Some Handy Properties of the Gaussian

Andriy Zatserklyaniy, zatserkl@fnal.gov

September 12, 2016

1 Area of the Gaussian

Let's consider histogram with bin width of 1/2 units (MeV on the figure), Fig.1.

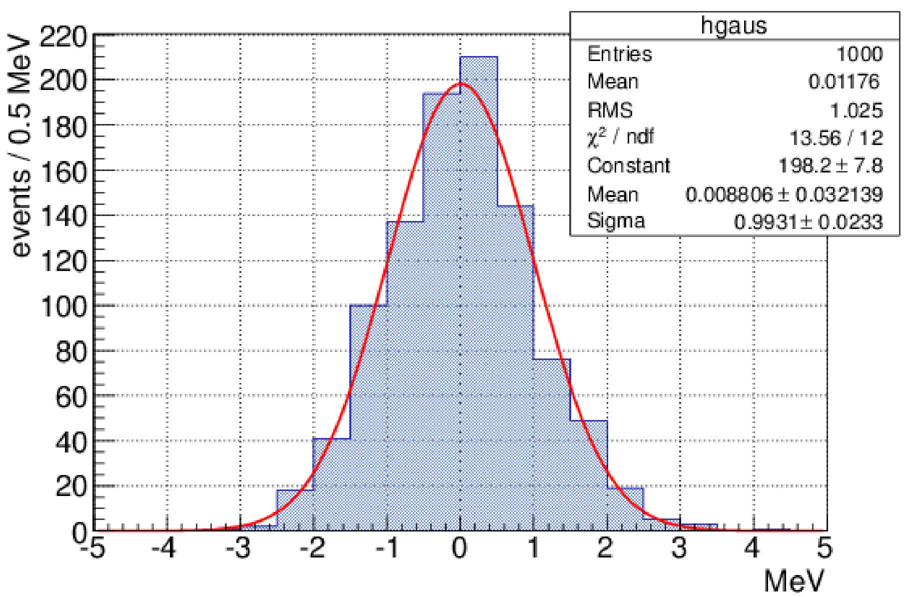

Figure 1: Histogram and Gaussian fit

Calculate the number of events in the histogram. The number of events is proportional to the histogram area

$$S = N \cdot s_1$$

where s_1 is area which corresponds to one event. The area of every histogram bin is a product of the bin width w and the number of events in the bin. Therefore, the area which corresponds to one event is a product of the bin width w and 1:

$$s_1 = w \cdot 1 = w$$

To calculate an area of the Gaussian-shape histogram in the Fig.1 let's approximate it by the area of the Gaussian

$$g(x) = Ae^{-\dfrac{(x-x_0)^2}{2\sigma^2}}$$

Fit results are:

$$A = 198.2$$
$$\sigma = 0.993$$

Calculate the area under the Gaussian

$$\begin{aligned} S = &\int_{-\infty}^{+\infty} Ae^{-\frac{(x-x_0)^2}{2\sigma^2}} \\ =&\sqrt{2\pi}\sigma A \underbrace{\int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_0)^2}{2\sigma^2}}}_{=1} \\ =&\sqrt{2\pi}\sigma A \\ \approx&2.5 \cdot \sigma \cdot A \end{aligned}$$

Express this area in terms of the number of events N and the area of the one event s_1:

$$\begin{aligned} S =&N \cdot s_1 \\ =&N \cdot w \end{aligned}$$

hence

$$N = S/w$$

or

$$N = \sqrt{2\pi} \cdot \sigma \cdot A/w$$

Because $\sqrt{2\pi} \approx 2.5066$

$$N \approx 2.5 \cdot \sigma \cdot A/w$$

The histogram on the Fig.1 was generated for 1000 events Gaussianly-distributed with $\sigma = 1$. In our approximation

the area under the Gaussian

$$S \approx 2.5 \cdot 198.2 \cdot 0.993 = 492.0$$

the number of events for $w = 0.5$ MeV

$$N \approx 2.5 \cdot 198.2 \cdot 0.993/0.5 = 984.1$$

2 Relation between the σ and the FWHM

Consider a Gaussian with a mean of 0 in in the form of (see Fig.2)

$$g(x) = Ae^{-\frac{x^2}{2\sigma^2}}, \qquad A = \frac{1}{\sqrt{2\pi}\sigma}$$

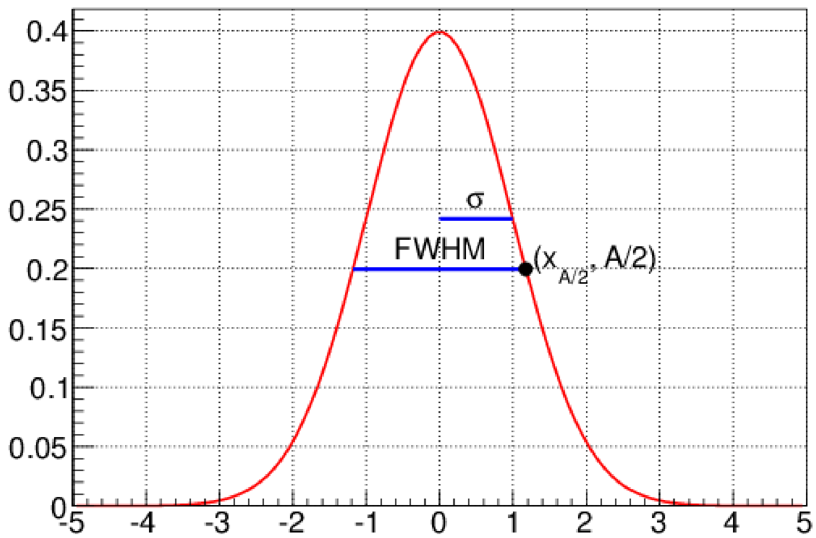

Figure 2: Gaussian with area 1: $A = \frac{1}{\sqrt{2\pi}\sigma}$, mean = 0 and $\sigma = 1$.

Let the Gaussian have half of the maximum value, $A/2$, at $x_{A/2}$. Then

$$A/2 = Ae^{-x_{A/2}^2/2\sigma^2}$$
$$log2 = \frac{x_{A/2}^2}{2\sigma^2}$$
$$x_{A/2} = \sqrt{2log2} \cdot \sigma$$

and

$$FWHM = 2x_{A/2} = \sqrt{8log2} \cdot \sigma$$

or

$$FWHM = \sqrt{log256} \cdot \sigma$$

a popular approximation is

$$FWHM \approx 2.35 \cdot \sigma$$

3 Integral over the Gaussian

$$\begin{aligned}
S(x) &= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{x} e^{-\dfrac{t^2}{2\sigma^2}}\,dt \\
&= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{0} e^{-\dfrac{t^2}{2\sigma^2}}\,dt + \frac{1}{\sqrt{2\pi}\sigma}\int_{0}^{x} e^{-\dfrac{t^2}{2\sigma^2}}\,dt \\
&= \frac{1}{2} + \frac{1}{\sqrt{2\pi}\sigma}\int_{0}^{x} e^{-\frac{t^2}{2\sigma^2}}\,dt \\
&\left| \begin{matrix} z = \dfrac{t}{\sqrt{2}\sigma} & t = \sqrt{2}\sigma z \\ t = x & dt = \sqrt{2}\sigma dz \end{matrix} \right| \\
&= \frac{1}{2} + \frac{1}{\sqrt{\cancel{2}\pi\cancel{\sigma}}}\sqrt{\cancel{2}\cancel{\sigma}}\int_{0}^{x} e^{-z^2}\,dz \\
&= \frac{1}{2} + \frac{1}{2}\frac{2}{\sqrt{\pi}}\int_{0}^{\dfrac{x}{\sqrt{2}\sigma}} e^{-z^2}\,dz \\
&= \frac{1}{2} + \frac{1}{2} erf\frac{x}{\sqrt{2}\sigma} \\
&= \frac{1}{2}(1 + erf\frac{x}{\sqrt{2}\sigma})
\end{aligned}$$

Finally,

$$S(x) = \frac{1}{2}(1 + erf\frac{x}{\sqrt{2}\sigma})$$

The $S(x)$ is shown in the Fig.3. The error function $erf(x)$ is shown in the Fig.4. Note, that

$$\begin{aligned}
S(-\frac{FWHM}{2}) &= 0.12 \\
S(\frac{FWHM}{2}) &= 0.88
\end{aligned}$$

See Appendix A for the ROOT macro code which was used to create the Figures.

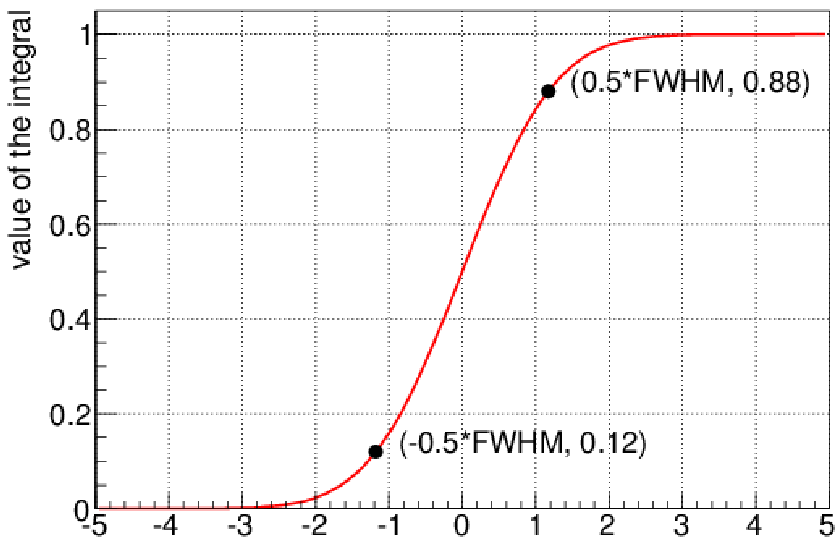

Figure 3: Integral over the Gaussian $S(x) = \frac{1}{2}(1 + erf\,\frac{x}{\sqrt{2}\sigma})$

A ROOT macro hgaus.C

Listing 1: ROOT macro hgaus.C

```
#include <TROOT.h>
#include <TH1.h>
#include <TF1.h>
#include <TMath.h>
#include <TCanvas.h>
#include <TMarker.h>
#include <TLatex.h>
#include <TLine.h>
#include <iostream>

using std::cout;    using std::endl;

void hgaus()
{
  Double_t area = 100;

  TH1F* hgaus = new TH1F("hgaus", ";MeV;events / 0.5 MeV", 20, -5,
      5);
  hgaus->SetFillStyle(3001);
  hgaus->SetFillColor(38);

  hgaus->FillRandom("gaus", 1000);
  new TCanvas;
```

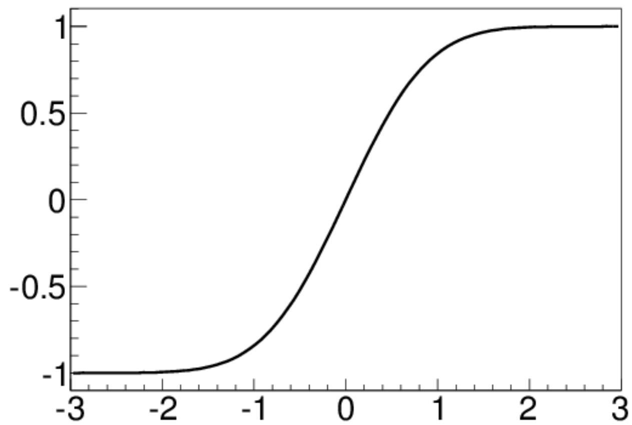

Figure 4: Error function $erf(x)$

```
hgaus->Draw();
hgaus->Fit("gaus");
gPad->SaveAs("hgaus.eps");
gPad->SaveAs("hgaus.png");

// picture to show relation between the FWHM and sigma

TF1* fgaus = new TF1("fgaus", "gaus", -5,5);
// fgaus->SetTitle("Gaussian y(x) = #frac{1}{#sqrt{2#pi}#sigma}
    exp(- #frac{x^{2}}{2#sigma^{2}})");
fgaus->SetTitle("");

Double_t mean = 0;
Double_t sigma = 1;
// Double_t A = area/(TMath::Sqrt(2.*TMath::Pi()) * sigma);
    // normalize the area to 100
Double_t A = 1./(TMath::Sqrt(2.*TMath::Pi()) * sigma);      //
    normalize the area to 1

fgaus->SetParameter(0, A);
fgaus->SetParameter(1, mean);
fgaus->SetParameter(2, sigma);

Double_t halfmax_y = A / 2.;
Double_t halfmax_x1 = fgaus->GetX(halfmax_y, mean, mean - 3.*
    sigma);
Double_t halfmax_x2 = fgaus->GetX(halfmax_y, mean, mean + 3.*
    sigma);

// FWHM line
TMarker* marker = new TMarker(halfmax_x2, halfmax_y, 20);
TLine* line_fwhm = new TLine(halfmax_x1, halfmax_y, halfmax_x2,
    halfmax_y);
line_fwhm->SetLineWidth(3);
```

```
line_fwhm->SetLineColor(4);
TLatex* text_fwhm = new TLatex(0, halfmax_y+0.02*A, "FWHM");
text_fwhm->SetTextAlign(21);
TLatex* text_marker = new TLatex(halfmax_x2+0.1*sigma, halfmax_y,
    "(x_{A/2}, A/2)");
text_marker->SetTextAlign(11);

// sigma line
Double_t sigma_x = sigma;
Double_t sigma_y = fgaus->Eval(sigma_x);    // A*exp(-1/2) = A/
    sqrt(e)
TLine* line_sigma = new TLine(0, sigma_y, sigma_x, sigma_y);
line_sigma->SetLineWidth(3);
line_sigma->SetLineColor(4);
TLatex* text_sigma = new TLatex((0+sigma_x)/2, sigma_y+0.02*A, "#
    sigma");
text_sigma->SetTextAlign(21);

new TCanvas;
fgaus->Draw();
line_fwhm->Draw();
marker->Draw();
text_fwhm->Draw();
text_marker->Draw();
line_sigma->Draw();
text_sigma->Draw();
gPad->SaveAs("fgaus.eps");
gPad->SaveAs("fgaus.png");

// integral over gaussian

cout<< "\nIntegral function" <<endl;

TF1* fint_gaus = new TF1("fint_gaus", "[0] * 0.5*(1.+TMath::Erf((
    x-[1])/(TMath::Sqrt(2)*[2])))", -5,5);
// fint_gaus->SetParameter(0, area);      // normalize the area to
    100
fint_gaus->SetParameter(0, 1);      // normalize the area to 1
fint_gaus->SetParameter(1, mean);
fint_gaus->SetParameter(2, sigma);
// fint_gaus->SetTitle("Integral over Gaussian: #frac{1}{2}(1+erf
    (#frac{x}{#sqrt{2}#sigma}))");
fint_gaus->SetTitle("");
fint_gaus->GetYaxis()->SetTitle("value of the integral");
new TCanvas;
fint_gaus->Draw();
Double_t offset = 0.3;
marker->DrawMarker(halfmax_x1, fint_gaus->Eval(halfmax_x1));
text_marker->DrawText(offset+halfmax_x1, fint_gaus->Eval(
    halfmax_x1), Form("(-0.5*FWHM, %0.2f)", fint_gaus->Eval(
    halfmax_x1)));
marker->DrawMarker(halfmax_x2, fint_gaus->Eval(halfmax_x2));
text_marker->DrawText(offset+halfmax_x2, fint_gaus->Eval(
    halfmax_x2), Form("(0.5*FWHM, %0.2f)", fint_gaus->Eval(
    halfmax_x2)));
gPad->SaveAs("fint_gaus.eps");
gPad->SaveAs("fint_gaus.png");
```

```
}
```