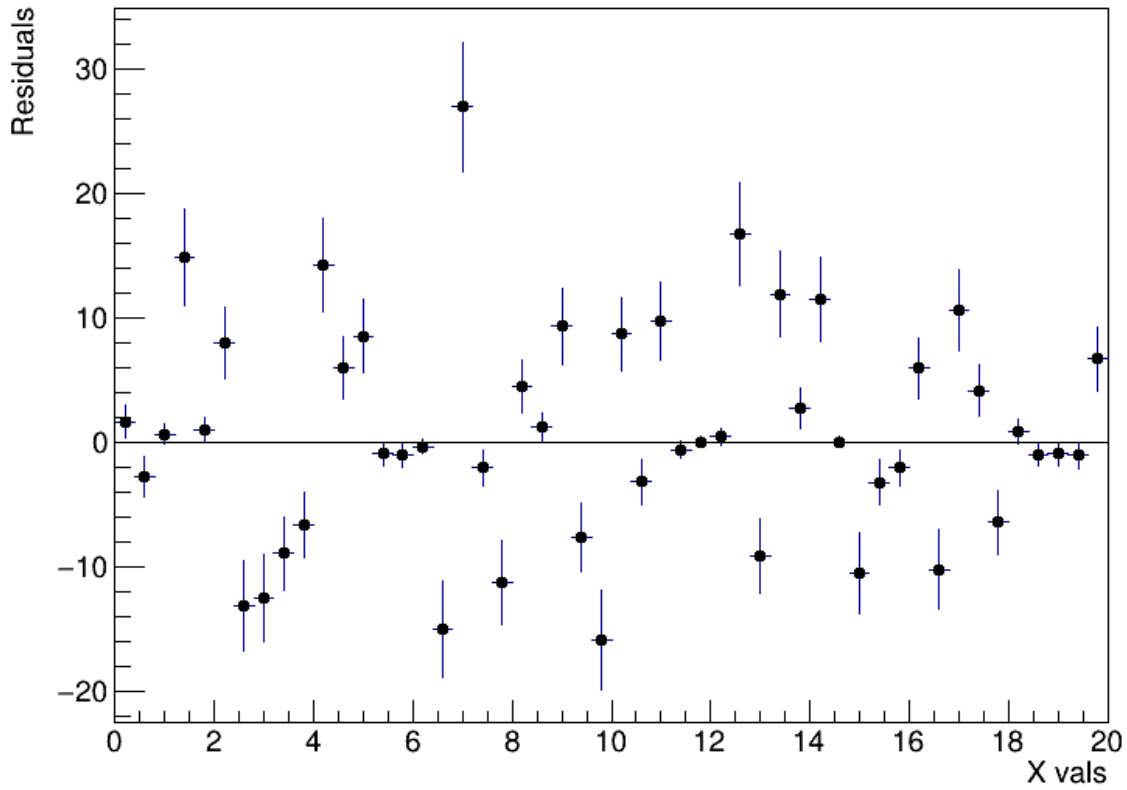
13.10 Updates

Tasks:

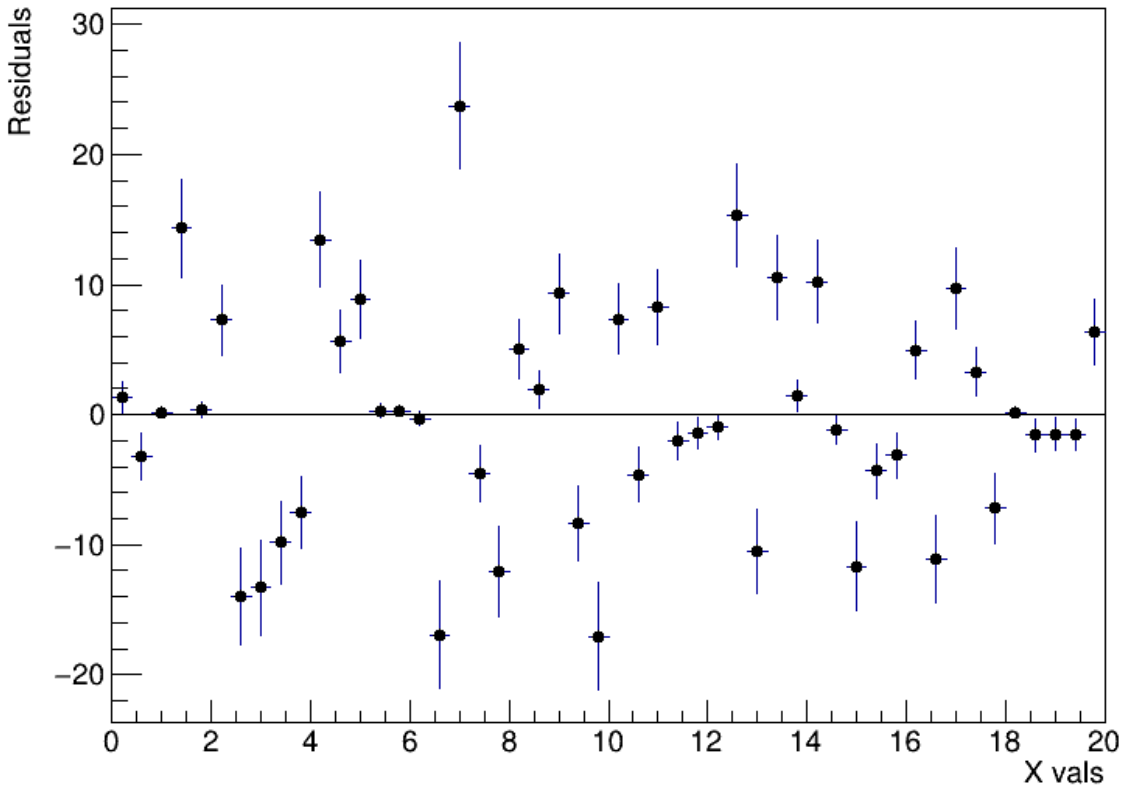
- ☑ ~~Generate the ratio plot (scipy/ROOT) or the residue (scipy-ROOT) plot~~
- ☑ ~~Make a presentation about scipy curve_fit~~
- ☑ ~~Add legend to SciPy fit~~
- ☑ ~~Build a table showing the difference between the fit parameters~~

Residual Plots

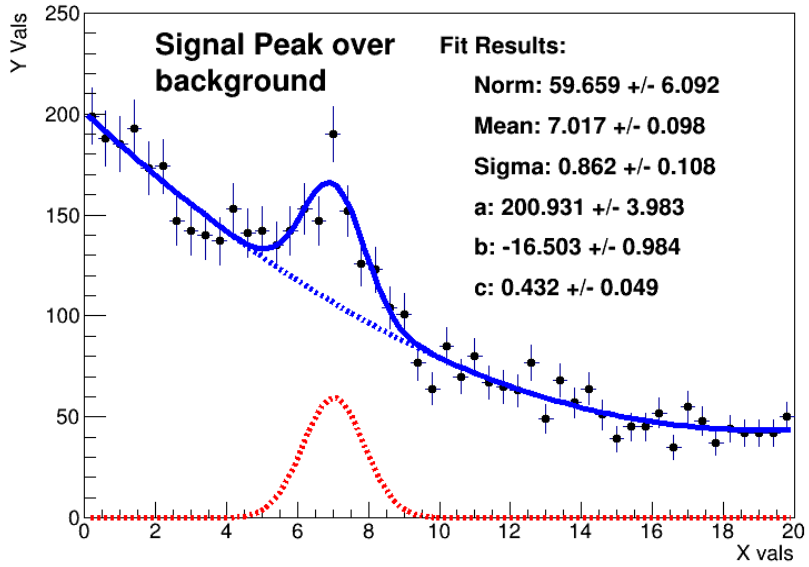
ROOT:

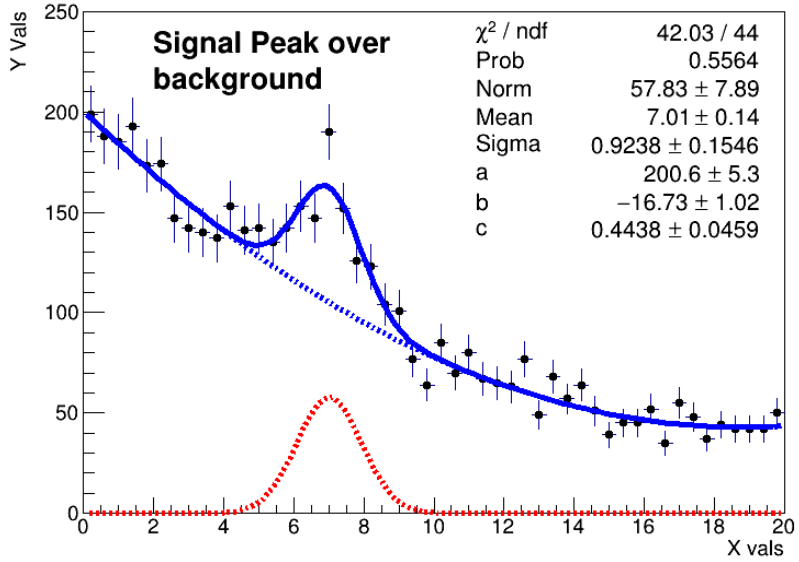


CURVE_FIT:


Residuals
30
20
10
0
−10
−20
0
2
4
6
8
10
12
14
16
18
20
X vals

Comparing Fit Parameters:





Root fit

SciPy fit

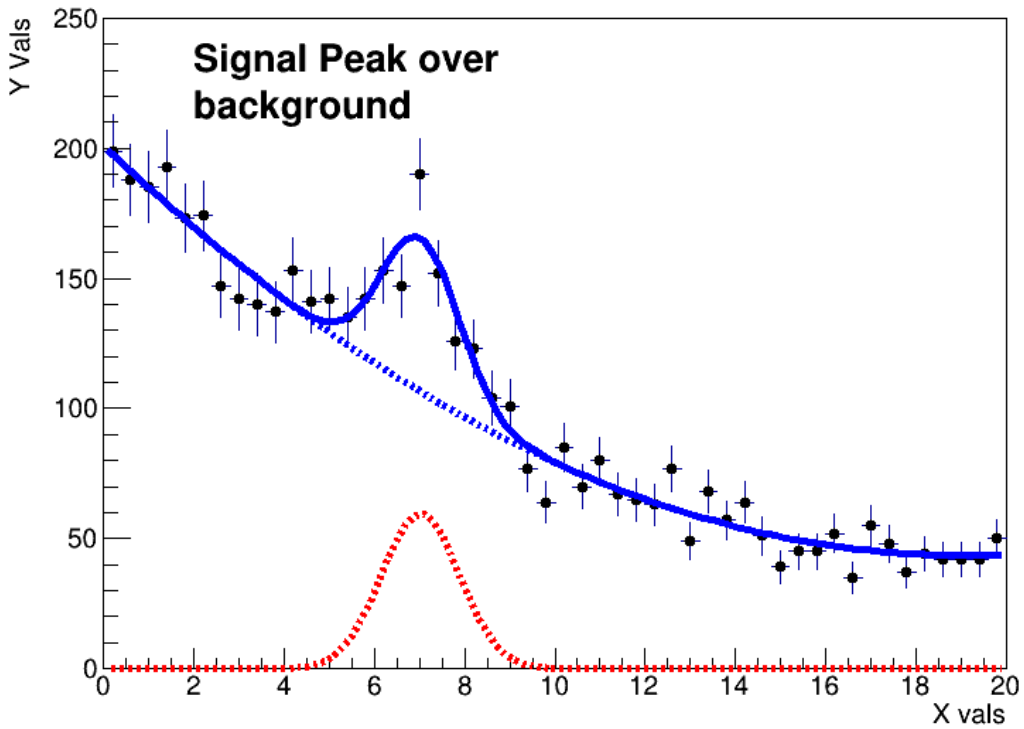
16.09 Updates

Tasks:

- ☑ ~~Resolve problem with Fit (it was actually ROOT)~~
- ☑ ~~Delete section of using .root files in code~~
- ☑ ~~Save two plots (with Root fit and SciPy fit) in one canvas~~
- ☐ Questions (how to show fitted parameters info when I use SciPy fit / book questions)

[Google Drive (here's where all the code and graphics are)]

Code, which use **SCIPY** library for fit:

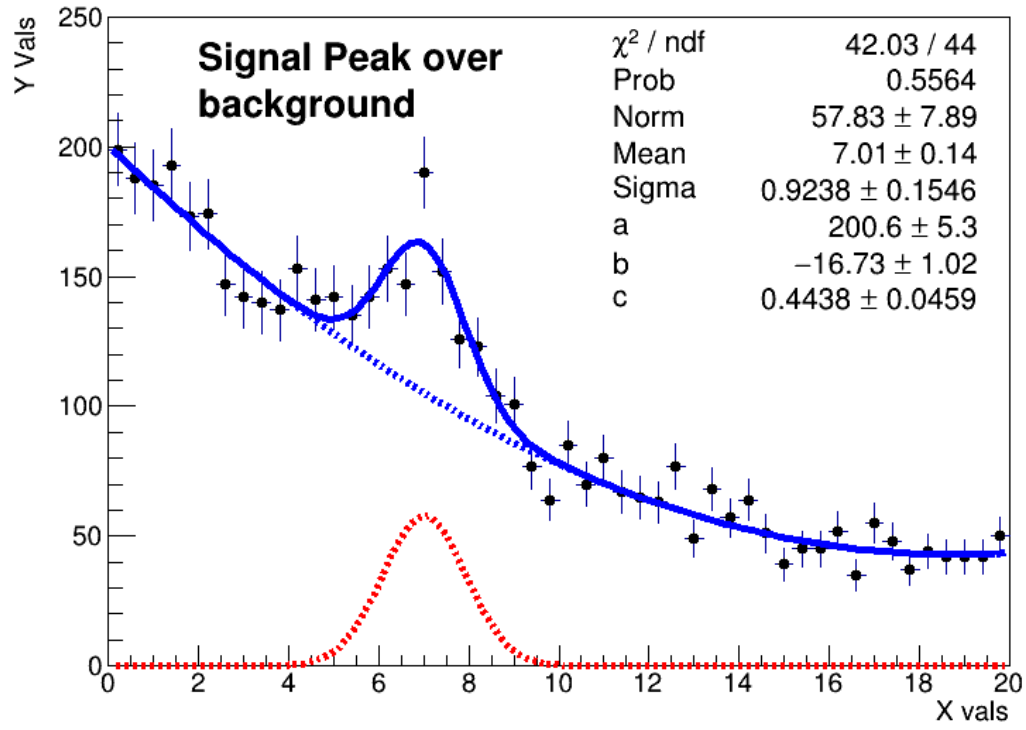



```
import ROOT as R
import numpy as np
from scipy.optimize import curve_fit
```

```
def format_line(line, col, sty):
    line.SetLineWidth(5)
    line.SetLineColor(col)
    line.SetLineStyle(sty)

def gausppar_func(vars, pars): # gaussian + parabola
    # f(x) = A * exp(-(x - B)^2 / (2 * C^2)) + D + E * x + F * x^2 - gaussian function augmented by a second degree polynomial
    return pars[0] * np.exp(-(vars[0] - pars[1])**2 / (2 * pars[2]**2)) + pars[3] + pars[4] * vars[0] + pars[5] * vars[0]**2

def gaussian_func(x, norm, mean, sigma, a, b, c):
    return norm * np.exp(-(x - mean)**2 / (2 * sigma**2)) + a + b * x + c * x**2

def scipy_fit(histo, initial_params):    x_vals = np.array([histo.GetBinCenter(i) for i in range(1, histo.GetNbinsX() + 1)])
    y_vals = np.array([histo.GetBinContent(i) for i in range(1, histo.GetNbinsX() + 1)])

    optimized_params, _ = curve_fit(gaussian_func, x_vals, y_vals, p0=initial_params)

    return optimized_params

def macro8():

    R.gStyle.SetOptTitle(0)
    R.gStyle.SetOptStat(0)
    R.gStyle.SetOptFit(1111)
    R.gStyle.SetStatBorderSize(0)
    R.gStyle.SetStatX(0.89)
    R.gStyle.SetStatY(0.89)


    parabola = R.TF1("parabola", "[0]+[1]*x+[2]*x**2", 0, 20) # f(x)=a+bx+cx^2
    format_line(parabola, R.kBlue, 2)

    gaussian = R.TF1("gaussian", "[0]*TMath::Gaus(x,[1],[2])", 0, 20)
    format_line(gaussian, R.kRed, 2)

    gausppar = R.TF1("gausppar", gausppar_func, 0, 20, 6)
    a = 15
    b = -1.2
    c = 0.03
    norm = 4
    mean = 7
    sigma = 1

    # initial parameters to match Gaussian
    gausppar.SetParameters(norm, mean, sigma, a, b, c)
    gausppar.SetParNames("Norm", "Mean", "Sigma", "a", "b", "c")

    format_line(gausppar, R.kBlue, 1)

    histo = R.TH1F("histo", "Signal plus background;X vals;Y Vals", 50, 0, 20)
    histo.SetMarkerStyle(8)

    # fake data
    for i in range(1, 5001):
        histo.Fill(gausppar.GetRandom())
```

```
    gausppar.SetParameter(0, 50)
    gausppar.SetParameter(1, 6)

    npar = gausppar.GetNpar()

    for ipar in range(2, npar):
        gausppar.SetParameter(ipar, 1)


    initial_params = [100, 6, 1, 1, 1, 1]

    # SciPy fit
    fit_results = scipy_fit(histo, initial_params)

    # adjust starting values of gausppar based on the fit results
    gausppar.SetParameter(0, fit_results[0])
    gausppar.SetParameter(1, fit_results[1])
    gausppar.SetParameter(2, fit_results[2])
    gausppar.SetParameter(3, fit_results[3])
    gausppar.SetParameter(4, fit_results[4])
    gausppar.SetParameter(5, fit_results[5])

    # set gaussian and parabola values
    for ipar in range(3):
        gaussian.SetParameter(ipar, fit_results[ipar])
        parabola.SetParameter(ipar, fit_results[ipar + 3])


    # show and save
    histo.GetYaxis().SetRangeUser(0, 250)
    histo.DrawClone("PE")
    parabola.DrawClone("Same")
    gaussian.DrawClone("Same")
    gausppar.DrawClone("Same")
    latex = R.TLatex(2, 220, "#splitline{Signal Peak over}{background}")
    latex.DrawClone("Same")

    c1 = R.TCanvas("c1", "Canvas", 800, 600)
    histo.DrawClone("PE")
    parabola.DrawClone("Same")
    gaussian.DrawClone("Same")
    gausppar.DrawClone("Same")
    latex.DrawClone("Same")
    c1.SaveAs("scipy-fit.png")

    outfile = R.TFile("scipy-fit.root", "RECREATE")
    histo.Write()
    parabola.Write()
    gaussian.Write()
    gausppar.Write()
    outfile.Close()

macro8()
```

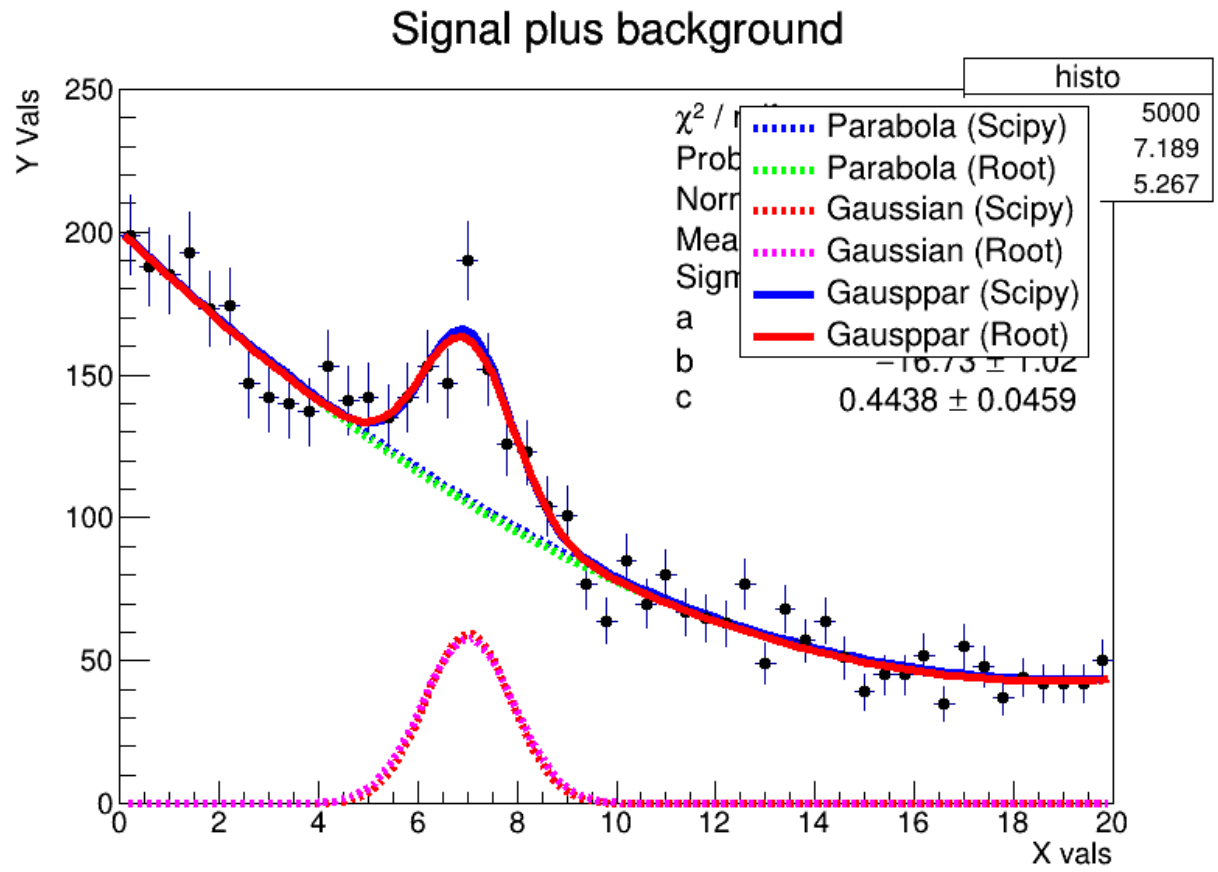
Code, which use **ROOT** library for fit:



```
import ROOT as R

def format_line(line, col, sty):
   line.SetLineWidth(5)
   line.SetLineColor(col)
   line.SetLineStyle(sty)

def the_gausppar(vars, pars):
   return pars[0] * R.TMath.Gaus(vars[0], pars[1], pars[2]) + pars[3] + pars[4] * vars[0] + pars[5] * vars[0] * vars[0]

def macro8():
   R.gStyle.SetOptTitle(0)
   R.gStyle.SetOptStat(0)
   R.gStyle.SetOptFit(1111)
   R.gStyle.SetStatBorderSize(0)
   R.gStyle.SetStatX(0.89)
   R.gStyle.SetStatY(0.89)

   parabola = R.TF1("parabola", "[0]+[1]*x+[2]*x**2", 0, 20)
   format_line(parabola, R.kBlue, 2)

   gaussian = R.TF1("gaussian", "[0]*TMath::Gaus(x,[1],[2])", 0, 20)
   format_line(gaussian, R.kRed, 2)

   gausppar = R.TF1("gausppar", the_gausppar, -0, 20, 6)
   a = 15
   b = -1.2
   c = 0.03
   norm = 4
   mean = 7
   sigma = 1
   gausppar.SetParameters(norm, mean, sigma, a, b, c)
   gausppar.SetParNames("Norm", "Mean", "Sigma", "a", "b", "c")
   format_line(gausppar, R.kBlue, 1)

   histo = R.TH1F("histo", "Signal plus background;X vals;Y Vals", 50, 0, 20)
   histo.SetMarkerStyle(8)

   for i in range(1, 5001):
      histo.Fill(gausppar.GetRandom())

   gausppar.SetParameter(0, 50)
   gausppar.SetParameter(1, 6)
   npar = gausppar.GetNpar()
   for ipar in range(2, npar):
      gausppar.SetParameter(ipar, 1)

   fitResPtr = histo.Fit(gausppar, "S")
   fitResPtr.Print()
   covMatrix = R.TMatrixDSym(fitResPtr.GetCovarianceMatrix())
   covMatrix.Print()

   for ipar in range(3):
      gaussian.SetParameter(ipar, gausppar.GetParameter(ipar))
      parabola.SetParameter(ipar, gausppar.GetParameter(ipar + 3))

   histo.GetYaxis().SetRangeUser(0, 250)
```

```
    histo.DrawClone("PE")
    parabola.DrawClone("Same")
    gaussian.DrawClone("Same")
    latex = R.TLatex(2, 220, "#splitline{Signal Peak over}{background}")
    latex.DrawClone("Same")

    c1 = R.TCanvas("c1", "Canvas", 800, 600)
    histo.DrawClone("PE")
    parabola.DrawClone("Same")
    gaussian.DrawClone("Same")
    gausppar.DrawClone("Same")
    latex.DrawClone("Same")
    c1.SaveAs("root-fit.png")

    outfile = R.TFile("root-fit.root", "RECREATE")
    histo.Write()
    parabola.Write()
    gaussian.Write()
    gausppar.Write()
    outfile.Close()

macro8()
```

Comparing ROOT and SCIPY fits



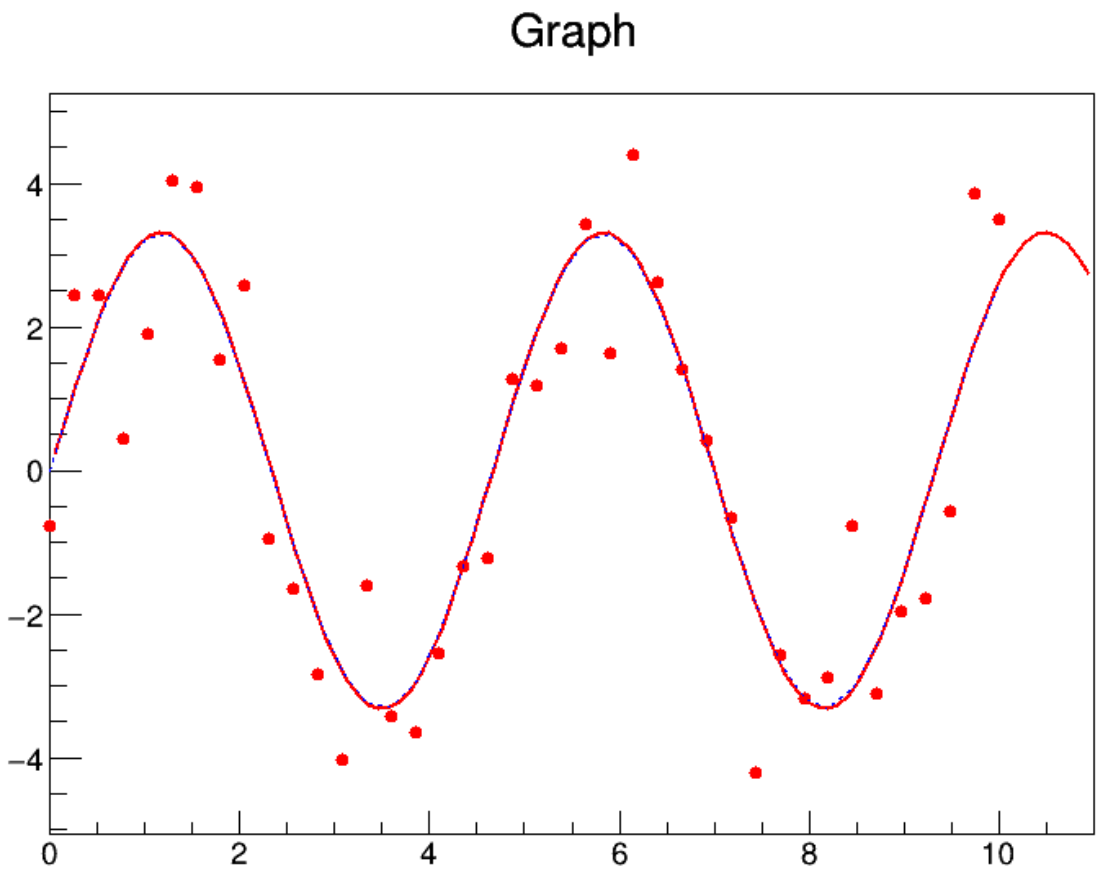
It is difficult to distinguish the difference between the two plots and the impact of root fit and scipy fit, so code was written that plots two plots on the same canva:

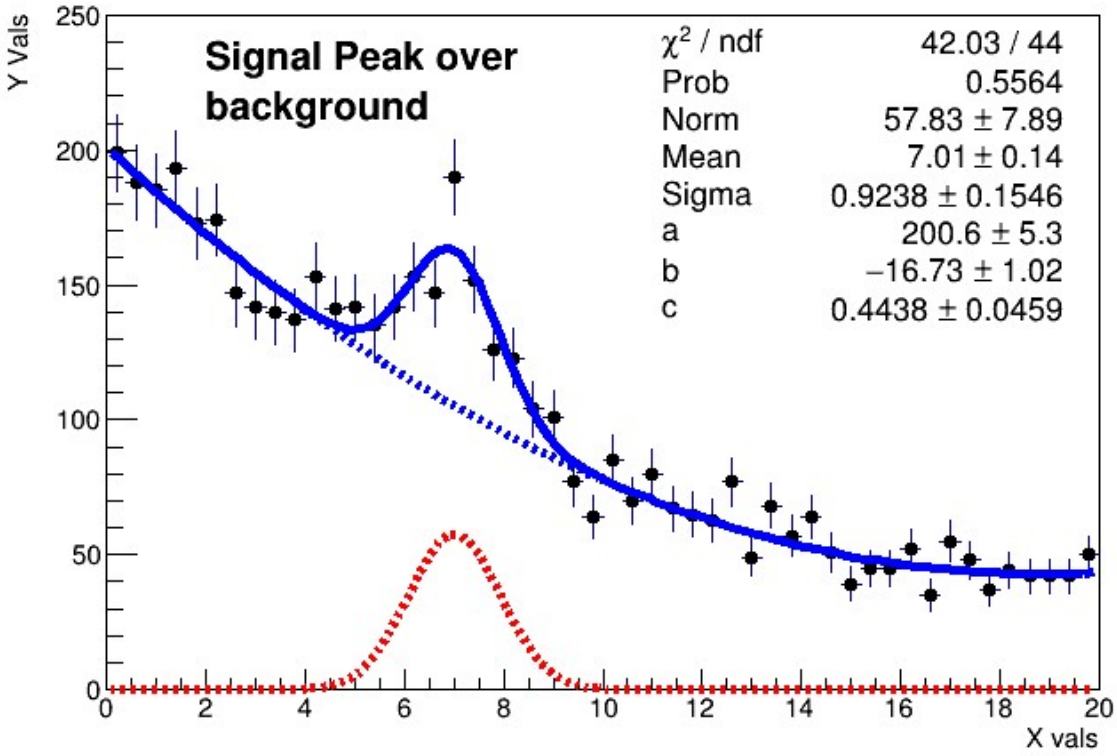
```
import ROOT as R

def format_line(line, col, sty):
    line.SetLineWidth(5)
    line.SetLineColor(col)
    line.SetLineStyle(sty)

def the_gausppar(vars, pars):
    return pars[0] * R.TMath.Gaus(vars[0], pars[1], pars[2]) + pars[3] + pars[4] * vars[0] + pars[5] * vars[0] * vars[0]

file1 = R.TFile("scipy-fit.root")
file2 = R.TFile("root-fit.root")

histo1 = file1.Get("histo")
histo2 = file2.Get("histo")

canvas = R.TCanvas("canvas", "Comparison of Fits", 800, 600)

# scipy fit
histo1.GetYaxis().SetRangeUser(0, 250)
```

```
histo1.Draw("PE")
parabola1 = file1.Get("parabola")
gaussian1 = file1.Get("gaussian")
gausppar1 = file1.Get("gausppar")
parabola1.Draw("Same")
gaussian1.Draw("Same")
gausppar1.Draw("Same")

# root fit
histo2.Draw("PE Same")
parabola2 = file2.Get("parabola")
gaussian2 = file2.Get("gaussian")
gausppar2 = file2.Get("gausppar")
parabola2.SetLineColor(R.kGreen)
gaussian2.SetLineColor(R.kMagenta)
parabola2.Draw("Same")
gaussian2.Draw("Same")
gausppar2.Draw("Same")
gausppar2.SetLineColor(R.kRed)

# Draw a legend
legend = R.TLegend(0.6, 0.6, 0.88, 0.88)
#legend.AddEntry(histo1, "Data from Scipy Fit", "l")
#legend.AddEntry(histo2, "Data from Root Fit", "l")
legend.AddEntry(parabola1, "Parabola (Scipy)", "l")
legend.AddEntry(parabola2, "Parabola (Root)", "l")
legend.AddEntry(gaussian1, "Gaussian (Scipy)", "l")
legend.AddEntry(gaussian2, "Gaussian (Root)", "l")
legend.AddEntry(gausppar1, "Gausppar (Scipy)", "l")
legend.AddEntry(gausppar2, "Gausppar (Root)", "l")
legend.Draw("Same")


canvas.SaveAs("comparison.png")

file1.Close()
file2.Close()
```

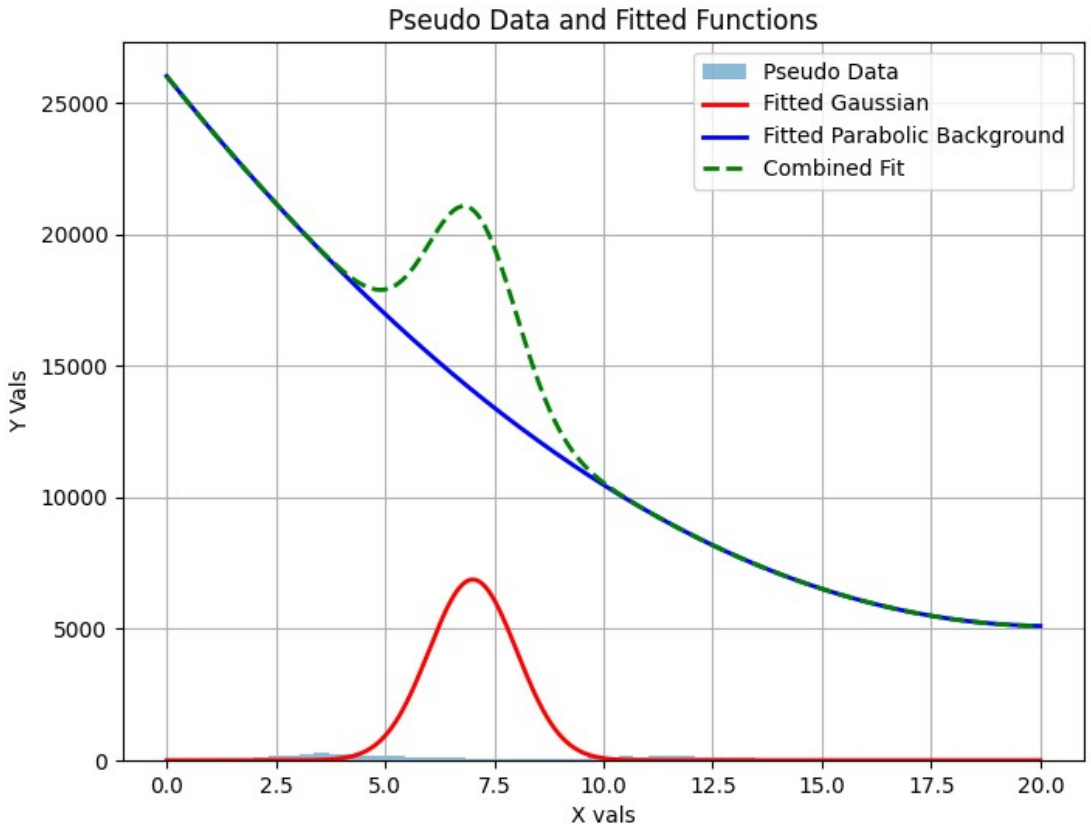
01.09 Updates

New version of rewrited ROOT code to Python, using PyROOT library and ROOT fit (Loaded in google drive)





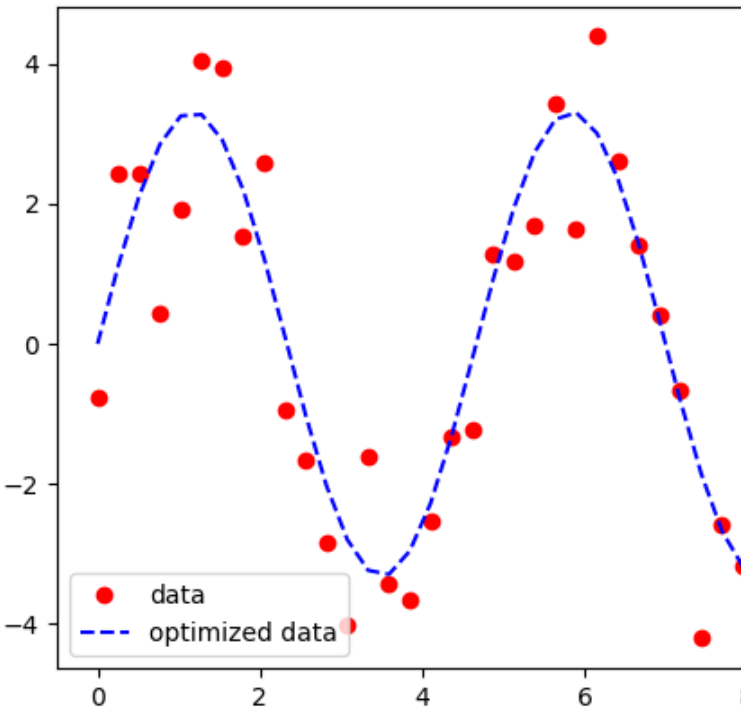
Python code using SciPy fit and Matplotlib:

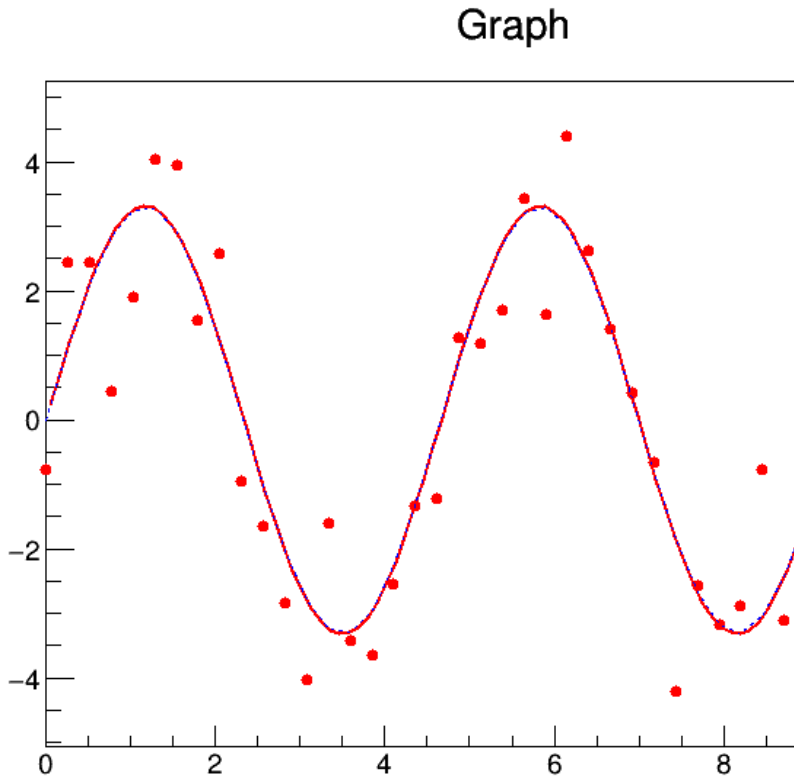


17.07 Updates

Tasks:

- ☑ ~~Read EIC report (concentrate on chapter 1,2,3)~~
- ☑ ~~Reproduce example from: SciPy | Curve Fitting - GeeksforGeeks~~
- ☑ ~~Write code with PyRoot that will use Scipy fit and root fit~~
- ☑ ~~Meet: ask Bill questions about the EIC yellow report~~





[Code with using PyRoot and Scipy]

07.07 Updates

Tasks:

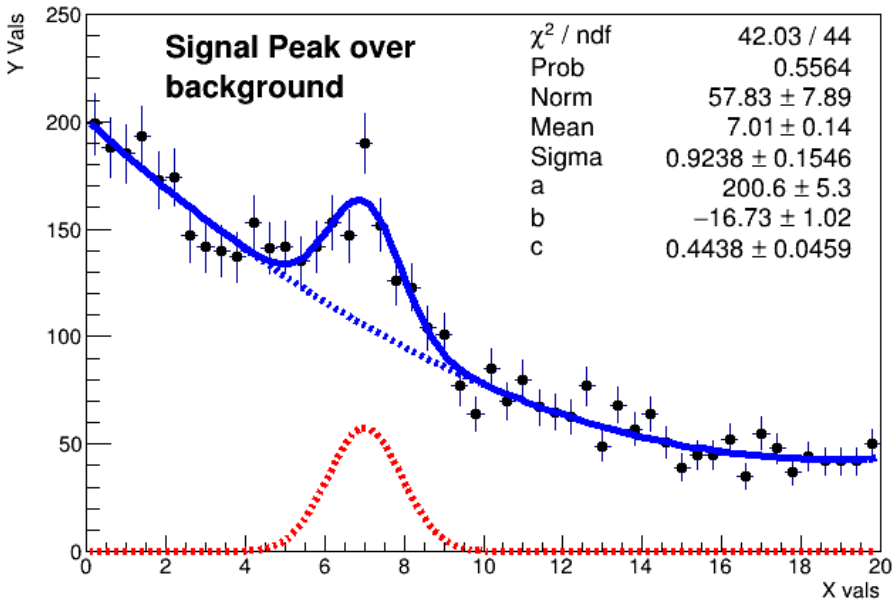
- ☑ ~~Add x_error and y_error to tree branches~~
- ☑ ~~Make the .root plot look the same as the original~~

All code uploaded to [google drive]

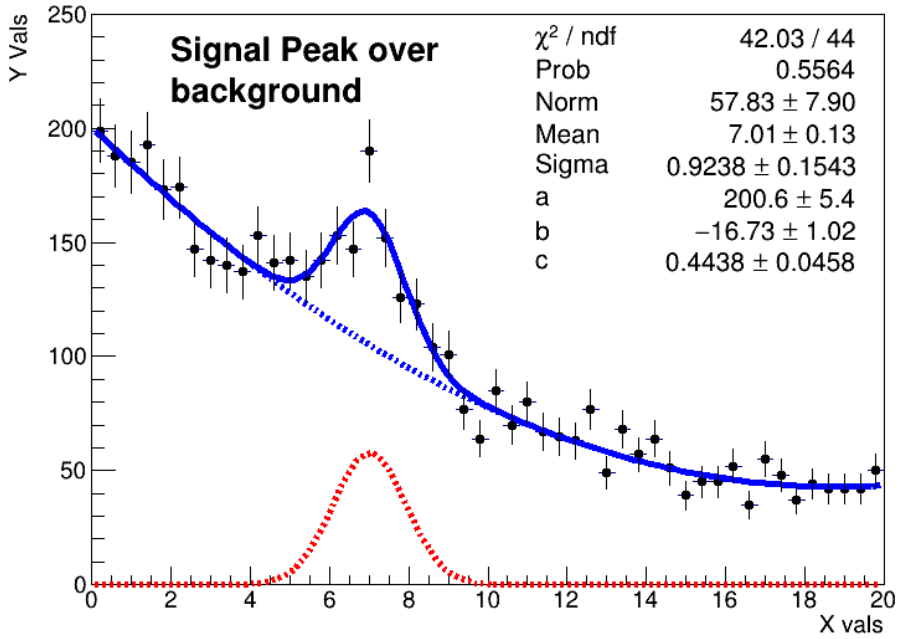
Printed Tree list from PseudoData.C:

```
******************************************************************************
*Tree    :Tree_PseudoData: tree
*Entries :       50 : Total =            4612 bytes  File  Size =          0 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :index     : index/I                                                 *
*Entries :       50 : Total  Size=        867 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :X         : X/F                                                     *
*Entries :       50 : Total  Size=        843 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :Y         : Y/F                                                     *
*Entries :       50 : Total  Size=        843 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    3 :x_error   : x_error/F                                               *
*Entries :       50 : Total  Size=        879 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    4 :y_error   : y_error/F                                               *
*Entries :       50 : Total  Size=        879 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
```

The code with the display of data from root file was rewritten, now it displays the graph in the same way as the original:



Original plot:



06.15 Updates

Tasks:

- ☑ ~~Get the data from the tree file and use it to plot the graph~~

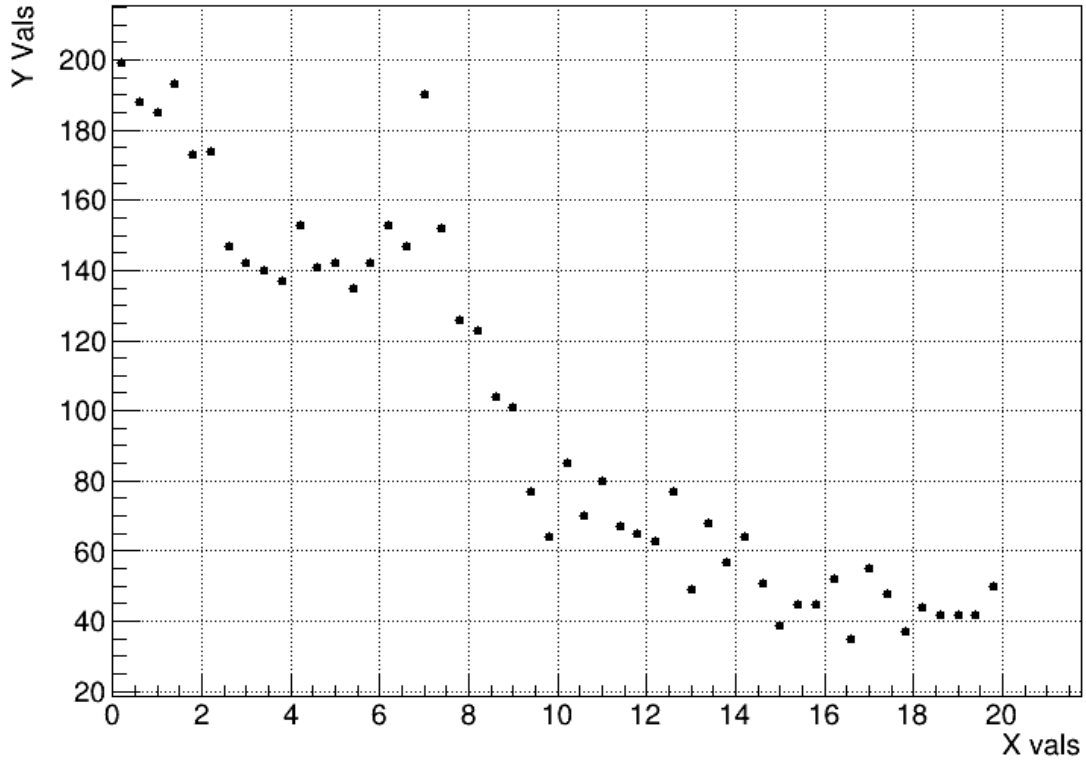
Updates in code PseudoData.root

New code for reading data and graphing from the root tree file has been written

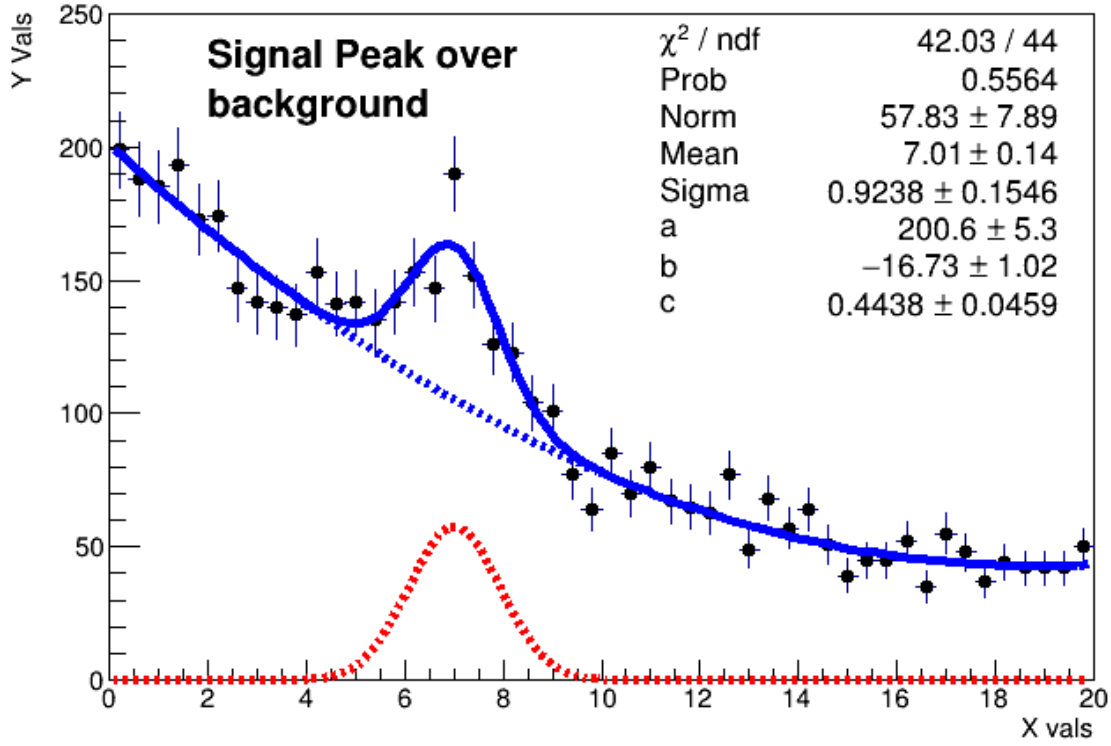
Printed tree:

```
******************************************************************************
*Tree    :Tree_PseudoData: tree
*Entries :       50 : Total =            2860 bytes  File  Size =          0 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :index     : index/I                                                 *
*Entries :       50 : Total  Size=        867 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :X         : X/F                                                     *
*Entries :       50 : Total  Size=        843 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :Y         : Y/F                                                     *
*Entries :       50 : Total  Size=        843 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
(int) 0
root [1]
```

The plot builded by .root file:



Original plot:



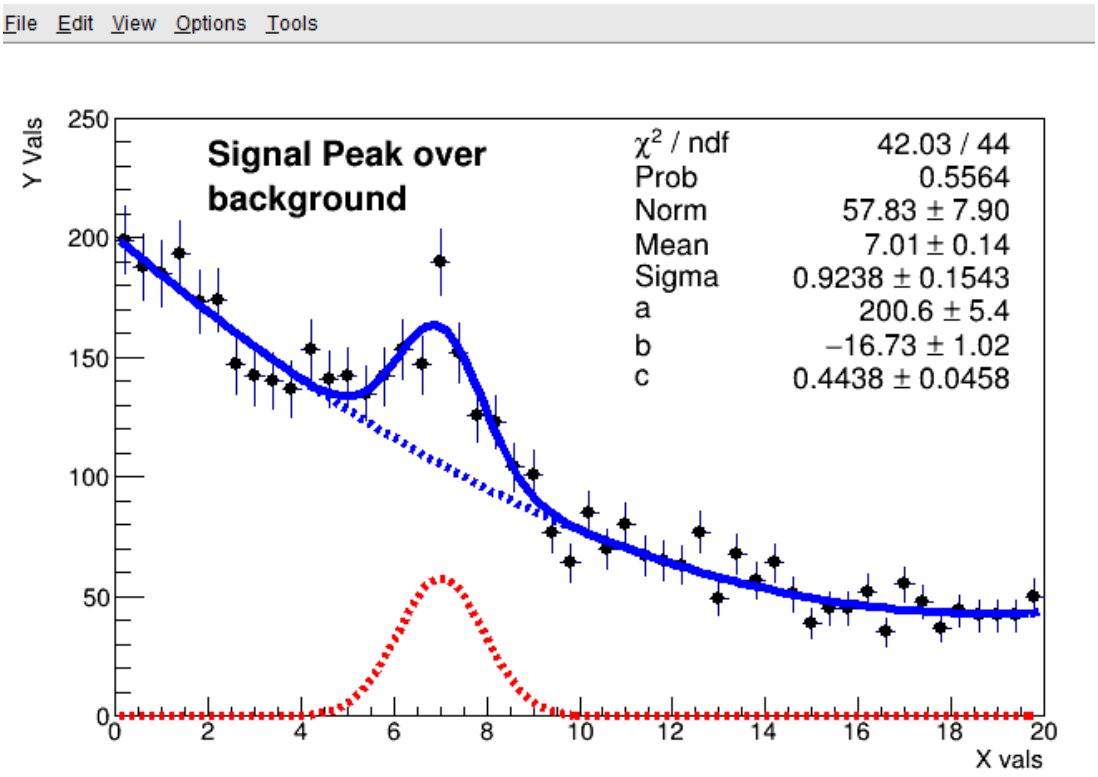
06.01 Updates

- ☑ ~~output the histogram information X, Y, Index into a tree data format~~

```
   4 |   -0.04503
   5 |   0.002095

******************************************************************************
*Tree    :PseudoData: PseudoData                                             *
*Entries :     5000 : Total =          102559 bytes  File  Size =          0 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :index     : index/I                                                 *
*Entries :     5000 : Total  Size=      20657 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :x         : x/D                                                     *
*Entries :     5000 : Total  Size=      40797 bytes  All baskets in memory   *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    2 :y         : y/D                                                     *
*Entries :     5000 : Total  Size=      40797 bytes  All baskets in memory   *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*                                                                            *
```

Start (Fitting Functions to Pseudo Data)

root.cern.ch/root/htmldoc/guides/primer/ROOTPrimer.html#fitting-functions-to-pseudo-data



Code demonstrates the use of the ROOT library to fit a model to a histogram of data. It's based on specifying a function consisting of a Gaussian component (signal) and a parabolic component (background), which will be fitted to the available data.The overall result is a process of fitting the model to the data and visualizing the results on a graph.