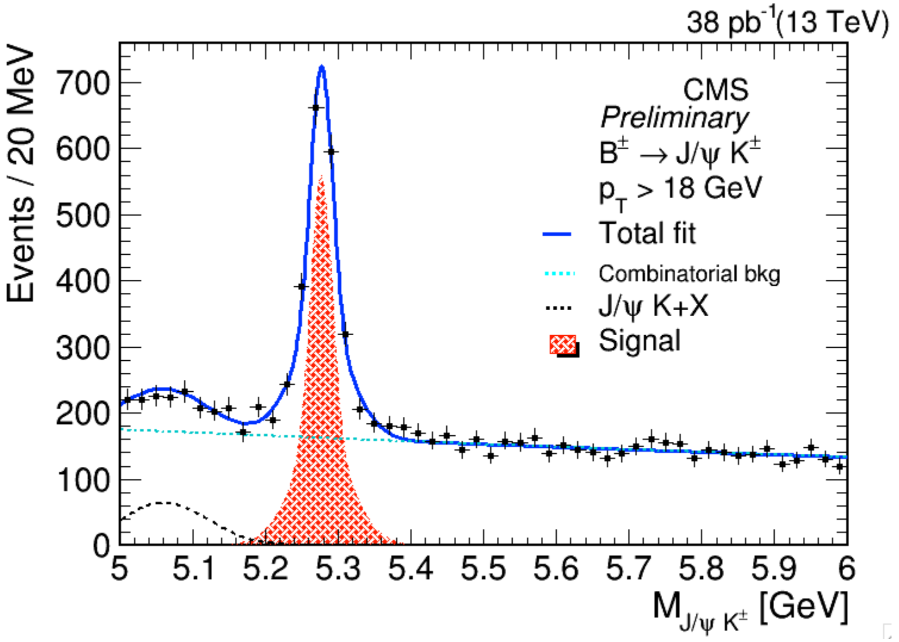
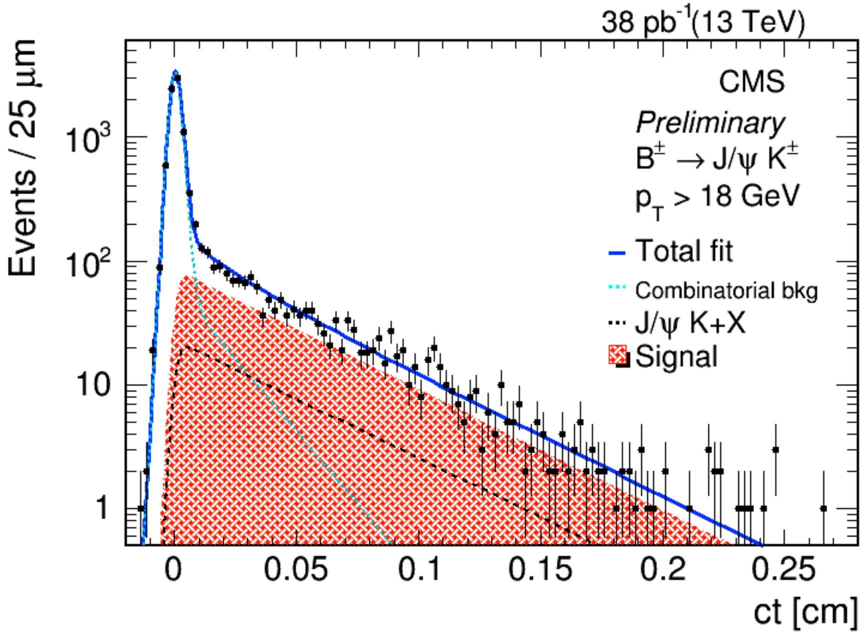
INTERPOLAZIONI BIDIMENSIONALI con ROOFIT

LEZIONE PRATICA per il Corso di Dottorato

Docente: A.Pompili – 8 ottobre 2015 [15.30-19.30]





Questi sono *approved plot* di CMS con i primi dati a $\sqrt{s} = 13 TeV$ **(ringrazio Monika Sharma di CMS) ma non saranno oggetto di pubblicazione poiche' sono ottenuti con un trigger inclusivo di** J/ψ **e non con un trigger** J/ψ **displaced (meno fondo!).**

Si vuole interpolare contemporaneamente due osservabili:

- la massa invar. $J/\psi(\mu^+\mu^-)K^\pm$ **con segnale del mesone *B*⁺ (** $B^\pm \to J/\psi K^\pm$ **)**

- il tempo proprio del suddetto spettro

Il fine e' la stima della vita media del mesone *B*⁺.

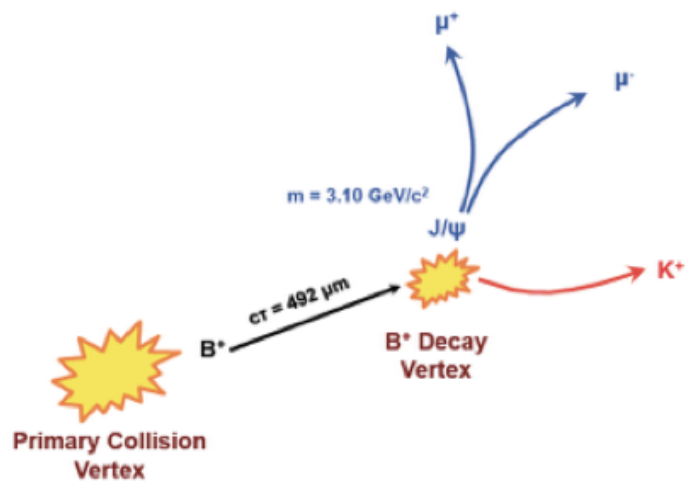
Si ricordi che:

tempo di volo ***distanza di volo***

tempo proprio **:** $t = \frac{t_{LAB}}{\gamma} = \frac{1}{\gamma} \cdot \frac{l_{DEC}}{\beta c}$ $\Rightarrow$ $ct = \frac{l_{DEC}}{\beta\gamma} \equiv \frac{l_{DEC}}{\beta\gamma} \cdot \frac{m_{B^+}^{PDG}}{m_{B^+}^{PDG}} = m_{B^+}^{PDG} \cdot \frac{l_{DEC}}{p_{B^+}}$

Quindi, a seconda che la distanza di volo sia 3D o nel piano trasverso, si ha:

$$ct = m_{B^+}^{PDG} \cdot \frac{l_{DEC}}{p_{B^+}} = m_{B^+}^{PDG} \cdot \frac{l_{DEC}^{\perp}}{p_{B^+}^{\perp}}$$

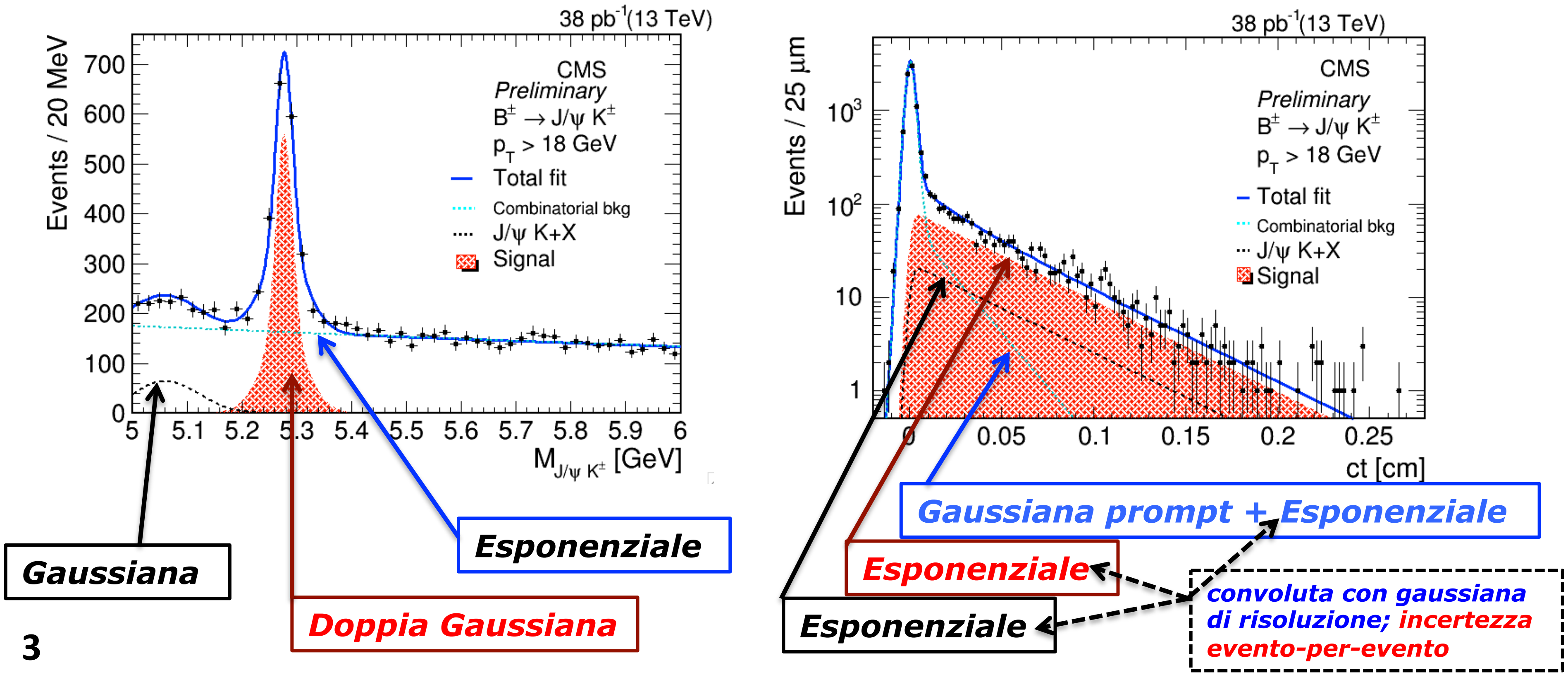


Si ricordi che, indicata con τ la vita media, si ha, per il B⁺: $c\tau_{B^+} \cong 492\mu m$

Prima di passare al dettaglio implementativo in RooFit cerchiamo di capire il modello fisico che definiremo per l'interpolazione.

Segnale : decadimenti $B^{\pm} \to J/\psi\, K^{\pm}$

Fondo :
- **combinatorio (dominato da prompt J/ψ + traccia random)**
- **fisico (decadimenti del tipo** $B \to J/\psi\, K + X$ **con X non ricostruito)**



RooFit *macro* per il fit bidimensionale: *myfitter2d.cc*

```
#include <TStyle.h>
#include <TAxis.h>
#include <TLatex.h>
#include <TPaveText.h>
#include <TFile.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TNtupleD.h>
#include <TH1D.h>
//
#include <RooRealVar.h>
#include <RooDataSet.h>
#include <RooGaussian.h>
#include <RooChebychev.h>
#include <RooExponential.h>
#include <RooAddPdf.h>
#include <RooProdPdf.h>
#include <RooDecay.h>
#include <RooGaussModel.h>
#include <RooAddModel.h>
#include <RooPlot.h>
//
#include "myloop.h"
#include "plotDressing2D.h"

using namespace RooFit;

// General fitting options
#define NUMBER_OF_CPU          1
#define DO_MINOS               kTRUE
// 0 - w/o DISPLAY
// 1 - w/  DISPLAY
#define DISPLAY                1

#define MASS_MIN               5.0
#define MASS_MAX               6.0
#define MASS_PEAK              BP_MASS
#define SOURCE                 "myloop.root"
```

Typical service ROOT classes included

RooFit classes to build fit mode included

Inclusione di file esterni

Istruzione necessaria per usare RooFit

Intervallo di valore di massa & massa nominale del B+ (PDG) [definita in *myloop.h*]

Rootupla di input: *myloop.root*

plotDressing2D.h **definisce opzioni per la canvas e i plot**

myloop.h **definisce la classe per leggere la rootupla in input !**
Viene generata con i comandi *makeClass* (o *makeSelector*) di ROOT.

Nel caso specifico si tratta della classe *ReducedBranches* :

```
class ReducedBranches{
    public:
    int      run;
    int      event;

    int      type;  // B hadron information
    double   mass;
    double   pt;
    double   eta;
    double   phi;
    double   y;
    double   vx;
    double   vy;
    double   vz;
    double   lxy;
    double   lxyz;
    double   errxy;
    double   errxyz;
    double   vtxprob;
    double   cosalpha2d;
    double   cosalpha3d;
    double   ctau2d;
    double   ctau3d;
    double   ctau2derr;
    double   ctau3derr;

    double   ujmass; // dimuon information
    double   ujpt;
    double   ujeta;
    double   ujphi;
    double   ujy;
    double   ujvtxprob;

    double   tktkmass; // ditrack information
    double   tktkpt;
    double   tktketa;
    double   tktkphi;
    double   tktky;
    double   tktkvtxprob;
    double   tktklxy;
    double   tktklxyz;
    double   tktkerrxy;
    double   tktkerrxyz;
    double   tktkblxy;
    double   tktkblxyz;
    double   tktkberrxy;
    double   tktkberrxyz;

    int      mu1idx;
    double   mu1pt;
    double   mu1eta;
    double   mu1phi;
    int      mu2idx;
    double   mu2pt;
    double   mu2eta;
    double   mu2phi;

    int      tk1idx;
    double   tk1pt;
    double   tk1eta;
    double   tk1phi;
    int      tk2idx;
    double   tk2pt;
    double   tk2eta;
    double   tk2phi;

    int nhltbook; // triggers
    int hltbook[N_HLT_BOOKINGS];

    void regTree(TTree *root){
        root->Branch("run",&run,"run/I");
        root->Branch("event",&event,"event/I");
        root->Branch("type",&type,"type/I");
        root->Branch("mass",&mass,"mass/D");
        root->Branch("pt",&pt,"pt/D");
        root->Branch("eta",&eta,"eta/D");
        root->Branch("phi",&phi,"phi/D");
        root->Branch("y",&y,"y/D");
```

•
•
•

Nella prima parte riportata c'e' l'insieme delle dichiarazioni delle variabili contenute nella rootupla (che puo' essere ispezionata nel solito modo: con il *TBrowser*).

A mano l'analista puo' aggiungere altre dichia-razioni di visibilita' (*scope*) generale, come, p.es., delle costanti :

```
#define MUON_MASS     0.10565837
#define PION_MASS     0.13957018
#define KAON_MASS     0.493677
#define KSHORT_MASS   0.497614
#define KSTAR_MASS    0.89594
#define PHI_MASS      1.019455
#define JPSI_MASS     3.096916
#define PSI2S_MASS    3.686109
#define PROTON_MASS   0.938272046
#define LAMBDA_MASS   1.115683
#define BP_MASS       5.27926
#define B0_MASS       5.27958
#define BS_MASS       5.36677
#define BC_MASS       6.2756
#define LAMBDAB_MASS  5.6195
```

La prima parte di *myfitter2d.cc* legge la rootupla per ricavare terne di valori (una terna per candidato B^+). La terna consiste nei valori di : 1) *massa*, 2) tempo proprio (*ct*), 3) errore su tempo proprio (*cterr*).

```
void myfitter2d()
{
  // define variables: mass, proper time and error on proper tim:
  RooRealVar mass("mass","mass",MASS_MIN,MASS_MAX);
  RooRealVar ct("ct","ct",-0.02,0.28);
  RooRealVar cterr("cterr","cterr",0.0001,0.008);

  //outout
  TFile *fout = new TFile("myfitter2d.root","recreate");      // output file
  TNtupleD *_nt = new TNtupleD("_nt","_nt","mass:ct:cterr"); // output ntuple

  // input
  TFile *fin = new TFile(SOURCE);
  TTree *tin = (TTree*)fin->Get("ntkp");

  // setting up rootuple for reading
  ReducedBranches br;
  br.setbranchadd(tin);

  // reading rootuple
  for (int evt=0;evt<tin->GetEntries();evt++)
     {
      tin->GetEntry(evt);
      //
      // cuts to select events/cands
      if (br.hltbook[HLT_Dimuon16_Jpsi_v1]!=1) continue;
      if (br.vtxprob<=0.15) continue;
      if (br.tk1pt<=2.0) continue;
      //
      // filling the 3D vector in the output ntuple
      double var[3];
      var[0] = br.mass;
      var[1] = br.ctau2d;
      var[2] = br.ctau2derr;
      _nt->Fill(var);
     }
  //
  fin->Close();
  //
  // the dataset contains only the 3 variables of interest
  RooDataSet *data = new RooDataSet("data","data",_nt,RooArgSet(mass,ct,cterr));
  //
  /////////////////////////////////////////////////////////
  //
```

Rootupla di ouput:
myfitter2D.root

Rootupla di input

Ulteriore selezione di eventi/candidati

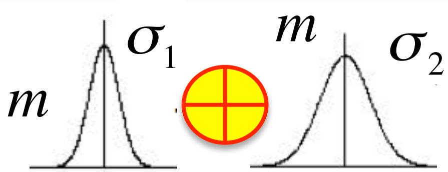
La ntupla nel file di output viene riempita

RooDataSet con la terna di variabili

Costruzione della PDF di segnale :

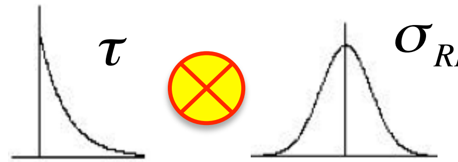
Doppia Gaussiana con media comune (e larghezza diversa);
Trattasi di *somma (RooAddPdf) con coefficiente pari alla "frazione"*

```
//
// signal PDF
//===========
//
// double gaussian for the signal in mass
//
RooRealVar m_mean("m_mean","m_mean",MASS_PEAK,MASS_MIN,MASS_MAX);
RooRealVar m_sigma1("m_sigma1","m_sigma1",0.016,0.001,0.045);
RooRealVar m_sigma2("m_sigma2","m_sigma2",0.035,0.001,0.090);
RooRealVar m_fraction("m_fraction","m_fraction",0.5);
//
RooGaussian m_gaussian1("m_gaussian1","m_gaussian1",mass,m_mean,m_sigma1);
RooGaussian m_gaussian2("m_gaussian2","m_gaussian2",mass,m_mean,m_sigma2);
//
RooAddPdf pdf_m_signal("pdf_m_signal","pdf_m_signal",RooArgList(m_gaussian1,m_gaussian2),RooArgList(m_fraction));
//
// exponential convoluted with gausian resolution for the signal in ct
//
RooRealVar res_sig_mean("res_sig_mean","res_sig_mean",0.0,-1.,1.);
RooRealVar res_sig_sigma("res_sig_sigma","res_sig_sigma",1.0,0.3,2.0);
//
RooGaussModel res_signal("res_signal","res_signal",ct,res_sig_mean,res_sig_sigma,cterr);
//
RooRealVar ctau("ctau","ctau",0.04911,0.010,0.090);
RooDecay pdf_t_signal("pdf_t_signal","pdf_t_signal",ct,ctau,res_signal,RooDecay::SingleSided);
//
// bidimensional signal pdf
//
RooProdPdf pdf_signal("pdf_signal","pdf_signal", RooArgSet(pdf_m_signal, pdf_t_signal));
//
```



coefficiente :
$cG_1 + (1\text{-}c)G_2$

S σ_i



$$\sigma_{RES} = S\sigma_i$$

Esponenziale convoluta con gaussiana di risoluzione sperimentale in tempo proprio

PDF bidimensionale del segnale (*RooProdPdf* delle due PDF per le due variabili)

***Scale Factor* to take into account eventual systematic under/over-estimation of proper-time event-by-event error**

class RooDecay: public RooAbsAnaConvPdf

Single or double sided decay function that can be analytically convolved with any RooResolutionModel implementation

Function Members (Methods)

public:

```
            virtual  ~RooDecay()
      static TClass*  Class()
    virtual TObject*  clone(const char* newname) const
    virtual Double_t  coefficient(Int_t basisIndex) const
        virtual void  generateEvent(Int_t code)
       virtual Int_t  getGenerator(const RooArgSet& directVars, RooArgSet& generateVars, Bool_t staticInitOK = kTRUE) const
     virtual TClass*  IsA() const
           RooDecay&  operator=(const RooDecay&)
                      RooDecay()
                      RooDecay(const RooDecay& other, const char* name = 0)
                      RooDecay(const char* name, const char* title, RooRealVar& t, RooAbsReal& tau, const RooResolutionModel&
                          model, RooDecay::DecayType type)
        virtual void  ShowMembers(TMemberInspector& insp) const
        virtual void  Streamer(TBuffer&)
                void  StreamerNVirtual(TBuffer& ClassDef_StreamerNVirtual_b)
```

protected:

Data Members

public:

```
static RooDecay::DecayType  DoubleSided
static RooDecay::DecayType  Flipped
static RooDecay::DecayType  SingleSided
```

protected:

```
                Int_t  _basisExp
         RooRealProxy  _t
         RooRealProxy  _tau
  RooDecay::DecayType  _type
```

Documentazione:
- **roofit.sourceforge.net**
- **https://root.cern.ch/root/html/**

RooGaussModel

res_signal

Il costruttore della classe *RooDecay* e':

RooDecay (const char* name, const char* title, RooRealVar& t, RooAbsReal& tau, const RooResolutionModel& model, RooDecay::DecayType type)

variabile *ct*   **parametro τ**

SingleSided

class RooGaussModel: public RooResolutionModel

Class RooGaussModel implements a RooResolutionModel that models a Gaussian distribution. Object of class RooGaussModel can be used for analytical convolutions with classes inheriting from RooAbsAnaConvPdf

Function Members (Methods)

public:

```
virtual              ~RooGaussModel ()
void                 advertiseAymptoticIntegral (Bool_t flag)
void                 advertiseFlatScaleFactorIntegral (Bool_t flag)
virtual Double_t     analyticalIntegral (Int_t code, const char* rangeName) const
virtual Int_t        basisCode (const char* name) const
static TClass*       Class ()
virtual TObject*     clone (const char* newname) const
virtual void         generateEvent (Int_t code)
virtual Int_t        getAnalyticalIntegral (RooArgSet& allVars, RooArgSet& analVars, const char* rangeName = 0) const
virtual Int_t        getGenerator (const RooArgSet& directVars, RooArgSet& generateVars, Bool_t staticInitOK = kTRUE) const
virtual TClass*      IsA () const
RooGaussModel&       operator= (const RooGaussModel&)
                     RooGaussModel ()
                     RooGaussModel (const RooGaussModel& other, const char* name = 0)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma,
                         RooAbsReal& msSF)
                     RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma,
                         RooAbsReal& meanSF, RooAbsReal& sigmaSF)
virtual void         ShowMembers (TMemberInspector& insp) const
virtual void         Streamer (TBuffer&)
void                 StreamerNVirtual (TBuffer& ClassDef_StreamerNVirtual_b)
```

protected:

```
static complex<Double_t>  evalCerf (Double_t swt, Double_t u, Double_t c)
static complex<Double_t>  evalCerfApprox (Double_t swt, Double_t u, Double_t c)
complex<Double_t>         evalCerfInt (Double_t sign, Double_t wt, Double_t tau, Double_t umin, Double_t umax, Double_t c) const
virtual Double_t          evaluate () const
```

Data Members

•

•

•

Uno dei costruttori della classe *RooGaussModel* e' :

RooGaussModel (const char* name, const char* title, RooRealVar& x, RooAbsReal& mean, RooAbsReal& sigma, RooAbsReal& msSF)

ct t_{RES} S

cterr

Costruzione della PDF del fondo combinatorio (traccia random):

Vita media *del fondo combinatorio*

Si ricorre alla stessa funzione di risoluzione gaussiana usata per il segnale: *res_signal*

```
//
// combinatorial background PDF (prompt or non-prompt J/psi + random track)
//====================================================================
//
// exponential for the combinatorial background in mass
//
RooRealVar m_par1("m_par1","m_par1",-0.3,-2.,+2.);
RooExponential pdf_m_combinatorial("pdf_m_combinatorial","pdf_m_combinatorial",mass,m_par1);
//
// exponential convoluted with gaussian resolution for the non-prompt background in ct
//
RooRealVar ctau_nonprompt("ctau_nonprompt","ctau_nonprompt",0.0500, 0.0010, 0.1000);
RooDecay pdf_t_nonprompt("pdf_t_nonprompt","pdf_t_nonprompt",ct,ctau_nonprompt,res_signal,RooDecay::SingleSided);
//
// Sum of gaussian resolution function (res_signal) for prompt background in ct and the previous exponential for NP-bkg
//
RooRealVar prompt_fraction("prompt_fraction","prompt_fraction",0.5,0.0,1.0);
//
RooAddPdf pdf_t_combinatorial("pdf_t_combinatorial","pdf_t_combinatorial",RooArgList(res_signal,pdf_t_nonprompt),RooArgList(prompt_fraction));
//
// bidimensional combinatorial-bkg pdf
//
RooProdPdf pdf_combinatorial("pdf_combinatorial","pdf_combinatorial",RooArgSet(pdf_m_combinatorial,pdf_t_combinatorial));
//
```

Costruzione della PDF del fondo fisico:

Si ricorre alla stessa funzione di risoluzione gaussiana usata per il segnale ed il fondo combinatorio NP: ***res_signal***

```
//
// B->J/psi+track+X background PDF
//==============================
//
// single gaussian for the physical background in mass
//
RooRealVar m_jpsix_mean("m_jpsix_mean","m_jpsix_mean",5.1,5.0,5.3);
RooRealVar m_jpsix_sigma("m_jpsix_sigma","m_jpsix_sigma",0.05,0.01,0.10);
RooGaussian pdf_m_jpsix("pdf_m_jpsix","pdf_m_jpsix",mass,m_jpsix_mean,m_jpsix_sigma);
//
// exponential convoluted with gaussian resolution for the physical background in ct
//
RooRealVar ctau_jpsix("ctau_jpsix","ctau_jpsix",0.0500, 0.0010, 0.1000);
RooDecay pdf_t_jpsix("pdf_t_jpsix","pdf_t_jpsix",ct,ctau_jpsix,res_signal,RooDecay::SingleSided);
//
// bidimensional physical-bkg pdf
//
RooProdPdf pdf_jpsix("pdf_jpsix","pdf_jpsix",RooArgSet(pdf_m_jpsix, pdf_t_jpsix));
//
```

Vita media ***del fondo fisico***

Costruzione del modello 2D complessivo (segnale+2fondi) :

```
//
// FULL MODEL (SIGNAL + 2 BKGS)
//
// define coefficients for addition of the 3 pdfs
//
RooRealVar n_signal("n_signal","n_signal",n_signal_initial,0.,data->sumEntries());
RooRealVar n_combinatorial("n_combinatorial","n_combinatorial",n_combinatorial_initial,0.,data->sumEntries());
RooRealVar n_jpsix("n_jpsix","n_jpsix",200.,0.,data->sumEntries());
//
RooAddPdf model("model","model",
               RooArgList(pdf_signal, pdf_combinatorial, pdf_jpsix),
               RooArgList(n_signal, n_combinatorial, n_jpsix));
//
```

c_1 c_2 c_3

```
RooAddPdf is an efficient implementation of a sum of PDFs of the form

 c_1*PDF_1 + c_2*PDF_2 + ... c_n*PDF_n

or

 c_1*PDF_1 + c_2*PDF_2 + ... (1-sum(c_1...c_n-1))*PDF_n

The first form is for extended likelihood fits, where the
expected number of events is Sum(i) c_i. The coefficients c_i
can either be explicitly provided, or, if all components support
extended likelihood fits, they can be calculated the contribution
of each PDF to the total number of expected events.

In the second form, the sum of the coefficients is enforced to be one,
and the coefficient of the last PDF is calculated from that condition.
```

Il # di candidati di segnale e di fondo combinatorio vengono in precedenza dichiarati ed inizializzati

```
// initialization
//
double n_signal_initial = data->sumEntries(TString::Format("abs(mass-%g)<0.015",MASS_PEAK))
- data->sumEntries(TString::Format("abs(mass-%g)<0.030&&abs(mass-%g)>0.015",MASS_PEAK,MASS_PEAK));
//
double n_combinatorial_initial = data->sumEntries() - n_signal_initial;
//
```

Interpolazione (e plotting) !

Extended(kTRUE) ? NON SERVE !

```
   //
   // finally go for fitting !
   //
   model.fitTo(*data,Minos(DO_MINOS),NumCPU(NUMBER_OF_CPU),Offset(kTRUE));
   //
   // go to display plots with fits superimposed on data distributions
#if DISPLAY
   //
   // Display mass plots
   //-------------------
   //
   TCanvas *c1 = canvasDressing("c1");
   //
   RooPlot* frame_m = mass.frame();
   //
   TH1D* histo_data_m = (TH1D*)data->createHistogram("histo_data_m", mass, Binning(50,mass.getMin(),mass.getMax()));
   //
      •
      •
      •
   //
   // Display c*proper-time plots
   //----------------------------
   //
   TCanvas *c2 = canvasDressing("c2");
   //
   RooPlot* frame_t = ct.frame();
   //
   TH1D* histo_data_t = (TH1D*)data->createHistogram("histo_data_t", ct, Binning(120,ct.getMin(),ct.getMax()));
   //
      •
      •
      •
```

Nota bene: il fit e' automaticamente del tipo EXTENDED !

Infatti:

- If **RooAddPdf** is given N coefficients instead of N-1 fractions
  → **RooAddPdf** is automatically extended
  → coefficients represent the expected #events for each PDF comp.

[da: http://roofit.sourceforge.net/docs/tutorial/intro/roofit_tutorial_intro.pdf]

Per eseguire la macro: ***.x myfitter2d.cc***

Oltre ad ottenere il plot della slide iniziale si provi a commentare il risultato del fit.

```
**********
**   23 **MINOS        7500
**********
FCN=-2992.64 FROM MINOS     STATUS=SUCCESSFUL    4324 CALLS        6110 TOTAL
                     EDM=5.11676e-05    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                  PARABOLIC         MINOS ERRORS
 NO.   NAME      VALUE            ERROR      NEGATIVE      POSITIVE
  1  ctau         4.44363e-02   1.25997e-03  -1.23372e-03   1.28786e-03
  2  ctau_jpsix   4.56622e-02   2.73766e-03  -2.61995e-03   2.87062e-03
  3  ctau_nonprompt   1.86478e-02   1.67187e-03  -1.73138e-03   1.71269e-03
  4  m_jpsix_mean   5.06123e+00   6.43382e-03  -7.33523e-03   5.85437e-03
  5  m_jpsix_sigma   6.06444e-02   5.98874e-03  -5.38280e-03   6.85032e-03
  6  m_mean       5.27737e+00   6.90799e-04  -6.93813e-04   6.88152e-04
  7  m_par1      -2.69325e-01   4.14541e-02  -4.14491e-02   4.14688e-02
  8  m_sigma1     4.09537e-02   3.02175e-03  -2.71070e-03   3.87860e-03
  9  m_sigma2     1.50309e-02   6.81301e-04  -6.68188e-04   6.96192e-04
 10  n_combinatorial   7.67237e+03   9.21780e+01  -9.20165e+01   9.23685e+01
 11  n_jpsix      4.13610e+02   2.59086e+01  -2.54124e+01   2.64232e+01
 12  n_signal     1.54505e+03   4.63785e+01  -4.56911e+01   4.72517e+01
 13  prompt_fraction   9.41981e-01   4.97410e-03  -5.06926e-03   4.87957e-03
 14  res_sig_mean   2.27907e-01   1.68816e-02  -1.68765e-02   1.68933e-02
 15  res_sig_sigma   1.28223e+00   1.33976e-02  -1.32768e-02   1.35262e-02
                               ERR DEF= 0.5
```