Measuring the mass of the Z^0 particle

Anton Granlund Gustafsson, Jona Nordin Nobuoka, Tim Sandgren

September 2022

Originally written for the course SH1015 Applied Modern Physics at KTH.
Translated from Swedish by Jona Nordin Nobuoka in December of 2023.

Contents

1 Background **3**
1.1 The Z^0 particle . . . 3
1.2 The ATLAS detector . . . 3
1.3 Calculation of invariant mass . . . 4

2 Method **4**
2.1 Data acquisition . . . 4
2.2 Distribution functions . . . 5
2.2.1 Breit-Wigner . . . 5
2.2.2 Gaussian . . . 5
2.2.3 Crystal Ball . . . 5
2.2.4 Convolutions . . . 6
2.3 Data processing . . . 6

3 Results **6**

4 Discussion **10**

5 Conclusion **11**

1 Background

In this lab, the mass of the Z^0 particle was investigated by studying collision data from the ATLAS experiment at CERN.

1.1 The Z^0 particle

The Z^0 boson is an elementary particle with electric charge 0 and spin 1. It mediates the weak interaction together with the $W^{\pm}$ bosons [1]. The particle is relatively heavy with a mass of 91.1876 ± 0.0021 GeV/c^2 [2], comparable to the mass of a silver atom. The Z^0 boson is unstable with a mean lifetime of around $3 \cdot 10^{-25}$ s. Two of its possible decays paths are to an electron-positron pair or to a muon-antimuon pair (see Figure 1). These are the cases which will be studied in this report.

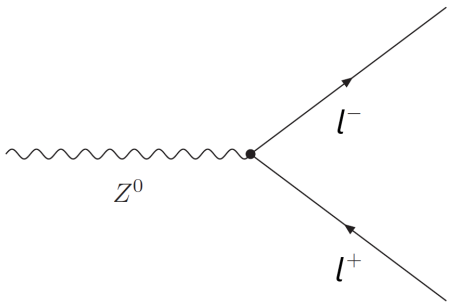


Figure 1: A Feynman diagram of a decaying Z^0 particle. l can be either an electron or a muon [1].

1.2 The ATLAS detector

The ATLAS detector collects data about the particles that are created when proton bunches collide in the particle accelerator LHC. Each collision can produce many different particles. Data from all particles that are created after a specific bunch collision is referred to as an *event*.

The detector is made of different materials that slows down the particles differently depending on their type. When a particle interacts with a detector material, the lost kinetic energy gets converted to electrical signals. This makes it possible to track the particle's trajectory through the detector.

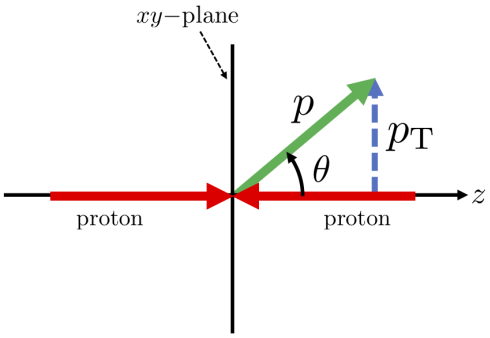
ATLAS also contains electromagnets that causes the paths of charged particles curve. This provides a way to calculate the momentum through:

$$F = qvB = \frac{mv^2}{r} \quad \Rightarrow \quad p = mv = qBr,$$

since the magnetic field B is known and the radius of curvature r can be found from the trajectory. The charge q depends on the particle type, which can be inferred from how it interacts with different detector components.

1.3 Calculation of invariant mass

During a collision, the two proton bunches are assumed to travel in opposite directions along the z-axis. For a particle produced in the collision, ATLAS measures the projection of its momentum on the xy-plane p_T, as well as the angle θ between the the total momentum p and the z-axis. The pseudorapidity η is introduced as described in Figure 2.



$$p_{\mathrm{T}} = p \sin\theta$$
$$\eta = -\ln[\tan(\frac{\theta}{2})]$$

Figure 2: Illustrates the quantities p_T and θ for a produced particle and defines the pseudorapidity η [1].

A short lived particle does not have time to enter the detector before it decays. However, the invariant mass m of an unstable particle that decays into two more stable particles can be calculated from p_T, θ and η of the daughter particles through [1]

$$m^2 \approx 2p_{T1}p_{T2}\left(\cosh(\eta_1 - \eta_2) - \cos(\theta_1 - \theta_2)\right).$$

This assumes that the daughter particles are ultrarelativistic ($pc \gg mc^2$) which is reasonable for the leptons in this lab since the Z^0 boson is so massive.

2 Method

2.1 Data acquisition

The data used in this lab comes from the ATLAS experiment at CERN in 2012. Two datasets are used, one with decay to electron-positron pairs, and one with decays to muon-antimuon pairs. The events were filtered according to the following criteria:

- The event contains only 2 leptons
- One of the leptons has to be an electron or a muon
- The other lepton has to be an antiparticle of the first one

This excludes the majority of events that do not come from Z^0-decay (se Figure 1).

The invariant mass of the filtered events were then calculated and put into histograms. The data from decay into electron pairs and decay into muon pairs were separated. The reason for this is that electrons lose more energy to bremsstrahlung since they are lighter than muons.

A histogram with 200 bins in the interval 0–120 GeV/c^2 was made to give an overview of the data.

2.2 Distribution functions

Four different distribution functions were fitted to the data:

- Gaussian distribution
- Breit-Wigner distribution (BW)
- Breit-Wigner distribution convolved with a Gaussian (BW∗Gauss)
- Breit-Wigner distribution convolved with a Crystal Ball function (BW∗CB)

2.2.1 Breit-Wigner

The Breit-Wigner distribution has the probability density function [3]:

$$f_{BW}(x; x_0, \Gamma) = \frac{1}{\pi} \frac{\Gamma/2}{(x - x_0)^2 + (\Gamma/2)^2},$$

where x_0 is the position of the peak and Γ is the width.

Since the Z^0 particle is unstable, the time-energy uncertianty principle implies that its energy is uncertain. In its rest frame, it only has mass-energy, which means that the rest mass is fundamentally uncertain. It can be shown that the mass has a Breit-Wigner distribution with restmass m_0 as median[1] [4]. The width is given by $\Gamma = \hbar/(\tau c^2)$ where τ is the mean lifetime of the particle. This assumes that relativistic effects can be neglected, which is reasonable for $\Gamma \ll m_0$.

2.2.2 Gaussian

The Gaussian distribution has the probability density function

$$f_{GAUSS}(x; \bar{x}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x - \bar{x})^2}{\sigma^2}\right)$$

where $\bar{x}$ is the mean and σ is the standard deviation.

2.2.3 Crystal Ball

The Crystal Ball distribution looks like a Gaussian with a larger tail to the left. It consists of a Gaussian that transitions into a power law. The probability density function is [5]

$$f_{CB}(x; \alpha, n, \bar{x}, \sigma) = N \cdot \begin{cases} \exp\left(-\frac{1}{2}\frac{(x-\bar{x})^2}{\sigma^2}\right) & : \quad \frac{x-\bar{x}}{\sigma} > -\alpha \\ K_1 \cdot \left(K_2 - \frac{x-\bar{x}}{\sigma}\right)^{-n} & : \quad \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases}$$

The parameter α determines where the function switches expression and n determines how quickly the power law falls off. The constants K_1 and K_2 ensures that the function and its first derivative are continuous while N ensures normalization.

[1] The Breit-Wigner distribution has no well defined mean since the corresponding integral diverges.

2.2.4 Convolutions

The probability density function of the sum of two stochastic variables is the convolution of their probability density functions. By fitting a convolution between Breit-Wigner and the distribution of some disturbance, the estimated parameter $m^* = m_0^*$ can be calculated as if that disturbance was not present.

If the disturbance comes from statistical errors then a Gaussian with mean 0, $f_{GAUSS}(m; \bar{m} = 0, \sigma)$, is a good model due to the central limit theorem. The distribution will be shifted to the left if leptons are slowed down in the detector. This effect can be incorporated with a Crystal Ball function, $f_{CB}(m; \alpha, n, \bar{m} = 0, \sigma)$.

2.3 Data processing

The mass of the Z^0 boson was estimated with the mean $\bar{m}$ for the Gaussian and with the Breit-Wigner peak position m_0 for the others.

The interval 84–98 GeV/c^2 was chosen for both the electron and muon data to minimize effects of other processes than Z^0-decay. Histograms with 50 bins were created for these intervals and distribution functions were fitted to the histograms.

An indication of the quality of a fit is given by the reduced χ^2 sum, i.e. the χ^2 divided by the number of degrees of freedom N_{DOF} for the least square approximation. A value of χ^2/N_{DOF} near 1 suggests that the fit is good.

The number N_{DOF} is the number of bins minus the number of free parameters to be fitted. The χ^2 sum is given by

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - f(x_i))^2}{{\sigma_i}^2}$$

where n is the number of bins and f is the function that is being fitted to the histogram. For bin i, x_i will be the position of the middle of its interval and y_i is the number of data points in the interval. σ_i is the uncertainty in how many data points are in the interval, which depends on y_i and the total number of data points in the histogram. The χ^2 sum is less affected by more uncertain measurements due to the σ_i^2 in the denominator.

3 Results

Histograms for number of events against invariant mass together with fitted distributions, produced from different numbers of data points, are shown in Figures 3-7. The Figures 3-5 show electron data and the Figures 6-7 show muon data.

Table 1 shows the χ^2/N_{DOF}-values for the different distributions for the datasets shown in the Figures 3-7. All fits are better for the narrower interval than for 0–120 GeV/c^2. The distribution with reduced χ^2 closest to 1 is BW$*$CB for all datasets. The other distributions also have reduced χ^2 near 1 when a smaller number of events (10 000) is studied. But their χ^2/N_{DOF} becomes much larger when more data points are included (circa 375 000 for electrons, 600 000 for muons).

Table 2 shows estimated rest masses for the Z^0 boson from different distribution functions. The estimated mass is greater for the muon data than for the electron data for all distribution. Fitting with BW$*$CB to

the electron data gives a rest mass of 90.566 ± 0.023 GeV/c^2. The corresponding fit to the muon data gives the mass 90.844 ± 0.010 GeV/c^2.

According to Particle Data Group, tha mass of the Z^0 boson should be 91.1876 ± 0.0021 GeV/c^2.

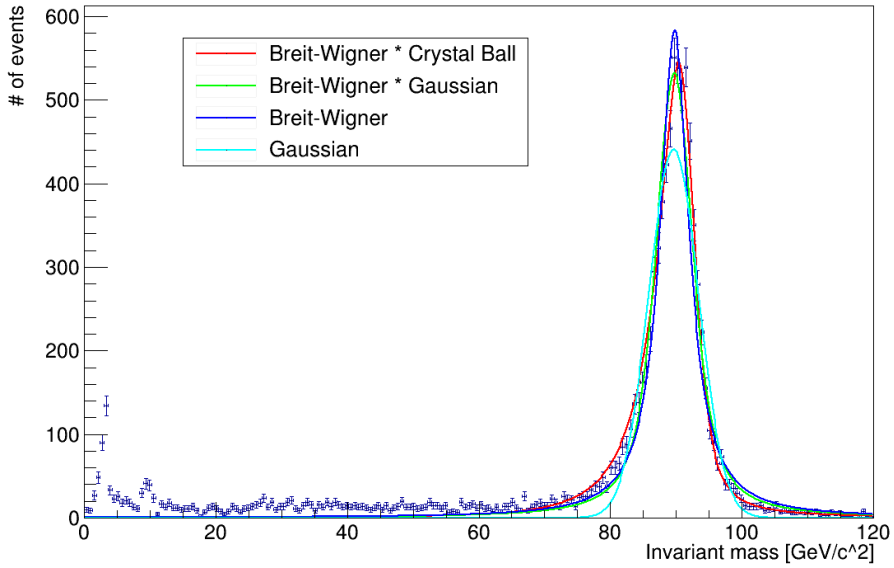


Figure 3: Curve fits for electron-positron pairs on the whole interval 0–120 GeV/c^2, 10 000 events.

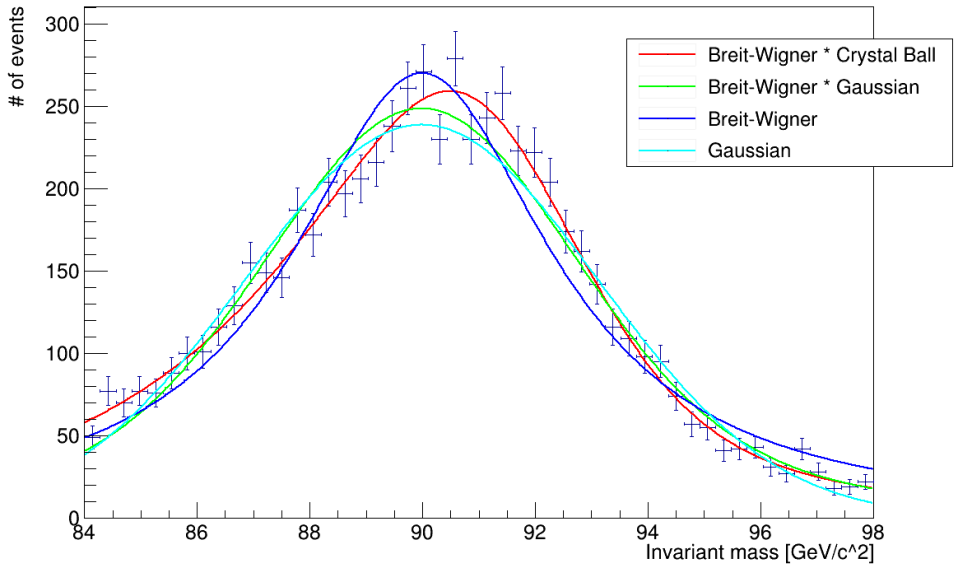


Figure 4: Curve fits for electron-positron pairs on the interval 84–98 GeV/c^2, 10 000 events.

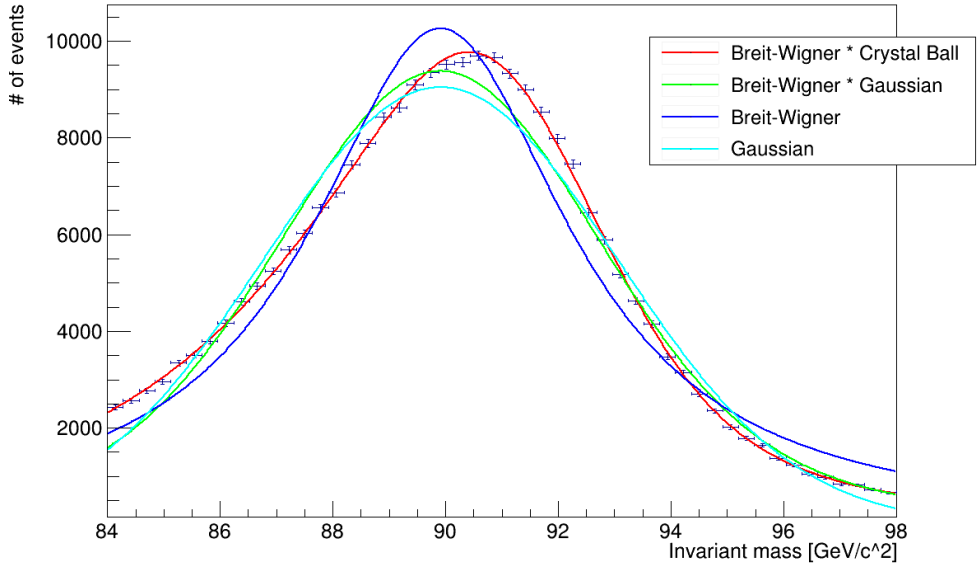


Figure 5: Curve fits for electron-positron pairs on the interval 84–98 GeV/c^2, circa 375 000 events.

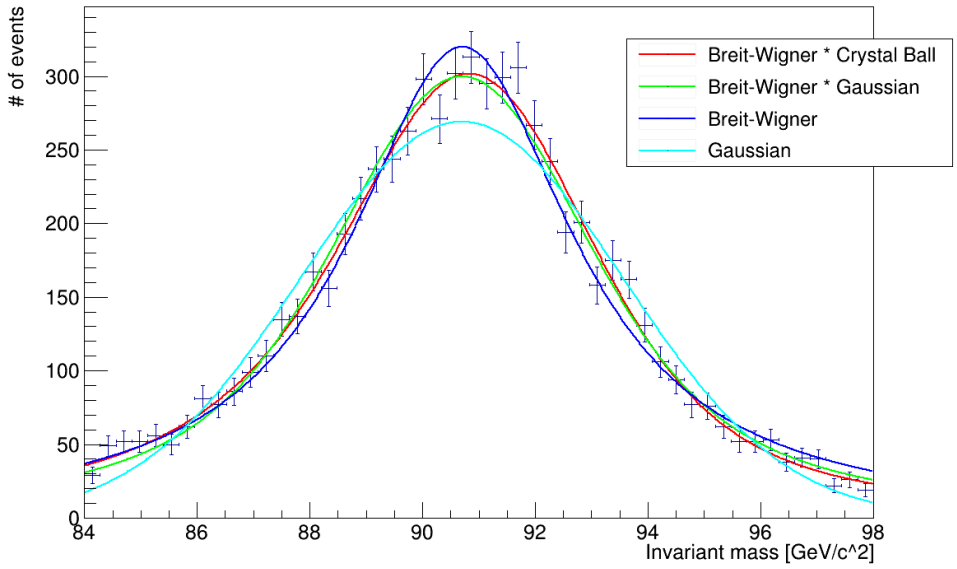


Figure 6: Curve fits for muon-antimuon pairs on the interval 84–98 GeV/c^2, 10 000 events.

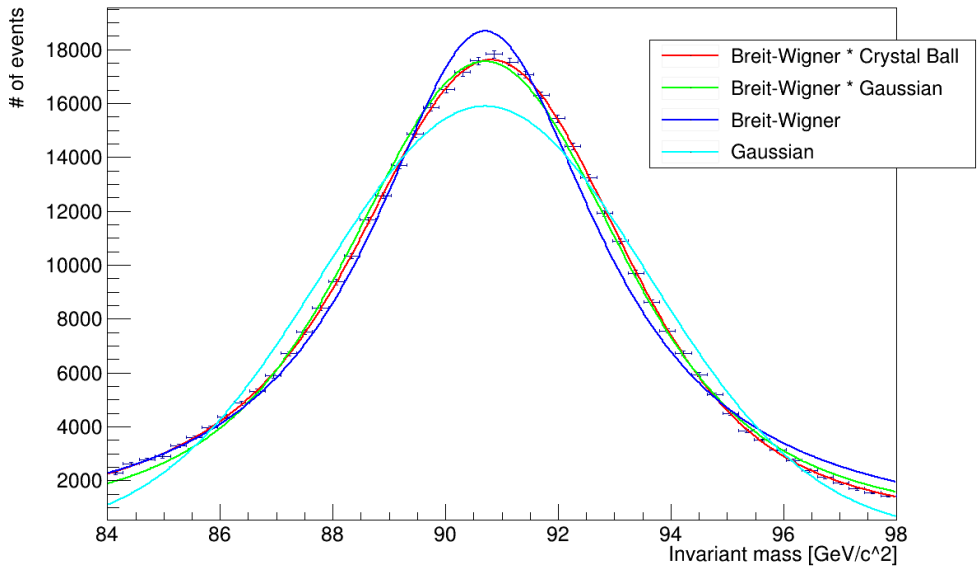


Figure 7: Curve fits for muon-antimuon pairs on the interval 84–98 GeV/c^2, circa 600 000 events.

Table 1: χ^2/N_{DOF}-values for different fits and different data sets.

Data set	BW∗CB	BW∗GAUSS	BW	GAUSS
e, 0-120 GeV/c^2, 10 000 events	8.6	9.5	10.0	14.4
e, 84-98 GeV/c^2, 10 000 events	0.9	1.7	3.4	2.1
e, 84-98 GeV/c^2, ∼375 000 events	1.2	37.6	109.3	50.5
μ, 84-98 GeV/c^2, 10 000 events	0.9	1.1	1.7	2.9
μ, 84-98 GeV/c^2, ∼600 000 events	0.9	14.7	54.3	115.3

Table 2: Least square estimates of the mass of the Z^0 boson m^* in GeV/c^2 from fits with the different distribution functions. The specified errors are one standard deviation.

Data set	BW∗CB	BW∗GAUSS	BW	GAUSS
e, 84-98 GeV/c^2, ∼375 000 events	90.566 ± 0.023	89.928 ± 0.007	89.909 ± 0.008	89.909 ± 0.007
μ, 84-98 GeV/c^2, ∼600 000 events	90.844 ± 0.010	90.726 ± 0.005	90.698 ± 0.005	90.699 ± 0.005

4 Discussion

When the whole interval 0–120 GeV/c^2 is studied, all the distributions underestimate the data between 0 and 60 GeV/c^2. Figure 3 shows a relatively constant tail to the left of the main peak as well as a smaller peak at around 4 GeV/c^2. The constant tail could possibly be explained by the electrons slowing down from bremsstrahlung before their momentum can be measured. The smaller peak could come from processes other than Z^0-decay which produce an electron-positron pair. For these reasons, the rest of the analysis was done on the a narrower interval 84–98 GeV/c^2.

The Crystal Ball function models the leptons' slowdown from bremsstrahlung as well as the statistical uncertainty when measuring their momentum. This explains why BW∗CB gives the best fit to the data, which is most visible for the larger data sets (see Figure 5 for the electrons and Figure 7 for the muons). This can be seen from the reduced χ^2-values, which are close to 1 in both cases (Table 1).

BW∗GAUSS gives a good fit when the number of data points is low, but gives a poor fit for the larger data sets. This is because the normal distribution does not model the slowdown of the particles in the detector, and therefore does not capture the skewness of the data. It does however model the statistical uncertainty from measuring the particle momenta. The fact that using just a Breit-Wigner distribution gives a much higher χ^2-quotient implies that the statistical uncertainty needs to be taken into account.

The parameters in the distribution functions have been estimated with high precision. This means that their values are expected to change very little if the fits were done on a similar dataset with the same distribution as the original.

At higher numbers of data points, the uncertainty in every bin decreases. This increases the precision of

the parameter estimations. But if the fitted curve deviates from the real distribution, then these deviations will also become more certain. The uncertainty of the parameters does not take this into account. But it can be noticed by the χ^2-quotient increasing. For this reason, the uncertainties in the estimated parameters are only accurate for fits with χ^2-quotients near 1. This is why only m^* from BW$*$CB is relevant in Table 2.

The estimated mass of the Z^0 boson agrees poorly with the known value [2] for both the electron and muon data. The deviation is greater than 10 standard deviations. This difference can only be explained by incomplete information bout which factors affect the measured data or how the measured data should be analyzed, i.e. systematic errors.

A possible systematic error could be that the Crystal Ball function is not sufficient to model the slowdown. Crystal Ball is a Gaussian where the left tail falls of slower, which is compensated by scaling down the rest of the function. In reality, the lepton breaking depends on their momentum. This is not taken into account by just scaling down the Gaussian part.

The masses of the leptons are neglected in the derivation of the invariant mass (see Background 1.3). This is still reasonable for the electron data. But the muon mass is about 0.1 GeV/c^2, which is an appreciable part of the deviation from the correct value of the Z^0 boson mass. Though it still does not account for the whole difference.

5 Conclusion

Through observing decay products form proton-proton collisions, an approximate value of the Z^0 mass could be determined. Of the distribution functions studied, BW$*$CB produced the best fits to the data. Our result was fairly close to the established value. Further investigations are needed to determine where the systematic errors come from.

References

[1] Christian Ohm. *Lab instruction*. URL: https://colab.research.google.com/github/cohm/OpenDataZmassLab/blob/SH1015-Google-Colab-support/1a-Introduction-information.ipynb. (In Swedish).

[2] R.L. Workmanet Workmanet. *2022: Summary tables*. 2022. URL: https://pdg.lbl.gov/2022/tables/contents_tables.html.

[3] CERN. URL: https://seal.web.cern.ch/seal/MathLibs/MathCore/html/group__StatFunc.html#g674162ea051bf687243264996d046f73.

[4] A.J.Barr. 2012. URL: http://www-pnp.physics.ox.ac.uk/~barra/teaching/resonances.pdf.

[5] 2020. URL: https://en.wikipedia.org/wiki/Crystal_Ball_function.

Appendix

```
import ROOT
from distributions import *
from time import sleep
from ROOT import TH1F, TCanvas, TF1, TLegend, kRed, kBlue, kGreen, kBlack, TMatrixD, gPad,
    kFALSE
import cppyy
```

```
from ROOT.Math import MinimizerOptions

cppyy.cppdef("""
double* make_double_array(int n) {
return (double*)malloc(sizeof(double) * n);
}
""")


def make_double_array(*args):
    arr = cppyy.gbl.make_double_array(len(args))
    for i, arg in enumerate(args):
        arr[i] = arg
    return arr


config = [
    dict(
        title="Breit-Wigner * Crystal Ball",
        distr=breit_wigner_conv_crystal_ball_distribution,
        electron_params=[125, 90.55, 2.7, 1, 0, 1.64, 0.61, 45],
        muon_params=[331 * 100, 90.84, 3, 1, 0, 1.57, 1.67, 1.8]
    ),
    dict(
        title="Breit-Wigner * Gaussian",
        distr=breit_wigner_conv_normal_distribution,
        electron_params=[568, 90, 3.92, 1, 0, 1.83],
        muon_params=[37815, 90.7, 3.6, 1, 0, 1.46]
    ),
    dict(
        title="Breit-Wigner",
        distr=breit_wigner_distribution,
        electron_params=[278, 90, 5.8],
        muon_params=[2432, 90.7, 4.8],
    ),
    dict(
        title="Gaussian",
        distr=normal_distribution,
        electron_params=[278, 90, 5.8],
        muon_params=[2432, 90.7, 4.8]
    )
]


kind_cfg = dict(
    muon=dict(
        title="Muon-antimuon mass; Invariant mass [GeV/c^2]; # of events"
    ),
    electron=dict(
        title="Electron-positron mass; Invariant mass [GeV/c^2]; # of events"
    )
)


def fit(kind, no_events, lowerEdge, upperEdge, n_bins=50, file_to_save=None):
    name = f"{kind}_{no_events or 'all'}_events_{lowerEdge}_to_{upperEdge}_GeV_{n_bins}
    _bins"
    file = ROOT.TFile(f"{name}.root")
```

```
    hist = file.Get(name)
    hist.SetTitle(";Invariant mass [GeV/c^2]; # of events")
    hist.SetStats(kFALSE)
    canvas = ROOT.TCanvas("Canvas", "Title", 1200, 800)
    if lowerEdge == 0 and upperEdge == 120:
        legend = TLegend(0.20, 0.65, 0.55, 0.85)
    else:
        legend = TLegend(0.65, 0.65, 0.98, 0.85)
    hist.Draw("e1")

    fns = []
    colors = [2, 3, 4, 7, 8, 9, 11, 29, 38, 46]
    for i, cfg in enumerate(config):
        fit_fn = cfg["distr"](lowerEdge, upperEdge)
        fit_fn.SetTitle(cfg["title"])
        params = list(cfg[f"{kind}_params"])
        if no_events is None and i == 1 and kind == "electron":
            params = [26731, 90.7, 1.85, 1, 0, 1.95]
        if no_events is None and i == 2 and kind == "electron":
            params[0] = 90000
        if no_events == 10000 and i == 0:  # hack to make Breit-Wigner * Crystal Ball work
 for n=10000
            params[0] = 250
        fit_fn.SetParameters(make_double_array(*params))
        fit_fn.SetLineColor(colors[i])
        fit_fn.SetNpx(1000)
        fitresult = hist.Fit(fit_fn, "S+", "e1")
        fns.append(fit_fn)
        legend.AddEntry(fit_fn, cfg["title"])

        print("\n*** Chi-square sum = {:.1f}, Number of degrees of freedom = {}, ratio =
    {:.1f}"
              .format(fitresult.Chi2(), fitresult.Ndf(), fitresult.Chi2() / fitresult.Ndf
    ()))

    legend.Draw()
    canvas.Draw()
    if file_to_save is not None:
        canvas.SaveAs(file_to_save)
        canvas.Close()
    else:
        while canvas.GetCanvasImp():
            sleep(1)


MinimizerOptions.SetDefaultMaxFunctionCalls(5000)

fit("electron", 10000,  0, 120, n_bins=200, file_to_save="
    plot_of_electron_background_and_fits.png")
fit("electron", 10000,  84, 98, n_bins=50,  file_to_save="
    plot_of_electron_mass_10000_events.png")
fit("electron", None,   84, 98, n_bins=50,  file_to_save="plot_of_electron_mass_all_events
    .png")
fit("muon",     10000,  84, 98, n_bins=50,  file_to_save="plot_of_muon_mass_10000_events.
    png")
fit("muon",     None,   84, 98, n_bins=50,  file_to_save="plot_of_muon_mass_all_events.png
    ")
```

```

import ROOT
from ROOT import TH1F, TCanvas, TF1, TLegend, kRed, kBlue, kGreen, kBlack
import math
from time import sleep
from event_analysis import *
from distributions import *
import itertools

electrons_file = ROOT.TFile.Open("./DataEgamma.root")
muons_file = ROOT.TFile.Open("./DataMuons.root")


def make_histogram(file_name, kind: {"electron", "muon"}, n_events, lowerEdge=86,
    upperEdge=94, nBins=50):
    print("making histogram", file_name, kind, n_events, lowerEdge, upperEdge, nBins)
    clear_stats()
    items = dict(electron=electrons_file, muon=muons_file)[kind].Get("mini")
    items = filter(dict(electron=is_z_boson_event_electrons, muon=is_z_boson_event_muons)[
    kind], items)
    items = limit(items, n_events)

    file = ROOT.TFile(f"{file_name}.root", "new")
    h_mass = ROOT.TH1F(file_name, "Dilepton mass; Invariant mass [GeV]; Number of events",
                       nBins, lowerEdge, upperEdge)
    fill_histogram(items, h_mass)
    h_mass.Write()
    print_stats()
    return h_mass


for (kind, n_events, lowerEdge, upperEdge, nBins) in [
    ("electron", 10000, 0, 120, 200),
    ("electron", 10000, 84, 98, 50),
    ("electron", None, 84, 98, 50),
    ("muon", 10000, 0, 120, 200),
    ("muon", 10000, 84, 98, 50),
    ("muon", None, 84, 98, 50),
]:
    make_histogram(f"{kind}_{n_events or 'all'}_events_{lowerEdge}_to_{upperEdge}_GeV_{
    nBins}_bins",
                   kind, n_events, lowerEdge, upperEdge, nBins)
```

```

import ROOT


# list of ROOT objects that should not go out of scope, in order to not be deleted
keep_list = []


def tf1(*args):
    res = ROOT.TF1(*args)
    keep_list.append(res)
    return res


def keep(val):
    keep_list.append(val)
```

```
    return val


def normal_distribution(lowerEdge,
                        upperEdge,
                        initN=2000,
                        initExpectationValue=90,
                        initStandardDeviation=2):
    normal = tf1("normal", "gaus", lowerEdge, upperEdge)
    normal.SetParameter(0, initN)
    normal.SetParameter(1, initExpectationValue)
    normal.SetParameter(2, initStandardDeviation)
    return normal


def breit_wigner_distribution(lowerEdge,
                              upperEdge,
                              initN=2000,
                              initPeak=90,
                              initDeviation=2):

    breit = tf1("breit_wigner", "breitwigner", lowerEdge, upperEdge)
    breit.SetParameter(0, initN)
    breit.SetParName(0, "N")
    breit.SetParameter(1, initPeak)
    breit.SetParName(1, "Peak")
    breit.SetParameter(2, initDeviation)
    breit.SetParName(2, "Deviation")
    return breit


def breit_wigner_conv_normal_distribution(lowerEdge,
                                          upperEdge):
    breit = tf1("breitwigner0", "breitwigner", -400, 400)
    normal = tf1("normal0", "gaus", -400, 400)
    conv = ROOT.TF1Convolution(breit, normal, -200, 200)
    keep(conv)

    fit_fn = tf1("breit_wigner_conv_normal", conv, lowerEdge, upperEdge, conv.GetNpar())
    fit_fn.FixParameter(3, 1)
    fit_fn.FixParameter(4, 0)

    fit_fn.SetParName(0, "N0")
    fit_fn.SetParName(1, "Peak")
    fit_fn.SetParName(2, "Deviation")
    fit_fn.SetParName(3, "N1")
    fit_fn.SetParName(4, "Normal-Mean")
    fit_fn.SetParName(5, "Normal-Deviation")
    return fit_fn


def breit_wigner_conv_crystal_ball_distribution(lowerEdge,
                                                upperEdge):
    breit = tf1("breit1", "breitwigner", -400, 400)
    crystalball = tf1("crystalball1", "crystalball", -400, 400)
    conv = ROOT.TF1Convolution(breit, crystalball, -200, 200)
    keep(conv)

```

```
    fit_fn = tf1("breit_wigner_conv_crystal_ball", conv, lowerEdge, upperEdge, conv.GetNpar())
    fit_fn.SetParName(0, "Constant")
    fit_fn.SetParName(1, "Peak")
    fit_fn.SetParName(2, "Deviation")
    fit_fn.SetParName(3, "CrystalBall-Constant")
    fit_fn.SetParName(4, "CrystalBall-Mean")
    fit_fn.SetParName(5, "CrystalBall-Sigma")
    fit_fn.SetParName(6, "CrystalBall-Alpha")
    fit_fn.SetParName(7, "CrystalBall-N")
    fit_fn.FixParameter(3, 1)
    fit_fn.FixParameter(4, 0)

    fit_fn.SetParLimits(0, 0.001, 100000)  # normalization breit-wigner
    # fit_fn.SetParLimits(3, 1-1e-12, 1+1e-12)
    # fit_fn.SetParLimits(4, -1e-12, 1e-12)

    return fit_fn
```

```
import itertools
import math

class stats:
    n_events = 0
    n_not_two_leptons = 0
    n_not_same_type = 0
    n_not_correct_type = 0
    n_not_opposite_charge = 0


def clear_stats():
    stats.n_events = 0
    stats.n_not_two_leptons = 0
    stats.n_not_same_type = 0
    stats.n_not_correct_type = 0
    stats.n_not_opposite_charge = 0


def print_stats():
    print("n_events=", stats.n_events)
    print("n_not_two_leptons=", stats.n_not_two_leptons)
    print("n_not_same_type=", stats.n_not_same_type)
    print("n_not_correct_type=", stats.n_not_correct_type)
    print("n_not_opposite_charge=", stats.n_not_opposite_charge)


def is_z_boson_event_electrons(event):
    stats.n_events += 1
    if event.lep_n == 2:
        if event.lep_type[0] == event.lep_type[1]:
            if event.lep_type[0] == 11 and event.lep_type[1] == 11:
                if event.lep_charge[0] == -event.lep_charge[1]:  # should be electron-positron pairs
                    return True
                else:
                    stats.n_not_opposite_charge += 1
            else:
                stats.n_not_correct_type += 1
```

```
        else:
            stats.n_not_same_type += 1
    else:
        stats.n_not_two_leptons += 1
    return False


def is_z_boson_event_muons(event):
    stats.n_events += 1
    if event.lep_n == 2:
        if event.lep_type[0] == event.lep_type[1]:
            if event.lep_type[0] == 13 and event.lep_type[1] == 13:
                if event.lep_charge[0] == -event.lep_charge[1]:  # should be electron-positron pairs
                    return True
                else:
                    stats.n_not_opposite_charge += 1
            else:
                stats.n_not_correct_type += 1
        else:
            stats.n_not_same_type += 1
    else:
        stats.n_not_two_leptons += 1
    return False


def is_z_boson_event(event):
    return is_z_boson_event_electrons(event) or is_z_boson_event_muons(event)


def getInvMass(lep1_pt, lep1_eta, lep1_phi, lep2_pt, lep2_eta, lep2_phi):
    ''' compute the invariant mass using the formula from the Introduction notebook (assume massless particles)'''
    return 1e-3 * math.sqrt(2 * lep1_pt * lep2_pt * (math.cosh(lep1_eta - lep2_eta) - math.cos(lep1_phi - lep2_phi)))


def fill_histogram(events, hist):
    n_events = 0
    for event in events:
        n_events += 1
        lep1_pt = event.lep_pt[0]
        lep1_eta = event.lep_eta[0]
        lep1_phi = event.lep_phi[0]
        lep2_pt = event.lep_pt[1]
        lep2_eta = event.lep_eta[1]
        lep2_phi = event.lep_phi[1]
        inv_mass = getInvMass(lep1_pt, lep1_eta, lep1_phi, lep2_pt, lep2_eta, lep2_phi)
        hist.Fill(inv_mass)
    print("# of events", n_events)


def limit(iterable, n):
    if n is None:
        return iterable
    else:
        return itertools.islice(iterable, 0, n)

```

```
# Table of event fields:
# | runNumber | int | run number
# | eventNumber | int | event number
# | channelNumber | int | channel number, relevant for MC
# | mcWeight | float | weight of an MC event
# | pvxp_n | int | number of reconstructed primary pp vertices
# | scaleFactor | float | overall scale factor for the preselected event
# | trigE | bool | indicates whether a standard trigger has fired in the egamma stream
# | trigM | bool | indicates whether a standard trigger has fired in the muon stream
# | passGRL | bool | indicates whether event passes the GRL may be put in isGoodEvent
# | lep_n | int | number of preselected leptons
# | lep_truthMatched | vector<bool> | indicates whether the lepton is matched to a truth lepton
# | lep_trigMatched | vector<bool> | indicates whether the lepton is the one triggering the event
# | lep_pt | vector<float> | transverse momentum of the lepton [MeV]
# | lep_eta | vector<float> | pseudorapidity of the lepton
# | lep_phi | vector<float> | azimuthal angle of the lepton
# | lep_E | vector<float> | energy of the lepton [MeV]
# | lep_z0 | vector<float> | z-coordinate of the track associated to the lepton wrt. the primary vertex [mm]
# | lep_charge | vector<float> | charge of the lepton (in unit charge)
# | lep_isTight | vector<bool> | boolean indicating whether the lepton is of tight quality
# | lep_flag | vector<int> | bitmask implementing object cuts of the top group
# | lep_type | vector<int> | number signifying the lepton type (e = 11, mu = 13, tau = 15) of the lepton
# | lep_ptcone30 | vector<float> | ptcone30 isolation for the lepton
# | lep_etcone20 | vector<float> | etcone20 isolation for the lepton
# | lep_trackd0pvunbiased | vector<float> | d0 of the track associated to the lepton at the point of closest approach (p.o.a.) [mm]
# | lep_tracksigd0pvunbiased | vector<float> | d0 signifcance of the track associated to the lepton at the p.o.a.
# | met_et | float | transverse energy of the missing momentum vector [MeV]
# | met_phi | float | azimuthal angle of the missing momentum vector
# | jet_n | int | number of selected jets
# | jet_pt | vector<float> | transverse momentum of the jet [MeV]
# | jet_eta | vector<float> | pseudorapidity of the jet
# | jet_phi | vector<float> | azimuthal angle of the jet
# | jet_E | vector<float> | energy of the jet [MeV]
# | jet_m | vector<float> | invariant mass of the jet [MeV]
# | jet_jvf | vector<float> | JetVertexFraction of the jet
# | jet_flag | vector<int> | bitmask implementing object cuts of the top group
# | jet_trueflav | vector<int> | true flavor of the jet
# | jet_truthMatched | vector<int> | information whether the jet matches a jet on truth level
# | jet_SV0 | vector<float> | SV0 weight of the jet (for tagging heavy-flavor hadrons)
# | jet_MV1 | vector<float> | MV1 weight of the jet (for tagging heavy-flavor hadrons)
# | scaleFactor_BTAG | float | scale factor for btagging
# | scaleFactor_ELE | float | scale factor for electron efficiency
# | scaleFactor_JVFSF | float | scale factor for jet vertex fraction
# | scaleFactor_MUON | float | scale factor for muon efficiency
# | scaleFactor_PILEUP | float | scale factor for pileup reweighting
# | scaleFactor_TRIGGER | float | scale factor for trigger
# | scaleFactor_ZVERTEX | float | scale factor for z-vertex reweighting
```