# New Package for RooFit Supporting Dalitz Analysis: `RooAmplitudes`

Grace Young
*CERN openlab summer student 2011*
gcyoung@mit.edu

# Contents

# 1 Project Overview

## 1.1 Background

### 1.1.1 ROOT

ROOT is a C++ based framework for handling and analysing data. It was specifically developed by CERN physicists to analyse data from high-energy physics experiments. First released in 1995, the software continues expanding to meet the needs of modern experiments, which generate hundreds of Terabytes of data monthly. Built from over 2,500 classes and 3,000,000 lines of code, ROOT includes extensive methods for 2D and 3D visualization, data fitting, threading, sharing memory, networking, and more. It is an open source software, available under the LGPL license. Over 600,000 binaries have been downloaded since 1997, and the estimated user base is currently 20,000 people [1]. ROOT interprets data from all high energy physics experiments in the world, as well as from other scientific fields and commercial industries [1].

### 1.1.2 The `RooFit` Library

ROOT includes about 100 shared libraries. One of the most popular libraries is `RooFit`, which provides tools for statistically analysing data. `RooFit` is especially useful in high-energy physics for modelling the expected distributions of events, preforming likelihood fits, producing plots, and running other statistical tests. A major objective of this project is to add functionality to `RooFit`.

### 1.1.3 Invariant Mass Distributions and Dalitz Plots in Physics

A particularly important and very common application of the `RooFit` library is to analyse invariant mass distributions. Dalitz analysis is closely related to this type of analysis, but is currently not well supported in `RooFit`. A major objective of this project is to provide an extention to `RooFit` to aid Dalitz analysis.

Briefly, invariant mass is a number that contains information about the energy and momentum of a particle; it is measured in units of energy over the speed of light squared. Particle detectors at CERN and other institutions measure the invariant masses of particles, and results are displayed in

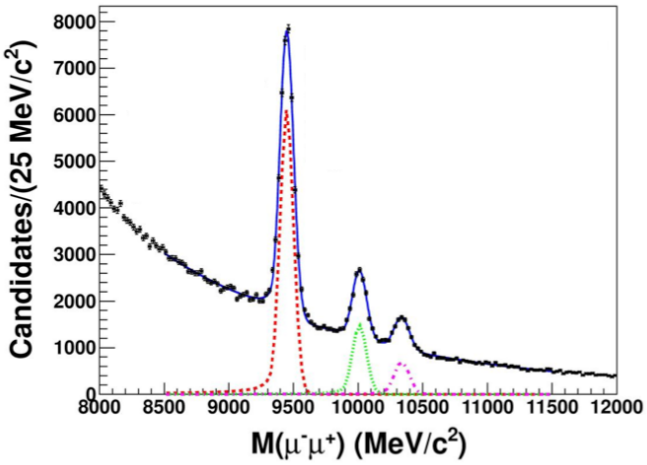


Figure 1: Example invariant mass distribution for a particle that decays into $\mu^+$ and $\mu^-$ particles [2]. The peaks in the graph indicate unstable particles, or resonances, associated with the decay.

histograms that show particles' invariant mass distributions.

An invariant mass distribution for a single resonance is usually a bell-shaped curve; it can be modelled by a probability function, most commonly either a Gaussian or Briet-Wigner (discussed in Section 3). A plot of the distribution marks invariant mass on the horizontal axis and the number of times a particle is observed on the vertical axis. Figure 1 contains an example invariant mass distribution for a system of particles.

A Dalitz plot conveys more information about particles than an invariant mass distribution plot; it is essentially a 2-dimensional invariant mass distribution plot that is also able to reveal information about the quantum states of intermediate decays of a particle. It is used specifically to convey information about the decay of a spin-0 particle into three other spin-0 particles. Both the horizontal and vertical axes of a Dalitz plot have units of invariant mass squared. Figure 2 contains example Dalitz plots, which are described in more detail in Section 4.

## 1.2 Project Objectives

The first goal of this project is to create working, readable, and well documented classes in C++ that can be implemented as extensions of the `RooFit`

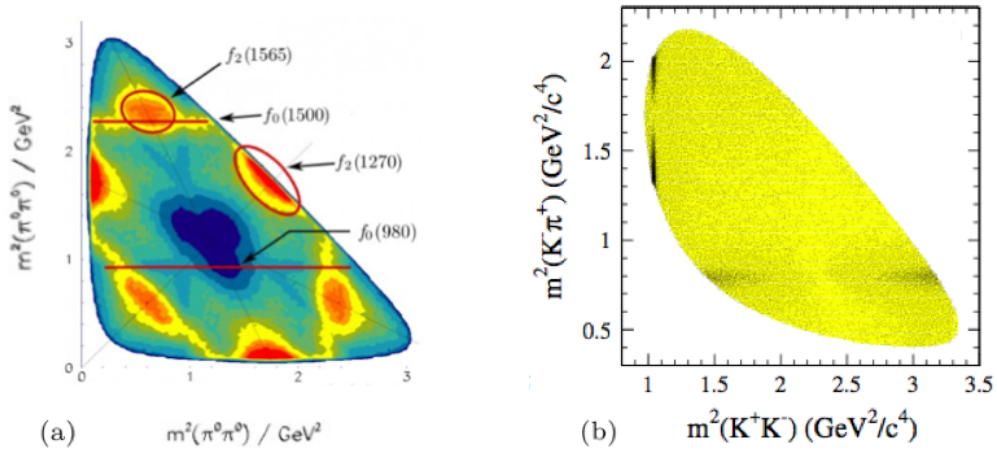


Figure 2: Example Dalitz plots for the decays $\bar{p}p \to \pi^0\pi^0\pi^0$ [3] (a) and $D_s^+ \to K^+K^-\pi^+$ [4] (b). Note both the horizontal and vertical axes have units of invariant mass squared. Warm colors on graph (a), and black on graph (b), indicate many observed particles at that mass. The rounded-triangle shape of the plot, or phase space, is defined by kinematics of the system, as discussed in Section 4.1.

library to provide support for complex-valued amplitudes, particularly Breit-Wigner amplitudes, and also to provide functions useful for Dalitz analysis. The code generated for this project will not be integrated into `RooFit` by the end of eight-week time frame for the project. The second goal, therefore, is to create a plan for how scientists will access the code and benefit from its features. Accomplishing the second goal involves establishing a community of developers who will implement and maintain the code.

# 2 Real and Complex Numbers in `RooFit` and `RooAmplitudes`

`RooAmplitudes`, the `RooFit` extension being developed for this project, contains support for complex-valued amplitudes and other features of Dalitz analysis. The backbone of `RooAmplitudes` is a set of classes that mirror existing classes in `RooFit` for handling real numbers and probability distributions. Figure 3 shows `RooFit` classes and their counterparts in `RooAmplitudes`.

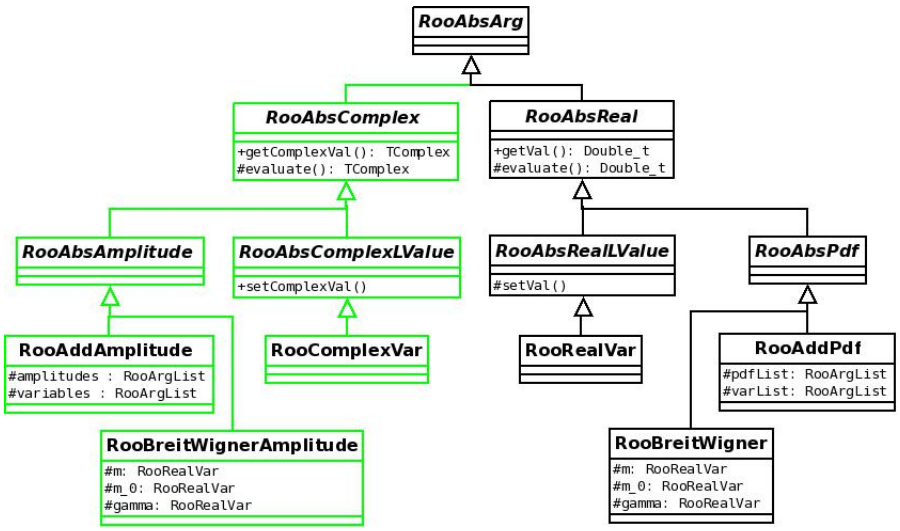


Figure 3: UML diagram showing the symmetry in the existing classes for real numbers (in black) and the new classes in `RooAmplitudes` for complex numbers (in green).

In addition, there are two classes in `RooAmplitudes` that convert complex numbers into real numbers and vice versa; both are shown in Figure 4. `RooReal2Complex` converts real variables into complex variables with zero imaginary component, and `RooAmplitudePdf` converts a complex-valued amplitude into a probability distribution.

# 3 Invariant Mass Distributions

## 3.1 Non-Relativistic Breit-Wigner Distribution

### 3.1.1 Physical Significance

The non-relativistic Breit-Wigner distribution, also known as the Cauchy distribution [5], is the simplest version of the Breit-Wigner distribution. It describes the invariant mass distribution of an unstable particle, or resonance, travelling at non-relativistic speeds as predicted by theory [6]. In other words, the Breit-Wigner distribution *models a resonance*, meaning it predicts how many times a particle is expected to be observed as a certain mass.

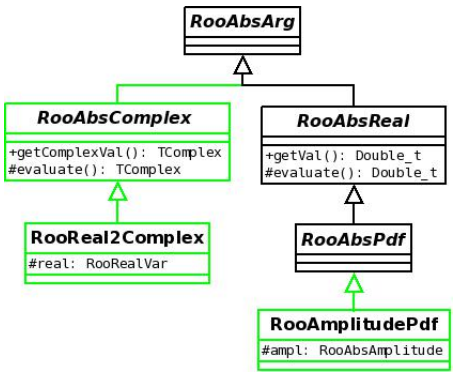


Figure 4: New classes for converting complex objects into real objects. `RooReal2Complex` converts a complex number into a real number. `RooAmplitudePdf` converts an amplitude into a probability distribution.

### 3.1.2 Mathematical Description

The amplitude $A_{BW}$ of the non-relativistic Breit-Wigner distribution for a resonance of invariant mass $m$ is given by [5]

$$A_{\mathrm{BW}}(m) = \frac{1}{(m - m_0) - i\Gamma/2}, \tag{1}$$

where $m_0$ is the central mass of the resonance and $\Gamma$ its width, where $\Gamma$ is inversely proportional to the lifetime of the particle assuming the reduced Planck's constant $\hbar = 1$.

Alternatively, we can write formula 1 using polar coordinates:

$$A_{\mathrm{BW}}(m) = |A_{\mathrm{BW}}(m)|e^{i\phi}, \tag{2}$$

where $\phi$ is the phase of the resonance.

A probability distribution is the module square of the amplitude; the Breit-Wigner distribution is therefore given by

$$|A_{\mathrm{BW}}(m)|^2 = \frac{1}{(m - m_0)^2 + (\Gamma/2)^2} \tag{3}$$

A plot of an example Breit-Wigner distribution compared to a Gaussian distribution is shown in Figure 5.

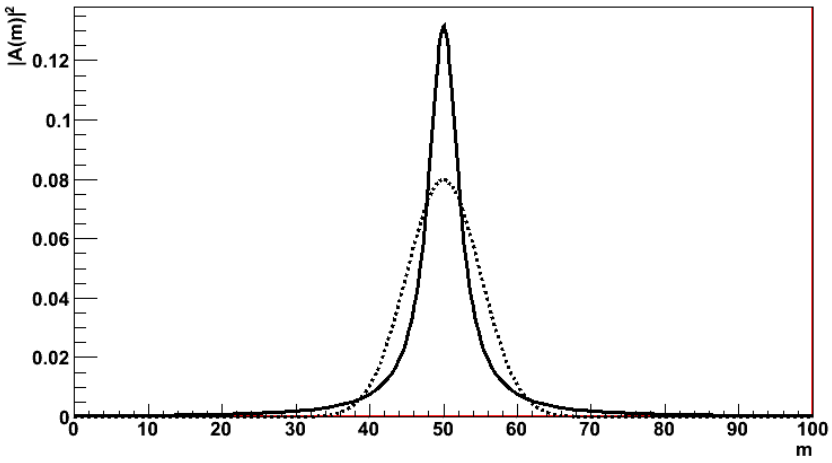


Figure 5: Example Breit-Wigner distribution (solid line) compared to a Gaussian distribution (dashed line). Both distributions have central mass $m_0 = 50MeV$ and width $\Gamma = 5MeV$.

### 3.1.3 Existing Function in ROOT: `RooBreitWigner`

The non-relativistic Breit-Wigner distribution is implemented in `RooFit` by an existing class `RooBreitWigner`, shown in the UML diagram in Figure 6.

Like all classes for probability distribution functions (PDFs), `RooBreitWigner` inherits from the abstract class `RooAbsPdf`. It has a public method `getVal()` which returns the normalized value of the PDF, meaning the value when the area under the probability curve is set to one. PDFs in `RooFit` also have a protected pure virtual method `evaluate()`, which is overloaded in the corresponding daughter class and calculates the value of the PDF that is returned by `getVal()`. It is currently not possible to return the complex-valued amplitude of distributions, which becomes a problem when combining amplitudes or preforming Dalitz Analysis, as discussed in Section 4. Appendix A.2 contains example code for defining a Breit-Wigner distribution using `RooBreitWigner`.

### 3.1.4 Implementation Using `RooBrietWignerAmplitude`

In the `RooFit` extension being developed for this project, `RooAmplitudes`, the non-relativistic Breit-Wigner distribution will be implemented by a class `RooBreitWignerAmplitude`. The class will inherit from an abstract class

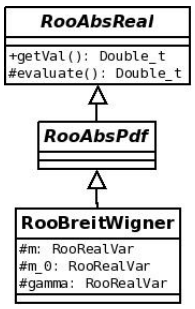


Figure 6: Existing classes in `RooFit` surrounding `RooBreitWigner`. The public method `getVal()` returns the normalized value of the Breit-Wigner function, so externally it is impossible to access the complex-valued amplitude of the function.

`RooAbsAmplitude` and have a method `evaluate()` that returns the complex-valued amplitude of the distribution (equation 1). The amplitude can be converted into a probability distribution by the class `RooAmplitudePdf`. Figure 7 shows the classes surrounding `RooBreitWignerAmplitude`, and Appendix A.3 contains an example on how to use the class.

## 3.2 Relativistic Breit-Wigner Distribution

### 3.2.1 Physical Significance

The relativistic Breit-Wigner commonly models the invariant mass distribution of an unstable particle travelling at relativistic speeds, as predicted by theory [6]. It is more commonly used in particle physics than the non-relativistic Breit-Wigner.

### 3.2.2 Mathematical Description

The amplitude $A_{BW_{rel}}$ of the relativistic Breit-Wigner distribution for a particle of invariant mass $m$ an mean mass $m_0$ is given by

$$A_{BW_{rel}} = \frac{B}{m_0^2 - m^2 - im_0\Gamma}. \tag{4}$$

Descriptions of the parameters are given in Table 1, and $B$, the Blatt-Weisskopf factor, is given in Table 2. $\Gamma$ is the mass-dependent width of the mother particle, given by [7]:

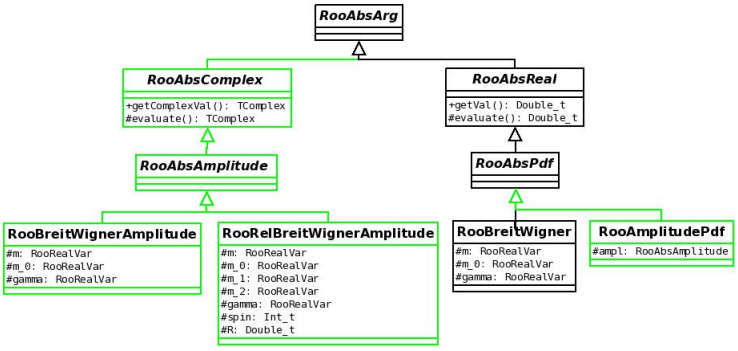


Figure 7: UML diagram for the classes related to `RooBreitWignerAmplitude` and `RooRelBreitWignerAmplitude`. The classes in black are existing in `RooFit`, and the green classes are part of the new package `RooAmplitudes`. The class `RooAmplitudePdf` converts probability amplitudes, such as `RooBreitWignerAmplitude` objects, into probability distribution functions. The existing class `RooBreitWigner` (also shown in Figure 6), is given for reference.

| parameter | description |
|---|---|
| $m$ | mass |
| $m_0$ | mass resonance |
| $l$ | spin |
| $R$ | constant (Barrier factor) |
| $m_1$ | daughter mass |
| $m_2$ | daughter mass |
| $\Gamma_0$ | resonant width |

Table 1: Parameters used in equations 4, 5, and 6 to calculate the amplitude of the relativistic Breit-Wigner distribution for a resonance decaying into two daughter particles. The default value of $R$ is 3 $GeV^{-1}$, as in reference [4].

| $l$ | $B$ |
|---|---|
| 0 | 1 |
| 1 | $\sqrt{\frac{1+R^2q_0^2}{1+R^2q^2}}$ |
| 2 | $\sqrt{\frac{(R^2q_0^2-3)^2+9R^2q_0^2}{(R^2q^2-3)^2+9R^2q^2}}$ |

Table 2: Blatt-Weisskopf factors [6] used in equations 4 and 5 to calculate the amplitude of the relativistic Breit-Wigner distribution; the parameter $R$ is described in Table 1, and $q$ and $q_0$ are given in equation 6.

$$\Gamma = \Gamma_0 \left(\frac{q}{q_0}\right)^{2l+1} \left(\frac{m_0}{m}\right) B^2. \tag{5}$$

In the case of the particle decaying into two daughter particles of masses $m_1$ and $m_2$, the value $q$ in the equations for $\Gamma$ and $B$ is given by [6]:

$$q = \frac{\sqrt{[m^2 - (m_1 + m_2)^2][m^2 - (m_1 - m_2)^2]}}{2m} \tag{6}$$

and for $q_0$, $m$ is fixed to $m_0$ in the above formula.

#### 3.2.3 Implementation Using `RooRelBreitWignerAmplitude`

Currently `RooFit` does not provide support for the relativistic Breit-Wigner function. This function will be supported in the extension `RooAmplitudes`, however, primarily by the class `RooRelBreitWignerAmplitude`, shown in the UML diagram in Figure 7. Like all probability amplitudes in the new package, it inherits from `RooAbsAmplitude` and contains an `evaluate()` method. The amplitude can be converted into a probability distribution function by the class `RooAmplitudePdf`. Appendix B.3 contains descriptions of the main constructors for `RooRelBreitWignerAmplitude`.

# 4 Dalitz Analysis

## 4.1 Basic Description

Dalitz analysis reveals information about the decay of a spin-0 particle into three other spin-0 particles (*i.e.,* three-body decay). The method is based

upon the Dalitz plot, described in Section 1.1.3, which conveys information about the invariant masses and quantum states of resonances associated with the decay. Example plots are given in Figure 2 of Section 1.1.3.

The rules of kinematics constrain the plot area, giving it the shape of a rounded-triangle known as the *phase space*. Figure 8 shows the boundaries of a phase space for a decay of a mother particle of mass $M$ into daughter particles of masses $m_1, m_2$, and $m_3$. The invariant mass obtained from the first and and second daughter particles is $m_{12}$, and the invariant mass obtained from the second and third particles is $m_{23}$. The invariant mass $m_{13}$ can be calculated from the other masses involved in the decay:

$$m_{13}^2 = M^2 + m_1^2 + m_2^2 + m_3^2 - m_{12}^2 - m_{23}^2. \tag{7}$$

The variables $m_{12}^2$ and $m_{23}^2$ are on the axes of the Dalitz plot, and the boundary formulas referenced in Figure 8 are:

$$(m_{23}^2)_{max} = (E_2 + E_3)^2 - \left(\sqrt{E_2^2 - m_2^2} - \sqrt{E_3^2 - m_3^2}\right)^2 \tag{8}$$

$$(m_{23}^2)_{min} = (E_2 + E_3)^2 - \left(\sqrt{E_2^2 - m_2^2} + \sqrt{E_3^2 - m_3^2}\right)^2, \tag{9}$$

where $E_2 = (m_{12}^2 - m_1^2 + m_2^2)/(2m_{12})$ and $E_3 = (M^2 - m_{12}^2 - m_3^2)/(2m_{12})$ are the energies of masses $m_2$ and $m_3$ in the $m_{12}$ rest frame [8].

The phase space is defined by the class `RooPhaseSpace`, which takes as input $m_{12}$, $m_{23}$, $M$, $m_1$, $m_2$, $m_3$, and $R$, as described in Appendix B.2.

## 4.2 Two-Dimensional Amplitudes

### 4.2.1 Mathematical Description

As mentioned previously, a Dalitz plot is essentially a 2-dimensional invariant mass distribution plot. The amplitude of the distribution in a Dalitz plot is, therefore, 2-dimensional. It is closely related, however, to the 1-dimensional amplitudes commonly seen in invariant mass distributions, such as Gausian or Breit-Wigner amplitudes.

Each resonance $r$ in the decay has a 2-dimensional amplitude $A_r$. The total 2-dimensional amplitude of the Dalitz plot is the sum of the amplitudes for single resonances multiplied by complex coefficients $c_r$:

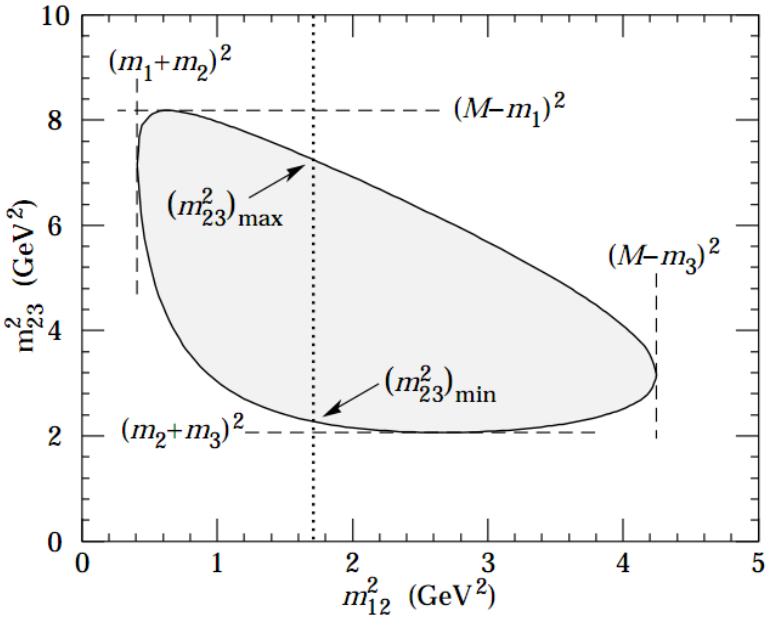


Figure 8: Example Dalitz plot for the decay of a mother particle of mass $M$ into daughter particles of masses $m_1, m_2$, and $m_3$ [8]. The rounded triangle-shaped plotting region, or phase space, is defined by the kinematics of the system. It has boundaries that depend on the masses involved in the decay, as indicated in the figure. Equations 8 and 9 describe $(m_{23}^2)_{max}$ and $(m_{23}^2)_{min}$.

$$A_{2D\,total} = \sum_r c_r A_r(m_{12}^2, m_{23}^2), \tag{10}$$

where $A_r$ depends on the squared invariant masses $m_{12}^2$ and $m_{23}^2$. $A_r$ is given in equation 11 as the product of a mass term $A$, a Blatt-Weisskopf factor $B_r$, and an angular term $T$ [4]. The resonance $r$ can decay to the first and second daughter particles, or the second and third daughter particles.

$$A_r(m_{12}^2, m_{23}^2) = A(m_r^2) \times B_r(m_r^2) \times T(m_r^2, m_{12}^2, m_{23}^2), \tag{11}$$

where $m_r$ is the invariant mass, either $m_{12}$ or $m_{23}$ depending on the resonance. The mass term $A$ is the amplitude of a 1-dimensional probability distribution, such as a Briet-Wigner amplitude (equations 2 and 4). The additional Blatt-

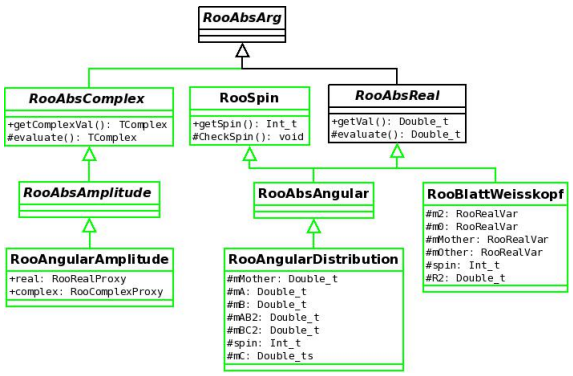


Figure 9: The classes in `RooAmplitudes` used to calculate 2-dimensional amplitudes: `RooAngularDistribution`, `RooAngularAmplitude`, and `RooBlattWeisskopf`. `RooSpin` is a class to hold the spin of the resonance.

Weisskopf factor $B_r$ is given in Table 2 of Section 3.2, where $R$ is the constant associated with the mother particle. In the formula for $q$ (equation 6) $m = m_r$, $m_1 = M$, and $m_2$ is the mass of the other particle not involved in the resonance (*i.e.,* $m_3$ if using $m_r = m_{12}$, and $m_1$ if using $m_r = m_{23}$). The spin used to calculate both $A$ and $B_r$ is that of the resonance $r$.

The angular term $T$ depends on the spin of the resonance and all masses involved in the decay; it is given by the following equation for $m_r = m_{12}$.

$$T = \begin{cases} 1, & \text{spin } 0 \\ m_{23}^2 - m_{13}^2 - \frac{(M^2 - m_3^2)(m_2^2 - m_1^2)}{m_{12}^2}, & \text{spin } 1 \\ a_1^2 - \frac{1}{3} a_2 a_3, & \text{spin } 2 \end{cases} \tag{12}$$

$$\begin{aligned} a_1 &= m_{23}^2 - m_{13}^2 + \frac{(M^2 - m_3^2)(m_1^2 - m_2^2)}{m_{12}^2} \\ a_2 &= m_{12}^2 - 2M^2 - 2m_3^2 + \frac{(M^2 - m_3^2)^2}{m_{12}^2} \\ a_3 &= m_{12}^2 - 2m_1^2 - 2m_2^2 + \frac{(m_1^2 - m_2^2)^2}{m_{12}^2} \end{aligned}$$

### 4.2.2 Implementation Using `RooAngular` Classes

In `RooAmplitudes`, three classes are involved in the computation of a 2-dimensional amplitude; they are shown in the diagram in Figure 9.

A class `RooAngularDistribution` computes the angular term $T$ using equation 12. It does not need to accept the parameter $m_{13}$ because it can be calculated using equation 7. Another convenient property of the class `RooAngularDistribution` is that it has a default constructor for the case of spin-0; in this case it doesn't need to accept the values of any of the masses. Both constructors are described in Appendix B.4. Another class `RooBlattWeisskopf` computes the Blatt-Weisskopf factor $B_r$.

The class `RooAngularAmplitude` returns the final 2-dimensional amplitude given by equation 11; it's constructor is described in Appendix B.5.

## 4.3 Existing Software for Dalitz Analysis

There are a few existing software packages for Daltz Analysis. Most are targeted towards a specific physical process; they are listed in reference [9]. Another software package is being developed by Dr. Jonas Rademacker at LHCb called MintDalitz [1], which provides extensions for `RooFit` to aid Dalitz analysis. This package provides many of the features of `RooAmplitudes`, but is currently unfinished.

# 5 Complete Class Structure of `RooAmplitudes`

Figure 10 contains a complete diagram of the classes in `RooAmplitudes`, which provides tools for basic Dalitz analysis within `RooFit`. An example script for Dalitz analysis with two resonances using `RooAmplitudes` is given in Appendix A.1.

# 6 Community of Users

`RooAmplitudes` is a complete package providing tools for basic Dalitz analysis. It can be used as an extension of `RooFit`, but it is not yet available for the public to download. There will be a meeting in September to discuss the code

[1] `http://www.phy.bris.ac.uk/people/rademacker_j/documentation_html/classes.html`

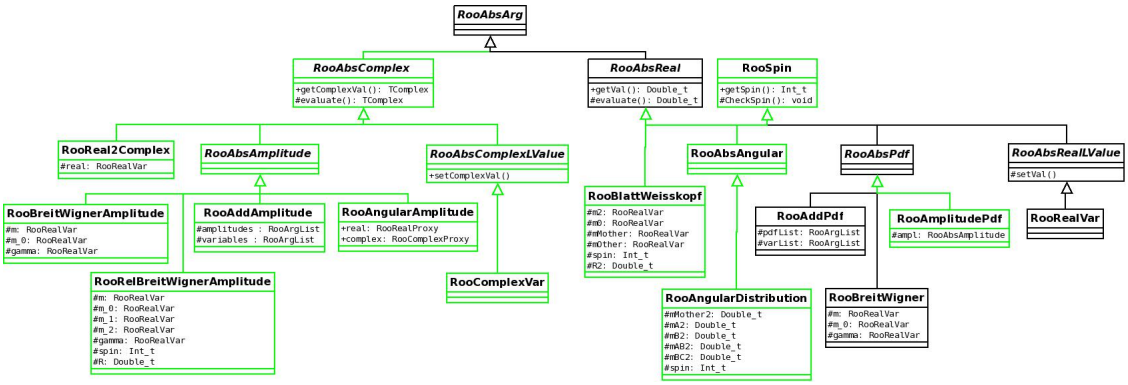


Figure 10: UML diagram for classes in `RooAmplitudes`. Black classes are existing in ROOT, and green classes are new classes included in the `RooAmplitudes` package.

with potential users, and to compare results generated using `RooAmplitudes` with results from other analysis programs. If the package gets a sizeable number of users, it will be integrated permanently into `RooFit`.

# References


[1] Jan Fiete Grosse-Oetringhaus, *Introduction to ROOT*, Summer Student Lecture, July 11th, 2011. `http://indico.cern.ch/getFile.py/access?resId=1&materialId=slides&confId=134329`

[2] *Beautiful atoms*. 6 September 2010. `http://lhcb-public.web.cern.ch/lhcb-public/`

[3] T. Gershon, *Introduction to Dalitz Plot Analysis*. University of Warwick (April 2011).

[4] BaBar Collaboration, *Dalitz plot analysis of* $D_s^+ \to K^+K^-\pi^+$. Physical Review 83 (2011).

[5] Frederick James, *Statistical Methods in Experimental Physics*. World Scientific Publishing Co. 2nd Edition (2006).

[6] D. Asner, *Charm Dalitz Plot Analysis Formalism and Results* (2003)

[7] D. Asner, *Dalitz Plot Analysis Formalism*. Pacific Northwest National Laboratory (January 2006).

[8] K. Nakamura *et al.,*. *Kinematics*. Particle Data Group. (July 30, 2010). `http://pdg.lbl.gov/2011/reviews/rpp2011-rev-kinematics.pdf`.

[9] Fernando Martinez-Vidal, *Dalitz Analysis Fitting Tools Survey*. (November 29, 2005) `http://ific.uv.es/~martinee/DalitzTools.html`

# A Worked Examples

## A.1 Creating a Dalitz Plot

This example uses many of the classes in `RooAmplitudes` to draw a Dalitz plot. The output is shown in Figure 11. The data is taken directly from the Particle Data Group via the TParticlePDG database. The same analysis was preformed by the BaBar collaboration at CERN [4].

```
#include <iostream>

#include "TApplication.h"
#include "TROOT.h"
#include "TCanvas.h"
#include "RooRealVar.h"
#include "RooBreitWigner.h"
#include "RooAddPdf.h"
#include "RooPlot.h"
#include "TLegend.h"
#include "RooGlobalFunc.h"
#include "TH1.h"
#include "RooDataSet.h"
#include "TStyle.h"
#include "TPaletteAxis.h"
#include "RooNumIntConfig.h"

// Retrive particle informations
#include "TDatabasePDG.h"
#include "TParticlePDG.h"

// new classes
#include "RooBreitWignerAmplitude.h"
#include "RooRelBreitWignerAmplitude.h"
#include "RooAddAmplitude.h"
#include "RooAmplitudePdf.h"
#include "RooComplexVar.h"
#include "RooAngularDistribution.h"
#include "RooBlattWeisskopf.h"
#include "RooAngularAmplitude.h"
#include "RooPhaseSpace.h"


int main(int argc, char** argv)
{
  gROOT->SetStyle("Plain");
  gStyle->SetPalette(1);

  // Declare an application to draw inside a compiled macro
  TApplication app("app",&argc,argv);

  // Retrieve particle masses from the PDG
  TParticlePDG *Ds = TDatabasePDG::Instance()->GetParticle("D_s+");
```

```
RooRealVar mDs("mDs","",Ds->Mass());
TParticlePDG *pi = TDatabasePDG::Instance()->GetParticle("pi+");
RooRealVar mpi("mpi","",pi->Mass());
TParticlePDG *kaon = TDatabasePDG::Instance()->GetParticle("K+");
RooRealVar mkaon("mkaon","",kaon->Mass());

// Define Dalitz variables
RooRealVar m2KK("m2KK","",0.8,3.5);
RooRealVar m2KPi("m2KPi","",0.3,2.3);

// Define phase space
RooPhaseSpace phsp("phsp","Phase Space D_{s}^{+} -> K^{+} K^{-} #pi^{+}",m2KK,m2KPi,mDs,mkaon,mkaon,mpi,5);

// Calculate angular term for angular amplitude
RooAngularDistribution angularSpin1AB("angularSpin1AB","",phsp.mass2AB(),phsp,1);
RooAngularDistribution angularSpin1BC("angularSpin1BC","",phsp.mass2BC(),phsp,1);

// Retrieve from PDG mass and width of each intermediate state, phi and K*
TParticlePDG *phi = TDatabasePDG::Instance()->GetParticle("phi");
RooRealVar m0_phi("m0_phi","",phi->Mass());
RooRealVar gamma_phi("gamma_phi","",phi->Width());
TParticlePDG *kstar = TDatabasePDG::Instance()->GetParticle("K*0");
RooRealVar m0_kstar("m0_kstar","",kstar->Mass());
RooRealVar gamma_kstar("gamma_kstar","",kstar->Width());

// Define relitivistic Breit-W amplitudes for phi and K*
RooRelBreitWignerAmplitude rbw_ampl_phi("rbw_ampl_phi","",phsp.mass2AB(),m0_phi,gamma_phi,phsp,1,3);
RooRelBreitWignerAmplitude rbw_ampl_kstar("rbw_ampl_kstar","",phsp.mass2BC(),m0_kstar,gamma_kstar,phsp,1,3);

// Calculate the 2D amplitudes for phi and K*
RooAngularAmplitude phiAmplitude("phiAmplitude","",angularSpin1AB,rbw_ampl_phi);
RooAngularAmplitude kstarAmplitude("kstarAmplitude","",angularSpin1BC,rbw_ampl_kstar);

// Define coefficients for phi and K*
RooRealVar coeffPhi("coeffPhi","",0.2);
RooRealVar coeffKstar("coeffKstar","",0.2);

// Calculate the total 2D amplitude
RooAddAmplitude totalAmpl("totalAmpl","",RooArgList(phiAmplitude,kstarAmplitude),RooArgList(coeffPhi,coeffKstar));
RooAmplitudePdf model("model","Model",totalAmpl,phsp);

// Configure graph
RooNumIntConfig customConfig(*RooAbsReal::defaultIntegratorConfig()) ;
customConfig.method1D().setLabel("RooAdaptiveGaussKronrodIntegrator1D");
model.setIntegratorConfig(customConfig);
RooDataSet *data = model.generate(RooArgSet(phsp.mass2AB(),phsp.mass2BC()),10000);
data->Print();
```

```
  TH1* plotData = data->createHistogram("plotData",phsp.mass2AB(),RooFit::YVar(phsp.mass2BC()));
  plotData->GetZaxis()->SetTitleOffset(1.8);
  TCanvas c3("c3",model.GetTitle(),140,140,1500,600);
  c3.Divide(3,1);

  // Draw Dalitz plot
  c3.cd(1);
  gPad->SetRightMargin(0.2);
  phsp.Draw();
  plotData->Draw("same colz");

  // Draw data and projection of Dalitz plot
  c3.cd(2);
  RooPlot *frame_m2KK = m2KK.frame();
  frame_m2KK->SetTitle("Projection of m_{KK}^{2}");
  data->plotOn(frame_m2KK);
  model.plotOn(frame_m2KK);
  frame_m2KK->Draw();

  // Draw data and projection of Dalitz plot
  c3.cd(3);
  RooPlot *frame_m2KPi = m2KPi.frame();
  frame_m2KPi->SetTitle("Projection of m_{K#pi}^{2}");
  data->plotOn(frame_m2KPi);
  model.plotOn(frame_m2KPi);
  frame_m2KPi->Draw();

  app.Run();

  return 0;

}
```

Listing 1: main.cxx

## A.2 Defining `RooBreitWigner`

```
// define parameters
RooRealVar M("M","M",800,1100,"MeV/c^{2}"); // variable mass
RooRealVar m0("m0","",900,8,1200); // mean mass
RooRealVar gamma("gamma","",10,0,120); // width

// define Breit Wigner distribution
RooBreitWigner bw("bw","",M,m0,gamma);
```

## A.3 Using `RooAmplitudePdf` to Create a Probability Distribution from `RooBreitWignerAmplitude`

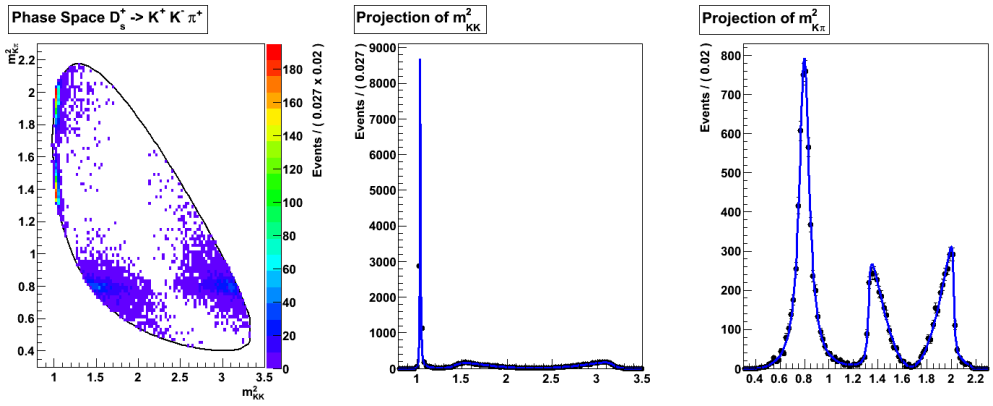


Figure 11: Output from Worked Example

```
// define parameters
RooRealVar M("M","M",800,1100,"MeV/c^{2}"); // variable mass
RooRealVar m0("m0","",900,8,1200); // mean mass
RooRealVar gamma("gamma","",10,0,120); // width

// define Breit Wigner amplitude
RooBreitWignerAmplitude bw_amp("bw","",M,m0,gamma);

// convert the amplitude into a probability distribution function
RooAmplitudePdf bw("bw","",bw_amp);
```

# B Descriptions of Constructors

## B.1 `RooRelBreitWignerAmplitude` without Dalitz Analysis

The main constructor for `RooRelBreitWignerAmplitude` accepts the parameters of the Breit-Wigner function listed in Table 1, and an optional argument `Bool_t _square`, which indicates whether or not all the input masses are squared.

```
// main ctor
RooRelBreitWignerAmplitude(const char *name, const char *title, RooAbsReal↩
    & _m, RooAbsReal& _m0, RooAbsReal& _gamma, RooAbsReal& _mA, RooAbsReal↩
    & _mB, Int_t _spin, Bool_t _square = kFALSE, Double_t _R = 3) ;
```

Another constructor for `RooRelBreitWignerAmplitude` accepts all the arguments of the main constructor (above), but allows the user to input only one daughter mass for the case when both daughter particles have the same mass.

```
// ctor in case of particle going to daughter particles of same mass
RooRelBreitWignerAmplitude(const char *name, const char *title, RooAbsReal&
    _m, RooAbsReal& _m0, RooAbsReal& _gamma, RooAbsReal& _mDaughters,
    Int_t _spin, Bool_t _square = kFALSE, Double_t _R = 3) ;
```

## B.2 `RooPhaseSpace`

```
RooPhaseSpace(const char *name, const char *title,
              RooAbsRealLValue& _mAB2, RooAbsRealLValue& _mBC2,
              RooAbsReal& _mMother, RooAbsReal& _mA,
              RooAbsReal& _mB, RooAbsReal& _mC,
              Double_t _RMother = 3);
```

## B.3 `RooRelBreitWignerAmplitude` with Dalitz Analysis

`RooRelBreitWignerAmplitude` has additional constructiors for when the function is used in Dalitz. The following two constructors match those described above for the case without Dalitz analysis, but accept additional arguments necessary for the calculation of the Blatt-Weisskopf factor.

```
// Does also the calculation of the RooBlattWeisskopf in case of Dalitz
    analysis
RooRelBreitWignerAmplitude(const char *name, const char *title, RooAbsReal&
    _m, RooAbsReal& _m0, RooAbsReal& _gamma, RooAbsReal& _mA, RooAbsReal&
    _mB, RooAbsReal& _mMother, RooAbsReal& _mOther, Int_t _spin, Bool_t
    _square = kFALSE, Double_t _R = 3, Double_t _RMother = 3) ;

// Does also the calculation of the RooBlattWeisskopf in case of Dalitz
    analysis
RooRelBreitWignerAmplitude(const char *name, const char *title, RooAbsReal&
    _m, RooAbsReal& _m0, RooAbsReal& _gamma, RooAbsReal& _mDaughters,
    RooAbsReal& _mMother, RooAbsReal& _mOther, Int_t _spin, Bool_t _square
     = kFALSE, Double_t _R = 3, Double_t _RMother = 3) ;
```

An additional constructor accepts an object of type `RooPhaseSpace` and does not require the user to input the particle masses.

```
// retrieve informations from the phase-space
  RooRelBreitWignerAmplitude(const char *name, const char *title, ↩
      RooAbsReal& _m, RooAbsReal& _m0, RooAbsReal& _gamma, RooPhaseSpace& ↩
      phsp, Int_t _spin = 0, Double_t _R = 3);
```

## B.4 RooAngularDistribution

```
RooAngularDistribution(const char *name, const char *title,
                RooAbsReal& _mAB2, RooAbsReal& _mBC2,
                RooAbsReal& _mMother, RooAbsReal& _mA,
                RooAbsReal& _mB, RooAbsReal& _mC,
                Int_t _spin, Int_t order = 0); // order 0 means taking AB ↩
                    as reference
RooAngularDistribution(const char *name, const char *title,
                RooAbsReal& _m2, RooAbsReal& _mOtherRes2); // spin-0

RooAngularDistribution(const char *name, const char *title,
                RooAbsReal& _m2, RooPhaseSpace& phsp, Int_t _spin = 0);
```

## B.5 RooAngularAmplitude

```
RooAngularAmplitude(const char *name, const char *title,
                      RooAbsAngular& _angularTerm,  RooBlattWeisskopf& _B,
                      RooAbsAmplitude& _massTerm) ;
RooAngularAmplitude(const char *name, const char *title,
                      RooAbsAngular& _angularTerm,
                      RooAbsAmplitude& _massTerm) ;
```