DATA ANALYSIS WITH A. CALDWELL

# SUMMARY AND EXERCISES

April 4, 2018

P. Krause
03663771

# Contents

# Chapter 1

# Introduction to Probabilistic Reasoning

## 1.1 Summary

The first chapter aims to give a first understanding of the basic principles of probabilistic reasoning. From variations of the simple coin toss experiment to the different schools of data analysis. This schools are the Frequentist Statistics - the classical approach, which is quiet objective but can only give statements about how much of the observed data lies within a defined range of expected values. On the other end there is the Bayesian analysis. It uses additional, subjective informations (the prior) to form a statement about the correctness of a given model with the observed data.

Both approaches have their weak points. For the Bayesian analysis the main critique is the subjectivity of the prior, this means different people choosing different priors could end up with different conclusions. The Frequentist approach on the other hand is criticised for being ad-hoc, it misses the force of deductive logic. This can lead to notorious misconceptions, like the p-value which states in how many cases a true null hypothesis is due to randomness rejected. But many people think wrongly that the p-value indicates the probability of the null hypothesis.

But both schools have also arguments in their favour. For example the Bayesian analysis gives a result with which a human mind can work; it tells the probability of a hypothesis. Furthermore varying the prior can show how sensitive the result is to the prior. In defence for the Frequentist analysis is the objectivity: Every analyst will agree on the same p-value. The rejection of the null hypothesis can then be decided by the individual. Furthermore a frequentist analysis needs a careful description of the experiment and methods of analysis before starting. This helps to minimize the bias from the experimentalist.

Concluding one can say there is no better school of analysis, it depends on the situation.

## 1.2 EXERCISES

### 1.2.1 Problem 1

In this problem one meets Jane on the street she tells that she has two children. From that sentence one can extract the following information:

$$\omega = [GG, BG, GB, BB] \tag{1.1}$$

$$P(\omega) = \frac{1}{4} \tag{1.2}$$

Where "G" stands for girls and "B" for boy. P($\omega$) is the probability that Jane has the $\omega$ combination of children. Now Jane says she has pictures of her children in her pocket, pulls one out and shows it; its a girl. Lets call this event "A". The question is now: Whats the probability that the second child of Jane is also a girl. Lets call this event "B". Therefore:

$$P(B|A) = \frac{P(B \cap A)}{P(A)} = \frac{P(GG)}{P(GG) + \frac{1}{2}P(BG) + \frac{1}{2}P(GB)} = \frac{\frac{1}{4}}{\frac{1}{4} + 2\frac{1}{2}\frac{1}{4}} = \frac{1}{2} \tag{1.3}$$

A variation of this problem is that Jane takes out both pictures, looks at them and is required to show a picture of a girl if she has one. Lets call this event "C". Again it is asked what the probability is that she has two girls (event "B"):

$$P(B|C) = \frac{P(GG)}{P(GG) + P(BG) + P(GB)} = \frac{\frac{1}{4}}{\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + 0 \cdot \frac{1}{4}} = \frac{1}{3} \tag{1.4}$$

### 1.2.2 Problem 2

Here one has to go back to section 1.2.3 of the script and come up with more possible definitions for the probability of the data. Since here nearly everything is possible, some definition are given:

- number of switches (TH,HT)
- number of a sequence of length 2 (TT, HH) or length 3 (TTT, HHH) ...
- even or odd number of T or H
- ...

### 1.2.3 Problem 3

In this problem one has a detector which measures energies with a resolution of 10 percent. It is asked if one measures an energy (E), what probabilities one can assign to possible true values of the energy and what the conclusion depends on.

The resolution only distinguishes two adjacent energy peaks (e.g. identify different decays of a measured radionuclide). But one also has influences from:

- How much energy did the particle deposited in the detector
- Conversion of gamma rays into a useful signal
- ...

In a nutshell one has not enough informations to make a statement about the probability that one measured the true energy for a detector signal.

### 1.2.4 Problem 4

In this problem the so called Mongolian swamp fever is investigated. It is such a rare disease that a doctor only expects to meet it once every 10000 patients. The following probabilities connect symptoms with the disease: These probabilities are independent. Now it is asked

| Symptomes | Having the disease | Not having the disease |
|---|---|---|
| Spots | $P(SP\|D) = 1.00$ | $P(SP\|\overline{D}) = 0.03$ |
| Acute lethargy | $P(AL\|D) = 1.00$ | $P(AL\|\overline{D}) = 0.10$ |
| raging thirst | $P(RT\|D) = 0.60$ | $P(RT\|\overline{D}) = 0.02$ |
| violent sneezes | $P(VS\|D) = 0.20$ | $P(VS\|\overline{D}) = 0.05$ |

what the probability is that one has Mongolian swamp fever going to the doctor with all four symptoms (P(D|[SP,AL,RT,VS])):

$$P(D|[SP, AL, RT, VS]) = \frac{P([SP, AL, RT, VS]|D)P(D)}{P([SP, AL, RT, VS])} \tag{1.5}$$

$$= \frac{P(SP|D)P(AL|D)P(RT|D)P(VS|D)P(D)}{P(SP|D)P(AL|D)P(RT|D)P(VS|D) + (1 - P(\overline{D})P(SP|\overline{D})P(AL|\overline{D})P(RT|\overline{D})P(VS|\overline{D})} \tag{1.6}$$

$$= 0.80 \tag{1.7}$$

This means 80 percent of patients with all four symptoms have the disease. Furthermore it is asked for the probability having Mongolian swamp fever with any three out of the four symptoms. This is calculated the same way as before. One yields the following probabilities:

$$P(D|[SP, AL, RT]) \approx 0.457 \tag{1.8}$$

$$P(D|[SP, AL, VS]) \approx 0.516 \tag{1.9}$$

$$P(D|[AL, VS, RT]) = P(D|[SP, RT, VS]) = 0 \tag{1.10}$$

# Chapter 2

# Binomial and Multinomial Distribution

## 2.1 Summary

This chapter takes a closer look on the binomial and multinomial distribution. A binomial distribution gives informations about the number of successes for a series of identical and independent experiments, with two possible outcomes. For example a coin toss has 50 percent chance to land on either side of the coin. For 100 coin tosses the number of times the "experiment" yields heads up, is given by a binomial distribution. Mathematically the binomial distribution is given by:

$$P(r|N,p) = \frac{N!}{r!(N-r)!} p^r (1-p)^{N-r} \tag{2.1}$$

Where N is the number of experiments, r the number of successes and p the probability to succeed. Often the binomial factor is used to simplify the notation:

$$\binom{N}{r} = \frac{N!}{r!(N-r)!} \tag{2.2}$$

The multinomial distribution is essentially the same, but allows more then two outcomes. Mathematically:

$$P(\vec{r}|N,\vec{r}) = \frac{N!}{r_1!r_2!\cdots r_m!} p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m} \qquad \sum_{n=1}^{m} p_n = 1 \tag{2.3}$$

In the following exercises this distributions are applied in the frequentist as well as in the Bayesian interface.

In the frequentist analysis the central interval and the smallest interval are one of many ways used to give the analysis results. The central interval is defined by finding the smallest members of the set of possible outcomes $r_1$ and $r_2$ such that:

$$P(r < r_1) \leq \alpha/2 \qquad P(r > r_2) \leq \alpha/2 \tag{2.4}$$

while maximizing the probabilities. The smallest interval on the other hand is given by the smallest set containing a least a given probability. In general is the central interval over covering that means it contains more probability then the set minimum. In the Bayesian analysis the pendant to the central and smallest interval are the median + central interval and the mode + shortest interval.

In the Frequentist approach there is also the concept of likelihood. Here instead of varying the r-parameter like it is done in the central and smallest interval, r is fixed and p is varied.

$$L(p) = P(r = 5|N = 100, p) \tag{2.5}$$

## 2.2 Exercises

### 2.2.1 Problem 8

In this problem one has the following function

$$P(x) = xe^{-x} \qquad 0 \leq x < \infty \tag{2.6}$$

in the beginning one should find the mean and the standard deviation. The mean calculates like this:

$$E[x] = \mu = \int_0^\infty xP(x)dx = \int_0^\infty x^2e^{-x}dx = \underbrace{\left[-x^2e^{-x}\right]_0^\infty}_{=0} + \int_0^\infty 2xe^{-x}dx \tag{2.7}$$

$$= \underbrace{\left[-2xe^{-x}\right]_0^\infty}_{=0} + \int_0^\infty 2e^{-x}dx \tag{2.8}$$

$$= \left[-2xe^{-x}\right]_0^\infty = 2 \tag{2.9}$$

Next the standard deviation:

$$E[x^2] = \int_0^\infty x^2 P(x) dx = \int_0^\infty x^3 e^{-x} dx = \cdots = 6 \int_0^\infty e^{-x} dx = 6 \tag{2.10}$$

$$\sigma^2 = \mu(x^2) - \mu(x)^2 = 6 - 4 = 2 \tag{2.11}$$

$$\Rightarrow \sigma = \sqrt{2} \tag{2.12}$$

Therefore the content in the interval [mean - standard deviation, mean + standard deviation] is $2 \pm \sqrt{2}$.

In the second part of this exercise one has to find the median and 68 percent central interval. Start with the median (M):

$$\int_0^M Px dx = \int_0^M xe^{-x} dx = \left[-xe^{-x}\right]_0^M + \int_0^M e^{-x} dx \tag{2.13}$$

$$= -Me^{-M} + 0 - \left[e^{-x}\right]_0^M = e^{-M}(M+1) \stackrel{!}{=} \frac{1}{2} \tag{2.14}$$

$$\Rightarrow M \approx 1.67835 \qquad \text{or} \qquad (M = -0.768039) \tag{2.15}$$

Now the central interval:

$$1 - \alpha = 0.68 \Rightarrow \alpha = 0.32 \Rightarrow \frac{\alpha}{2} = 0.16 \tag{2.16}$$

$$\int_0^{x_1} P(x) dx \leq \frac{\alpha}{2} \qquad \Rightarrow x_1 = 0.71 \tag{2.17}$$

$$\int_{x_2}^\infty P(x > x_2) \leq \frac{\alpha}{2} \qquad \Rightarrow x_2 = 3.29 \tag{2.18}$$

$$\Rightarrow [0.71, 3.29]_{0.68}^{CI} \tag{2.19}$$

In the last part one has to find the mode and the 68 percent smallest Interval. Start by finding the mode $(x^*)$:

$$\frac{d}{dx} P(x) = \frac{d}{dx}\left(xe^{-x}\right) = e^{-x}(1-x) \stackrel{!}{=} 0 \tag{2.20}$$

$$\Rightarrow x^* = 1 \Rightarrow P(1) = e^{-1} \approx 0.37 < 0.68 \tag{2.21}$$

To find the smallest interval $[x_1, x_2]$ one has to satisfy the following conditions:

$$0.68 = 1 - \alpha = \int_{x_1}^{x_2} P(x)dx \tag{2.22}$$

$$x_1 < x^* < x_2 \tag{2.23}$$

In this problem the function from A.1 (C++ and ROOT) is used to find the smallest interval. Plugging in the previous calculated mode and a step size of 0.001 one yields the following result:

$$[0.27, 2.49]^{SI}_{0.68} \tag{2.24}$$

### 2.2.2 Problem 10

In this problem the data from table 2.1 is given: For a flat prior for each energy, one has to

| Energy | Trials (N) | Successes (r) |
|---|---|---|
| 0.5 | 100 | 0 |
| 1.0 | 100 | 4 |
| 1.5 | 100 | 20 |
| 2.0 | 100 | 58 |
| 2.5 | 100 | 92 |
| 3.0 | 1000 | 987 |
| 3.5 | 1000 | 995 |
| 4.0 | 1000 | 998 |

**Table 2.1:** Data given in exercise 2.10

find an estimate for the efficiency (success parameter p) as well as an uncertainty. Lets start with Bayes' Theorem:

$$P(p|N,r) = \frac{P(r|N,p)P_0(p)}{P(r|N)} = \frac{P(r|N,p)P_0(p)}{\int P(r|N,p)P_0(p)dp} \tag{2.25}$$

As mentioned one assumes a flat prior thus:

$$P_0(p) = 1 \tag{2.26}$$

$$\Rightarrow P(p|N,r) = \cdots = \frac{(N+1)!}{r!(N-r)!} p^r (1-p)^{N-r} \tag{2.27}$$

From that one can easily obtain the mode.

$$p^* = \frac{r}{N} \tag{2.28}$$

Now the smallest interval for the 68 percent probability range can be calculated with the ROOT script from A.2. This script returns plots showing the smallest interval for the measured energies (see fig. 2.1) and the PDFs of the measurements.

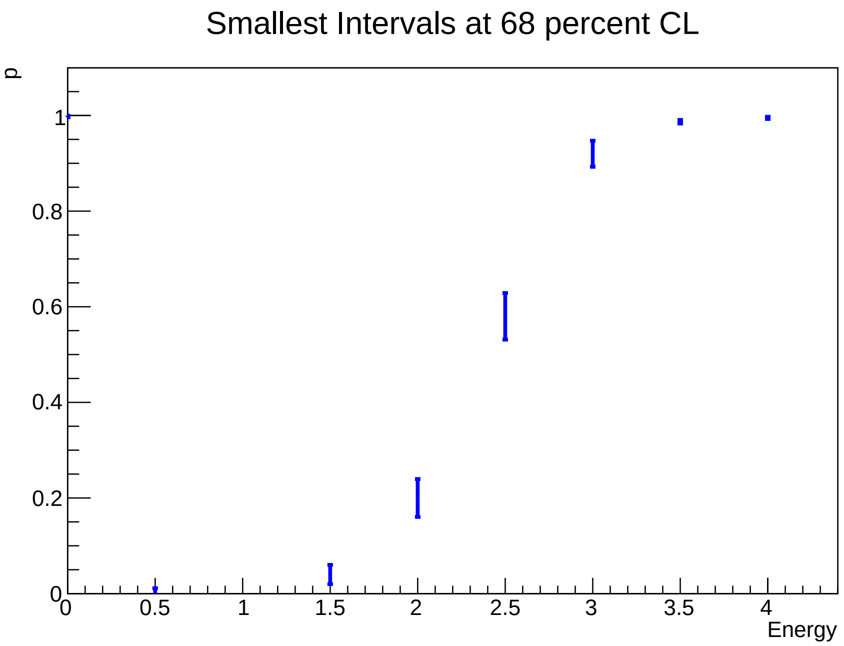


**Figure 2.1:** Smallest intervals at 68 percent for the measurements taken according to the table given in this problem

### 2.2.3 Problem 11

In this problem one has to look once more on the data from the previous task but this time in a frequentist approach by finding the 90 percent confidence interval for p as a function of E. With the code from A.3 one can find the central interval for N trials with a success probability p and 1-$\alpha$.

Now one performs the experiment and finds r successes. With that information, a given confidence level and the code from A.4 one can obtain the probability range.

The code snippet A.5 takes the two previous defined functions to produce a plot of the the 90 percent confidence level interval for p as a function of the energy (see fig.2.2).

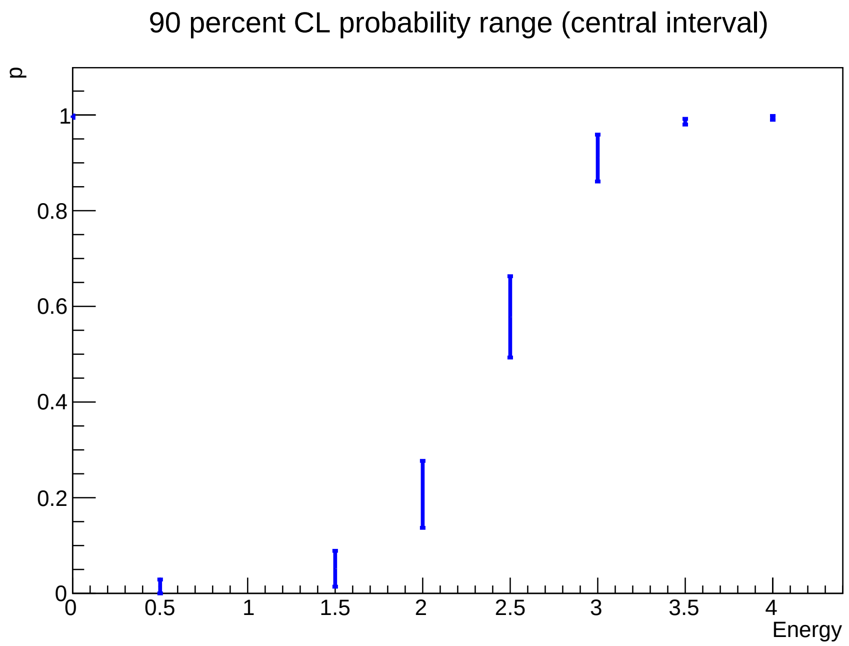


**Figure 2.2:** Confidence intervals of the given data, 90 percent central interval used

### 2.2.4 Problem 13

In this problem one has to investigate what happens if one reuses the a set of data multiple times. It is asked to prove that one obtains the following expression

$$P_n(p|r,N) = \frac{(nN+1)!}{(nr)!(nN-nr)!} p^{nr}(1-p)^{n(N-r)} \tag{2.29}$$

if one reuses the data n times. Starting with a flat prior and then uses the posterior from the first use of the data as the prior for the next use. For the first use of data one gets the same posterior as in exercise 10 of this chapter:

$$P_1(p|N,r) = \cdots = \frac{(N+1)!}{r!(N-r)!} p^r (1-p)^{N-r} \tag{2.30}$$

So after the second use of the data one yields:

$$P_2(p|N,r) = \frac{p^r(1-p)^{N-r}P_1(p|N,r)}{\int_0^1 p^r(1-p)^{N-r}P_1(p|N,r)dp} \tag{2.31}$$

$$= \frac{p^{2r}(1-p)^{2(N-r)}}{\int_0^1 p^{2r}(1-p)^{2(N-r)}dp} \tag{2.32}$$

From that one can easily derive the posterior for the n fold use of the data:

$$P_n(p|N,r) = \frac{p^{nr}(1-p)^{n(N-r)}}{\int_0^1 p^{nr}(1-p)^{n(N-r)}dp} \tag{2.33}$$

$$= \frac{p^{nr}(1-p)^{n(N-r)}}{\beta(nr+1, n(N-r)+1)} \tag{2.34}$$

$$= \frac{(Nn+1)!}{(nr)!(nN-nr)!}p^{nr}(1-p)^{n(N-r)} \qquad \text{q.e.d.} \tag{2.35}$$

The second part of this exercise asks use to calculate the expectation value and variance for p in the limit $n \to \infty$. Start with the expectation value:

$$E_n[p] = \int_0^1 pP_n(p|N,r)dp = \frac{(Nn+1)!}{(nr)!(nN-nr)!}p^{nr+1}(1-p)^{n(N-r)} \tag{2.36}$$

Use the following property of the beta-function to solve the integral:

$$\beta(x+1,y) = \frac{x}{x+y}\beta(x,y) \tag{2.37}$$

$$\Rightarrow E_n[p] = \frac{nr+1}{nN+2} \tag{2.38}$$

$$\Rightarrow \lim_{n\to\infty} E_n[p] \overset{\text{l'Hospital}}{=} \frac{r}{N} \tag{2.39}$$

For the variance we use the property of the beta-function twice and therefore:

$$E_n[p^2] = \frac{nr+1}{nN+2}\frac{nr+2}{nN+3} \tag{2.40}$$

$$\Rightarrow V_n[p] = E_n[p^2] - E_n[p]^2 = \frac{(nr+1)(nN-nr+1}{(nN+2)^2(nN+3)} = \frac{E_n[p](1-E_n[p])}{nN+3} \tag{2.41}$$

$$\Rightarrow \lim_{n\to\infty} V_n[p] = \lim_{n\to\infty} \frac{\mathcal{O}(n^2)}{\mathcal{O}(n^3)} \to 0 \tag{2.42}$$

# Chapter 3

# Poisson Distribution

## 3.1 Summary

The Poisson distribution can be, for the limit of an infinite number of trials, derived from the binomial distribution discussed in the previous chapter. Hence it is the distribution of choice for situation in which only the number of events is known but not the number of trials (except for the fact that it has to be large). Mathematically it is given by:

$$P(n|\nu) = \frac{e^{-\nu}\nu^n}{n!} \tag{3.1}$$

Where n is the number of successes and $\nu$ is the expectation of n. Again one will study this distribution in both schools of data analysis. Furthermore one also takes a look at experiments where a background is present in the data and how to analyse such kind of data (e.g. on/off problem). Also a new interval is introduced; the Feldman-Cousins-Interval. Here the values of n are ranked by:

$$r = \frac{P(n|\mu = \nu + \lambda)}{P(n|\hat{\mu})} \tag{3.2}$$

Where $\lambda$ is the background expectation. The cumulative probability of this ranking is used to form an interval. It follows the same logic as the smallest interval but instead of ranking it by the probability of n for given $\mu$ it is ranked by the probability ratio r.

## 3.2 EXERCISES

### 3.2.1 Problem 4

First one has to find the mean of the function $f(x) = \frac{1}{2}e^{-|x|}$ for $-\infty < x < \infty$

$$f(x) = f(-x) \Rightarrow E[x] = \mu = 0 \tag{3.3}$$

Next the standard deviation is calculated.

$$\sigma^2 = E[x^2] - E[x]^2 = E[x^2] - 0^2 = E[x^2] \tag{3.4}$$

$$E[x^2] = \int_{-\infty}^{\infty} x^2 \frac{1}{2} e^{-|x|} dx = \int_{-\infty}^{0} x^2 \frac{1}{2} e^{x} dx + \int_{0}^{\infty} x^2 \frac{1}{2} e^{-x} dx = 2 \tag{3.5}$$

$$\Rightarrow \sigma = \sqrt{2} \approx 1.414 \tag{3.6}$$

In the second part we were asked to compare the standard deviation to the full width half maximum (FWHM) of the given function.

$$FWHM = 2x_{\frac{1}{2}} \tag{3.7}$$

$$\frac{1}{2} f(x_{\frac{1}{2}}) = \frac{1}{2} f(\mu) = \frac{1}{4} \tag{3.8}$$

$$\Rightarrow \frac{1}{4} = \frac{1}{2} e^{-|x_{\frac{1}{2}}|} \Rightarrow |x_{\frac{1}{2}}| = ln(2) \approx 0.693 \tag{3.9}$$

Comparing now the FWHM with the standard deviation $\frac{FWHM}{\sigma} = \sqrt{2} ln(2) \approx 1$ one yields that they are nearly the same for the given function. Finally it is asked which probability is contained in the $\pm 1$ standard deviation interval around the peak:

$$F_{tot} = \int_{-\infty}^{\infty} f(x) dx = \int_{-\infty}^{0} \frac{1}{2} e^{x} dx + \int_{0}^{\infty} \frac{1}{2} e^{-x} dx = 1 \tag{3.10}$$

$$F_{sigma} = \int_{-\sigma}^{\sigma} f(x) dx \int_{-\sigma}^{0} \frac{1}{2} e^{x} dx + \int_{0}^{\sigma} \frac{1}{2} e^{-x} dx \approx 0.757 \tag{3.11}$$

Therefore 75.7 percent is contained in one standard deviation interval around the peak.

### 3.2.2 Problem 7

In this problem 9 events are observed in an experiment modelled with a Poisson probability distribution. For the first part of this problem one has to assume a flat prior and calculate the 95 percent lower probability limit for the expectation value $\nu$. For that one has to perform a bayesian analysis. Start as usual with:

$$P(\nu|n) = \frac{P(n|\nu)P_0(\nu)}{\int_0^\infty P(n|\nu)P_0(\nu)d\nu} = \frac{\nu^n e^{-\nu} P_0(\nu)}{\int_0^\infty \nu^n e^{-\nu} P_0(\nu)d\nu} \tag{3.12}$$

Assume a flat prior $P_0(\mu) = C = 1/\nu_{max}$ thus

$$P(\nu|n) = \frac{\nu^n e^{-\nu}}{\int_0^{\nu_{max}} \nu^n e^{-\nu} d\nu} \tag{3.13}$$

For $\nu_{max} >> n$ approximate the integral:

$$\int_0^{\nu_{max}} \nu^n e^{-\nu} d\nu \approx \int_0^\infty \nu^n e^{-\nu} d\nu = n! \tag{3.14}$$

$$\Rightarrow P(\nu|n) = \frac{e^{-\nu}\nu^n}{n!} \tag{3.15}$$

The expectation value of $\nu$ is:

$$E[\nu] = \int_0^\infty \nu P(\nu|n) d\nu = n+1 \tag{3.16}$$

Since P is normalized to one, one can also calculate the five percent upper limit:

$$0.05 = F(\nu|9) = \int_0^{\nu_{95}} \frac{e^{-\nu}\nu^9}{9!} \tag{3.17}$$

This is solved with the code snippet A.6. Running the code gives us a 5 percent upper limit of 5.426 which is equal to the lower limit with 95 percent.

In the second part of this exercise it is asked to give the 68 percent confidence level for $\nu$ using the Neyman construction and the smallest interval definition. For this the best solution is to plot the smallest interval of $\nu$ against the number of events (n). This is done with the code from A.7. In figure 3.1 one can find the Neyman plot produced by the code.

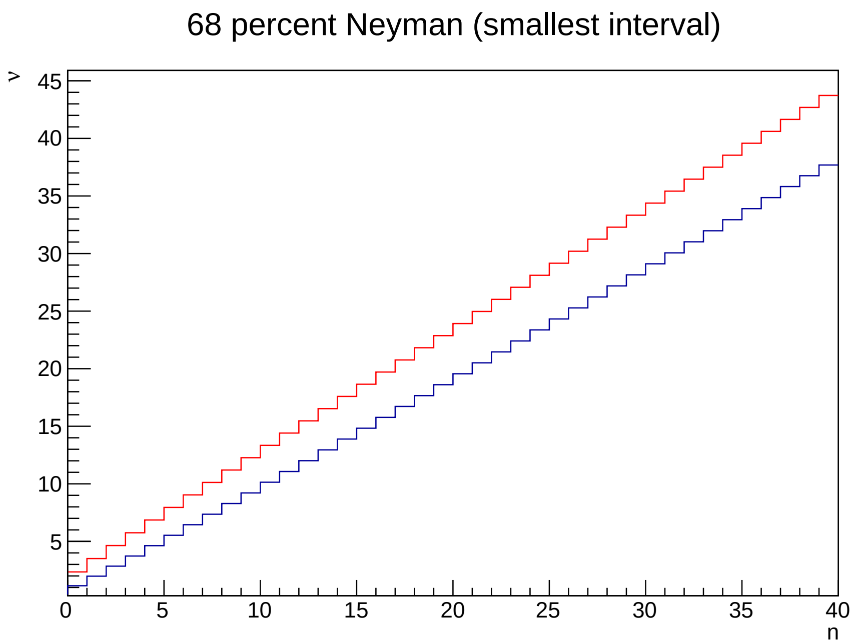


**Figure 3.1:** Neyman plot for 68 percent smallest interval. The red line is the upper limit, the blue one the lower limit.

### 3.2.3 Problem 8

Here one is asked to repeat the previous exercise but this time one assumes to know a background of 3.2 events (with 9 observed events). One starts by finding the Feldman-Cousins 68 percent Confidence Level Interval. It is based on different ranking of possible data outcomes. For that the ratio of the probability assigned to a possible outcome to the probability of that outcome calculated with the value of the Poisson parameter that maximizes the outcomes probability is defined. Mathematically:

$$r = \frac{P(n|\mu = \lambda + \nu)}{P(n|\hat{\mu})} \tag{3.18}$$

Where n are the outcomes, $\lambda$ is the background and $\nu$ the expectation value. The outcomes are ranked according to r and the cumulative probability according to this ranking is used to form an interval. With the code snippet from A.8 the Feldman-Cousins 68 percent CL interval is calculated. This snippet outputs an interval of $[3.135, 9.595]$ (68 percent Feldman-Cousins CL). Next one wants to find the Neyman 68 percent CL interval. Since one can replace $\nu$

with $\mu = \nu + \lambda$ not much has to be done to get the Neyman 68 percent SI CL interval. It was already calculated in the previous exercise (fig. 3.1). One has only to shift the the y-axis by $\lambda$ to obtain the new band plot. One yields a Neyman 68 percent SI CL interval of $[5.09, 8.29]$ at a background of $\lambda = 3.2$. Finally it is asked to find the 68 percent credibility interval for $\nu$. For this Bayes formula reads:

$$P(\nu|n,\lambda) = \frac{P(n|\nu,\lambda)P_0(\nu)}{\int P(n|\nu,\lambda)P_0(\nu)d\nu} \tag{3.19}$$

Using a flat prior and assuming $\nu_{max} \to \infty$ one yields:

$$P(\nu|n,\lambda) = \frac{e^{-\nu}(\lambda+\nu)^n}{n!\sum_{i=0}^n \frac{\lambda^i}{i!}} \tag{3.20}$$

The cumulative probability is given by

$$F(\nu|n,\lambda) = 1 - \frac{e^{-\nu}\sum_{i=0}^n \frac{(\lambda+\nu)^i}{i!}}{\sum_{i=0}^n \frac{\lambda^i}{i!}} \tag{3.21}$$

Now one can give the the Bayesian smallest interval with 68 percent credibility for the given data (n=9,$\lambda = 3.2$). This is done by the snippet A.9.

One yields the interval $[3.12, 9.19]$ (smallest interval at 68 percent credibility). The plot produced by this snippet can be found in figure 3.2

### 3.2.4 Problem 13

In this exercise one takes a look at the relationship between an unbinned likelihood and a binned Poisson probability. A one dimensional density function $f(x|\lambda)$ with a dependency on $\Lambda$ is defined and normalized between [a,b]. n events are measured with x values $x_i$ i = $1, \cdots, n$. The unbinned likelihood is defined as the product of the densities:

$$L(\lambda) = \prod_{I=1}^{n} f(x_i|\lambda) \tag{3.22}$$

Now one considers that the interval is divided into K subintervals (bins). Thus the expectation in bin j:

$$\nu_j = \int_{\Delta_j} f(x|\lambda)dx \tag{3.23}$$

where the integral is over the x range in interval j, which is denoted as $\Delta_j$. Now one is asked to give the probability of the data as a product of Poisson probabilities in each bin. As already

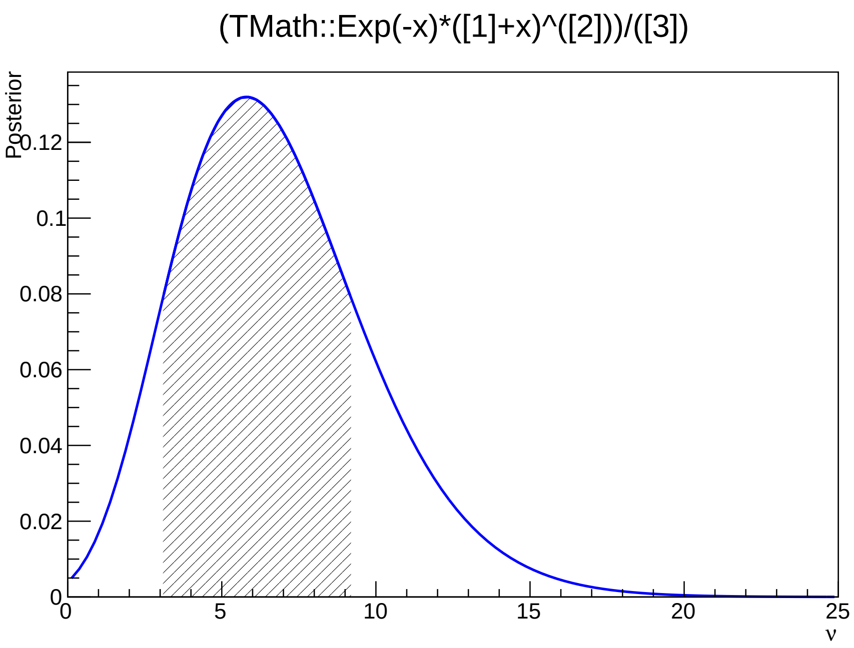


**Figure 3.2:** Blue: Posterior for the given data. Shaded:Bayesian smallest interval at 68 percent credibility

mentioned the Poisson distribution is given by

$$P(n|\nu) = \frac{e^{-\nu}\nu^n}{n!} \tag{3.24}$$

This means for $n_j$ events per bin this equation can be written as:

$$P(n|\nu_1\cdots\nu_K) = \prod_{j=1}^{K} \frac{e^{-\nu_j}\nu_j^n}{n_j!} \qquad \sum_{j=1}^{K} = n \tag{3.25}$$

Now suppose the limit $K \to \infty$ then (if no two measurements yield the same x) each bin will have either $n_j = 1$ or $n_j = 0$ it will be shown, that with this assumption one yields:

$$\lim_{K\to\infty} \prod_{j=1}^{K} \frac{e^{-\nu_j}\nu_j^n}{n_j!} = \prod_{i=1}^{n} f(x_i|\lambda)\Delta \tag{3.26}$$

where $\Delta$ is the interval for all j with fixed size in x. If $K \to \infty$ then $\Delta_j \to 0$ hence:

$$\nu_j = \int\limits_{\Delta_j} f(x|\lambda)dx \overset{K\to\infty}{\to} f(x_j|\lambda)\Delta \tag{3.27}$$

where $\Delta \to 0$. n is the number of non empty bins:

$$\lim_{K\to\infty}\prod_{j=1}^{K}\frac{e^{-\nu_j}\nu_j^{n}}{n_j!}=\lim_{K\to\infty}\prod_{i=1}^{n}\frac{e^{-\nu_i}\nu_i}{1!}\prod_{j=1}^{K-n}\frac{e^{-\nu_j}\nu_j^{0}}{0!}=\lim_{K\to\infty}\prod_{j=1}^{K}e^{-\nu_j}\prod_{i=1}^{n}\nu_i \tag{3.28}$$

Continue with the exponential term:

$$\lim_{K\to\infty}\prod_{j=1}^{K}e^{-\nu_j}=\lim_{K\to\infty}e^{-\sum_{j=1}^{K}\nu_j}=exp\left(-\sum_{j=1}^{K}f(x_j|\lambda)\Delta\right) \tag{3.29}$$

Recalling that $f(x_j|\lambda)$ is normalized and $\Delta \to 0$:

$$exp\left(-\sum_{j=1}^{K}f(x_j|\lambda)\Delta\right)=e^{-\Delta}\to 1 \tag{3.30}$$

Plugging this into (3.28) the proof completes:

$$\lim_{K\to\infty}\prod_{j=1}^{K}\frac{e^{-\nu_j}\nu_j^{n}}{n_j!}=\prod_{i=1}^{n}\nu_i=\prod_{i=1}^{n}f(x_i|\lambda)\Delta \qquad q.e.d. \tag{3.31}$$

### 3.2.5 Problem 16

In this problem one considers thinned Poisson process. Here one has a random number of occurrences N, distributed according to a Poisson distribution with mean $\nu$. Each of the N occurrences, $X_n$, can take on values of 1, with probability p, or 0 with probability $(1-p)$. It is asked to derive the probability distribution for

$$X=\sum_{n=1}^{N}X_n \tag{3.32}$$

Introducing R the number of times $X_n$ takes the value 1, X can be rewritten as:

$$X=\sum_{n=1}^{R}1+\sum_{n=R+1}^{N}0=R \tag{3.33}$$

Therefore the probability distribution for X with given N can be written like this:

$$P(X|N,p)=P(R|N,p) \tag{3.34}$$

P(X|N,p) is a binomial distribution. In the upcoming part one drops the "p" in the notation since it will always be given. N is poisson distributed. Now sum over N from 0 to infinity where N = X+L, L are the unsuccessful trials.

$$P(X) = \sum_{N=0}^{\infty} P(X|N)P(N|\nu) \tag{3.35}$$

P(X) while be 0 if N < X. Therefore

$$P(X) = \sum_{N=X}^{\infty} P(X|X+L)P(X+L|\nu) \tag{3.36}$$

$$\Rightarrow P(X) = \sum_{L=0}^{\infty} P(X|X+L)P(X+L|\nu) \tag{3.37}$$

$$\tag{3.38}$$

Now insert the distributions:

$$P(X) = \sum_{L=0}^{\infty} \frac{(X+L)!}{X!L!} p^X (1-p)^L \frac{e^{-\nu}\nu^{X+L}}{(X+L)!} \tag{3.39}$$

$$= \sum_{L=0}^{\infty} \frac{(\nu p)^X (\nu(1-p))^L e^{-\nu}}{X!L!} \tag{3.40}$$

$$= \frac{(\nu p)^X}{X!} \sum_{L=0}^{\infty} \frac{(\nu(1-p))^L e^{-\nu}}{L!} \tag{3.41}$$

$$= \frac{(\nu p)^X}{X!} e^{-\nu+p\nu-p\nu} \sum_{L=0}^{\infty} \frac{(\nu(1-p))^L}{L!} \tag{3.42}$$

$$= \frac{(\nu p)^X}{X!} e^{-\nu(1-p)-p\nu} \sum_{L=0}^{\infty} \frac{(\nu(1-p))^L}{L!} \tag{3.43}$$

$$= e^{-p\nu} \frac{(\nu p)^X}{X!} e^{-\nu(1-p)} e^{\nu(1-p)} \tag{3.44}$$

$$= \frac{e^{-p\nu}(\nu p)^X}{X!} \qquad q.e.d. \tag{3.45}$$

# Chapter 4

# Gaussian Probability Distribution Function

## 4.1 SUMMARY

In this chapter the Gauss probability distribution is in the centre of attention. It is the limiting distribution of the Poisson and Binomial distribution for a large number of events. It is commonly used because of the CLT theorem that states in its most general form that the proper normalized sum of independent random variables converges towards a normal distribution, even if the original variables were not Gaussian distributed. Furthermore the Gauss distribution appears in many physical processes and many results and methods of the Gaussian distribution can be derived analytically. Mathematically the Gauss Distribution is defined as follows:

$$G(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4.1}$$

where $\mu$ is the expectation value (also the mode and median) and $\sigma$ the standard deviation. The following exercises take a closer look at the Central Limit Theorem, analytical methods possible with the Gauss distribution as well as multivariate Gauss distributions.

## 4.2 EXERCISES

### 4.2.1 Problem 8 (a,b)

In this problem one looks at two probability distributions. In the first one the distribution is given as

$$p(x) = \lambda e^{-\lambda x} \tag{4.2}$$

the Central Limit Theorem (CLT) can be applied. First one wants to estimate what to expect for the mean of a Gauss distribution with n samples from the above probability distribution. As usual first calculate the mean for one measurement:

$$E[x] = \mu = \int_0^\infty p(x) x dx = \lambda \int_0^\infty x e^{-\lambda x} dx = \frac{1}{\lambda} \tag{4.3}$$

(one restricts x to be bigger then 0 because of the simulation below). Next the variance:

$$\sigma^2 = Var[x] = E[x^2] - E[x]^2 = \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2} \tag{4.4}$$

The expectation value and variance of n measurements is:

$$E[\bar{x}] = \frac{1}{n}\sum_{i=1}^{n} E[x_i] = E[x] \qquad Var[\bar{X}] = Var[x] \tag{4.5}$$

where $\bar{x}$ - the average of n measurements is given by:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.6}$$

The CLT states that for $n \to \infty$ one expects a normal distribution with $N(\mu, \frac{\sigma^2}{\sqrt{n}})$. To control the previous result one can do a simulation. Therefore one produces random numbers for x with:

$$x = -\frac{ln(U)}{\lambda} \tag{4.7}$$

where U is a random number between zero and one.

For each experiment the script A.10 does either n = 4, 10 or 100 measurements. 100000 experiments are done. The distribution of the average of each measurement ($\bar{x}$) can be found in fig 4.1.

In the second part of this exercise one takes a look at a distribution for which the CLT can

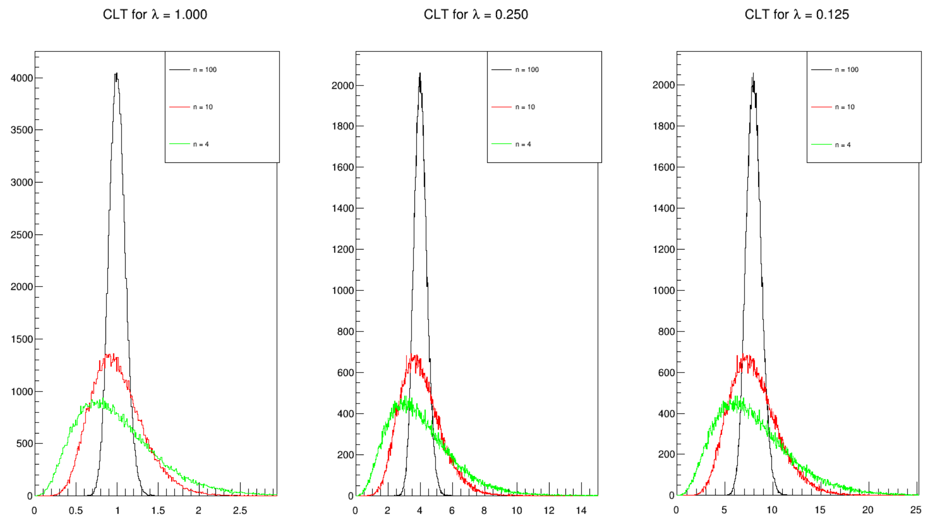


**Figure 4.1:** Distribution of $\bar{x}$ for different $\lambda$. The curves can be fitted with $N(\mu, \frac{\sigma^2}{\sqrt{n}})$

not be applied, namely the Cauchy distribution:

$$f(x) = \frac{1}{\pi\gamma} \frac{\gamma^2}{(x - x_0)^2 + \gamma^2} \tag{4.8}$$

Now one tries to find the expectation value:

$$E[x] = \mu = \int_0^\infty f(x) x dx \tag{4.9}$$

Simplify the integral by choosing a convenient value for $x_0 = 0$:

$$E[x] = \frac{1}{\pi} \int_0^\infty \frac{\gamma^2 x}{x^2 + \gamma^2} \tag{4.10}$$

This integral does not converge. Therefore no expectation value exist and so most likely, the CLT will not work. One can check again with a simulation. This time x will be a random numbers from the Cauchy distribution:

$$x = \gamma tan(\pi U - \pi/2) + x_0 \tag{4.11}$$

Where U is like before a random number between zero and one from an uniform distribution. For the parameters one chooses $x_0 = 25$ and $\gamma = 3$. The script A.11 produces a distribution for x (see fig. 4.2) and a distribution of the mean of x for 100 samples of x (see fig. 4.3). As it can

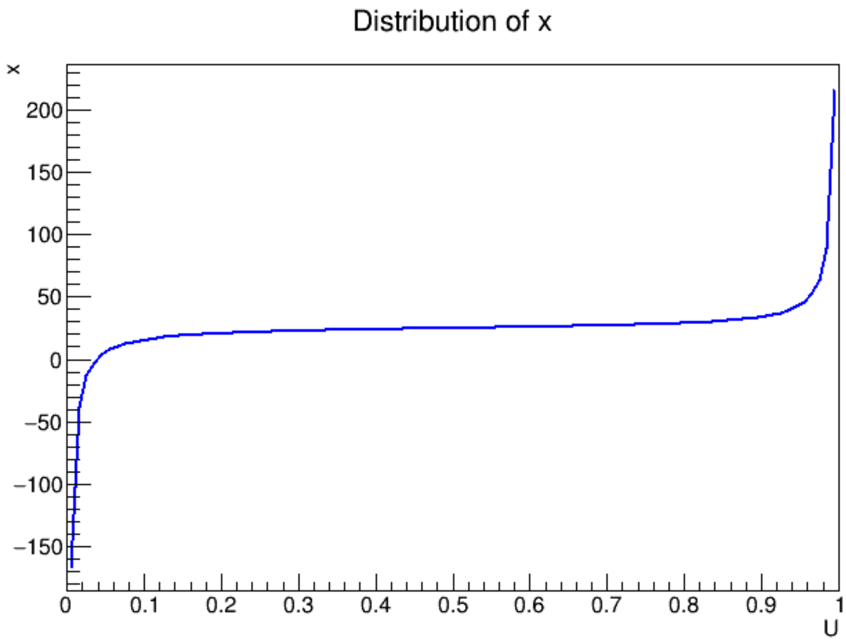


**Figure 4.2:** Distribution of x for $x_0 = 25$ and $\gamma = 3$.

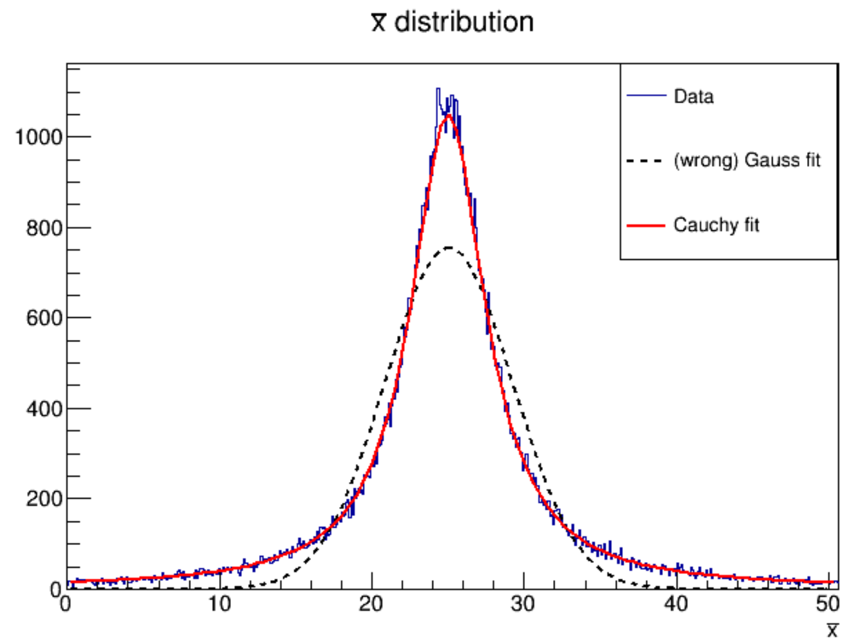


**Figure 4.3:** Distribution of $\bar{x}$ for $x_0 = 25$ and $\gamma = 3$. Data fitted with a Cauchy function (red) and a wrong Gauss fit (black)

be seen in figure 4.3 one can not apply the Central Limit Theorem here. Instead the predicted

Gaussian distribution, one yields again a Cauchy distribution for $\bar{x}$, with the mode $x_0$.

### 4.2.2 Problem 11

In this task one is asked to plot some contours of the bivariate Gauss function given by:

$$P(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left(\frac{-1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right)\right) \tag{4.12}$$

(see next exercise for more detail) for the following parameters: In figure 4.4 one finds the

| figure | $\mu_x$ | $\mu_y$ | $\sigma_x$ | $\sigma_y$ | $\rho$ |
|---|---|---|---|---|---|
| 4.4a | 0 | 0 | 1 | 1 | 0 |
| 4.4b | 1 | 2 | 1 | 1 | 0.7 |
| 4.4c | 1 | -2 | 1 | 2 | -0.7 |

plots to the parameters from the table above.

### 4.2.3 Problem 12

In this problem one takes a closer look on the bivariate Gauss function from the previous exercise. In the first part it will be shown that the PDF can be written like:

$$P(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left(\frac{-1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right)\right) \tag{4.13}$$

Start with the definition of the multivariate normal distribution:

$$P(x_1,\ldots,x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \tag{4.14}$$

with the covariance matrix:

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \\ \vdots & \vdots & \ddots & \vdots \\ \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix} \tag{4.15}$$

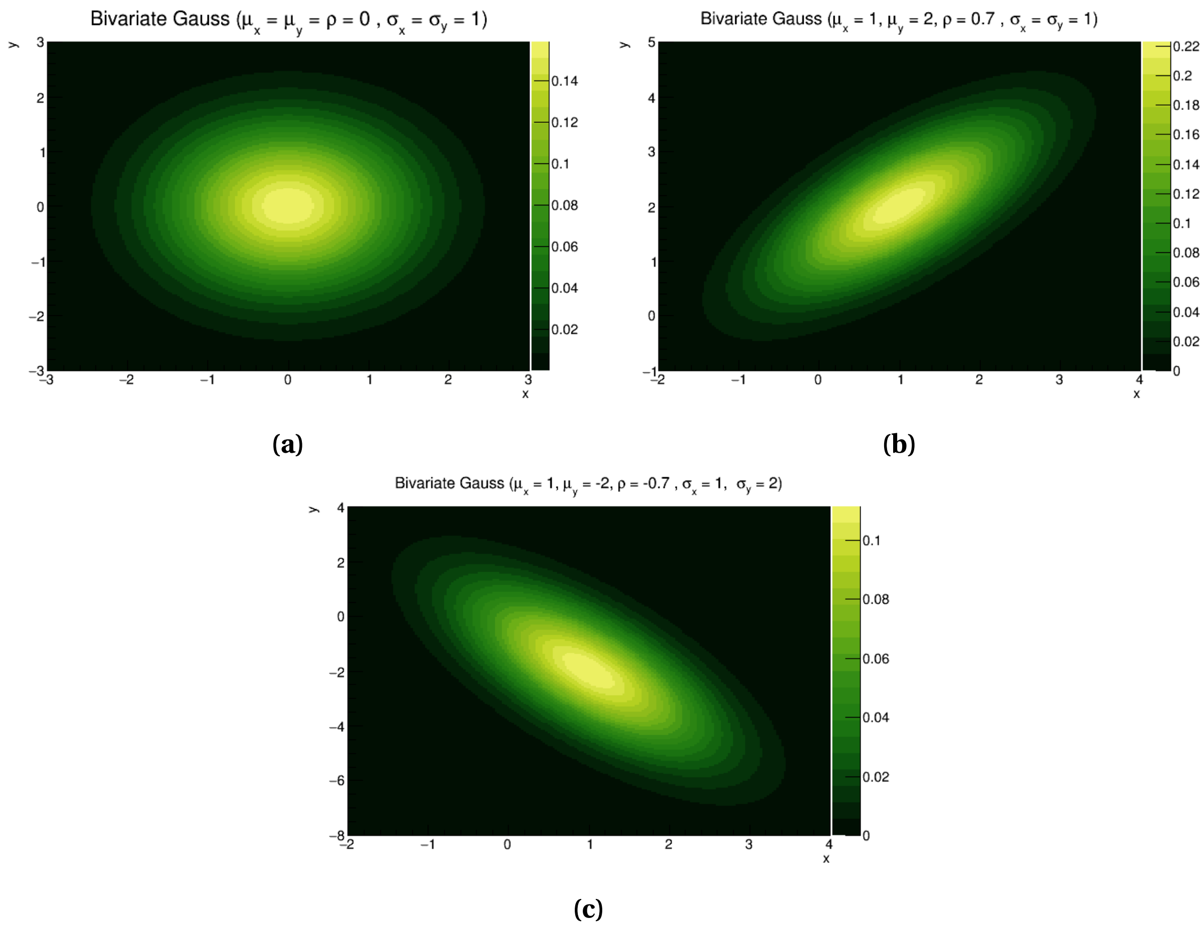


**Figure 4.4:** Contour plots of the bivariate Gauss function for different parameters

and k the number of variables. Now set k to two (bivariate case) and use the relation

$$\sigma^2 = \mathrm{var}(X) = \mathrm{E}[(X - \mathrm{E}(X))^2] = \mathrm{E}[(X - \mathrm{E}(X)) \cdot (X - \mathrm{E}(X))] \tag{4.16}$$

the covariance matrix, $\boldsymbol{\mu}$ and $\boldsymbol{x}$ are collapsing to:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_X^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix} \tag{4.17}$$

Therefore the matrix calculations get quiet simple:

$$|\boldsymbol{\Sigma}| = \sigma_x^2\sigma_y^2(1 - \rho^2) \tag{4.18}$$

and

$$\mathbf{\Sigma}^{-1} = \frac{1}{|\mathbf{\Sigma}|}\begin{pmatrix} \sigma_y^2 & -\rho\sigma_x\sigma_y \\ -\rho\sigma_x\sigma_y & \sigma_x^2 \end{pmatrix} \tag{4.19}$$

Plugging everything into the multivariate distribution, one yields the bivariate gauss distribution (if the mean for x and y is zero) given in this problem.

In the second part of this exercise it will be shown that the bivariate Gauss will result in a Gaussian distribution of $z = x - y$. Where the parameters for this Gauss are given by:

$$\mu_z = \mu_x - \mu_y \qquad \sigma_z^2 = \sigma_x^2 + \sigma_y^2 - 2\rho\sigma_x\sigma_Y \tag{4.20}$$

For that one first substitutes $y = x - z$:

$$P(x,y) = P(x, z-x) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \tag{4.21}$$

$$\cdot\, exp\left(\frac{-1}{2(1-\rho^2)}\left(\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{((x-\mu_x)-(z-\mu_z))^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)((x-\mu_x)-(z-\mu_z))}{\sigma_x\sigma_y}\right)\right) \tag{4.22}$$

Now one marginalizes x:

$$P(z) = \int P(x, x-z)\,dx \tag{4.23}$$

$$= \cdots \int exp\left(\frac{-1}{2(1-\rho^2)}\left(\frac{\tilde{x}^2}{\sigma_x^2} + \frac{(\tilde{x}-\tilde{z})^2}{\sigma_y^2} - \frac{2\rho\tilde{x}(\tilde{x}-\tilde{z})}{\sigma_x\sigma_y}\right)\right)d\tilde{x} \tag{4.24}$$

$$= \cdots e^{\frac{-\tilde{z}^2}{2\sigma_y^2(1-\rho^2)}} \int exp\left(\left(\tilde{x}^2\left(\frac{1}{\sigma_x^2} + \frac{1}{\sigma_y^2} - \frac{2\rho}{\sigma_x\sigma_y}\right) + \tilde{x}\tilde{z}\left(\frac{2\rho}{\sigma_y\sigma_x} - \frac{2}{\sigma_y^2}\right)\right)\frac{-1}{2(1-\rho^2)}\right)d\tilde{x} \tag{4.25}$$

$$= \cdots e^{\frac{-\tilde{z}^2}{2\sigma_y^2(1-\rho^2)}} \int exp\left(\left(\tilde{x}^2\left(\frac{\sigma_z^2}{\sigma_x^2\sigma_y^2}\right) + \tilde{x}\tilde{z}\left(\frac{\sigma_y^2-\sigma_x^2-\sigma_z^2}{\sigma_y^2\sigma_x^2}\right)\right)\frac{-1}{2(1-\rho^2)}\right)d\tilde{x} \tag{4.26}$$

$$= \cdots e^{\frac{-\tilde{z}^2}{2\sigma_y^2(1-\rho^2)}} \int exp\left(\frac{-1}{2(1-\rho^2)\sigma_y^2\sigma_x^2}\left(\tilde{x}^2\sigma_z^2 + \tilde{x}\tilde{z}\left(\sigma_y^2-\sigma_x^2-\sigma_z^2\right)\right)\right)d\tilde{x} \tag{4.27}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_z} e^{\frac{\tilde{z}^2}{2\sigma_y^2(1-\rho^2)}} e^{\frac{-\tilde{z}^2(\sigma_x+\rho\sigma_y)^2}{2\sigma_y^2\sigma_z^2(1-\rho^2)}} \tag{4.28}$$

where one writes $\tilde{x} = x - \mu_x$ and $\tilde{z} = z - \mu_z$ to simplify the notation of the integral. The result is already Gaussian. If now $\rho = 0$ then:

$$P(z) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{\frac{-\tilde{z}^2}{2\sigma_z^2}} = \frac{1}{\sqrt{2\pi}\sigma_z} e^{\frac{-(z-\mu_z)^2}{2\sigma_z^2}} \tag{4.29}$$

One yields a true Gaussian in z.

### 4.2.4 Problem 13

In this exercise one takes a closer look onto the convolution of gauss distributions. Suppose the variable x follows a true Gauss distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{\frac{-(x-x_0)^2}{2\sigma_x^2}} \tag{4.30}$$

and the variable y is following a Gauss distribution around x

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{\frac{-(y-x)^2}{2\sigma_y^2}} \tag{4.31}$$

So what is the predicted distribution of y ( g(y) )? P(y|x) is a convolution of f(x) and g(y). In the Fourier space a convolution is a simple multiplication, hence one switches to the Fourier space:

$$\hat{f}(k) = \frac{1}{\sqrt{2\pi}} \int f(x) e^{-2\pi i x k} dx = \frac{1}{2\pi} e^{\frac{-\sigma_x^2 k^2}{2} + i x_0 k} \tag{4.32}$$

and

$$\hat{P}(k|l) = \frac{1}{2\pi} \int \int P(x|y) e^{-2\pi i (xk + yl)} dx dy = \frac{1}{(2\pi)^{3/2}} e^{-\frac{k^2}{2}\sigma_y^2} \delta(k+l) \tag{4.33}$$

Thus:

$$\hat{g}(l) = \frac{\hat{P}(k|y)}{\hat{f}(k)} = \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{k^2}{2}\sigma_y^2} \delta(k+l)}{e^{\frac{-\sigma_x^2 k^2}{2} + i x_0 k}} \tag{4.34}$$

This is only none zero for k = -l. Therefore:

$$\hat{g}(l) = \frac{1}{\sqrt{2\pi}} e^{-\frac{l^2}{2}(\sigma_y^2 - \sigma_x^2) + i x_0 l} \tag{4.35}$$

Now one just has to transform back to the y space:

$$g(y) = \int \hat{g}(l) e^{2\pi i y l} dl = \frac{1}{\sqrt{2\pi(\sigma_y^2 - \sigma_x^2)}} e^{\frac{-(y-x_0)^2}{2(\sigma_y^2 - \sigma_x^2)}} \tag{4.36}$$

One yields a Gaussian distribution for y. One can conclude; a convolution of two Gaussian distributions yields a Gaussian distribution.

### 4.2.5 Problem 14

In this exercise one takes a look at some measurements of nuclear cross sections. The measurement results can be found in table 4.1. A model with the following form is assumed:

| $\theta[°]$ | 30 | 45 | 90 | 120 | 150 |
|---|---|---|---|---|---|
| cross section | 11 | 13 | 17 | 17 | 14 |
| error | 1.5 | 1.0 | 2.0 | 2.0 | 1.5 |

**Table 4.1:** Data of the nuclear cross section measurements. The units of cross section are $10^-30$ cm2/steradian. Assume the quoted errors correspond to one Gaussian standard deviation

$$\mu(\theta) = A + Bcos(\theta) + Ccos(\theta^2) \tag{4.37}$$

First one is asked to set up a formula for the posterior probability density, assuming flat priors for the parameters A, B, C. As usual start with Bayes Theorem:

$$P_i(\mu(\theta)|x,\theta) = P(\mu(\theta_i)|x_i,\theta_i) = \frac{P_i(x_i,\theta_i|\mu(\theta_i))P_0(\mu(\theta_i))}{\int \cdots d\mu(\theta_i)} \tag{4.38}$$

Since A, B and C have flat priors $\mu(\theta_i)$ has a flat prior too. Hence:

$$P_i(\mu(\theta)|x,\theta) \propto P_i(x_i,\theta_i|\mu(\theta_i)) = G_{\theta_i}(\mu(\theta_i),\sigma_i) \tag{4.39}$$

where x are the given cross sections, $\sigma$ the given errors and $\theta$ the given angles of measurement. Now the total posterior density is given by the product of the ones for each measurement:

$$P = \prod_{i=1}^{5} P_i(\mu(\theta)|x,\theta) = \prod_{i=1}^{5} G_{\theta_i}(\mu(\theta_i),\sigma_i) \tag{4.40}$$

Now one can calculate the values at the mode of the posterior pdf. Therefore one uses the log-pdf since it simplifies the calculations enormously:

$$ln(P) = \sum_{i=1}^{5} ln(G_{\theta_i}(\mu(\theta_i),\sigma_i)) \tag{4.41}$$

To find the mode one has take the derivative of the above formula in A,B,C. This yields the following system of equations which has to be equal to zero:

$$\begin{pmatrix} \frac{dln(P)}{dA} \\ \frac{dln(P)}{dB} \\ \frac{dln(P)}{dC} \end{pmatrix} = \sum_{i=1}^{5} \frac{dln(P)}{d\mu_i} \begin{pmatrix} \frac{d\mu_i}{dA} \\ \frac{d\mu_i}{dB} \\ \frac{d\mu_i}{dC} \end{pmatrix} \stackrel{!}{=} \vec{0} \tag{4.42}$$

with

$$\frac{dln(P)}{d\mu_i} = \frac{ln(\sqrt{2\pi}\sigma_i)}{\sigma_i^2}(\mu_i - x_i) \tag{4.43}$$

and

$$\begin{pmatrix} \frac{d\mu_i}{dA} \\ \frac{d\mu_i}{dB} \\ \frac{d\mu_i}{dC} \end{pmatrix} = \begin{pmatrix} 1 \\ cos(\theta_i) \\ cos(\theta_i^2) \end{pmatrix} \tag{4.44}$$

now one "just" has to solve this linear system of equations. But to make things easier the parameters will be marginalized with numerical methods. The resulting plots can be found in fig. (4.5). The data fitted with the probability band can be found in fig. (4.6). The modes for the three parameters are:

$$A^* = 15.356 \qquad B^* = -1.076 \qquad C^* = -2.567 \tag{4.45}$$

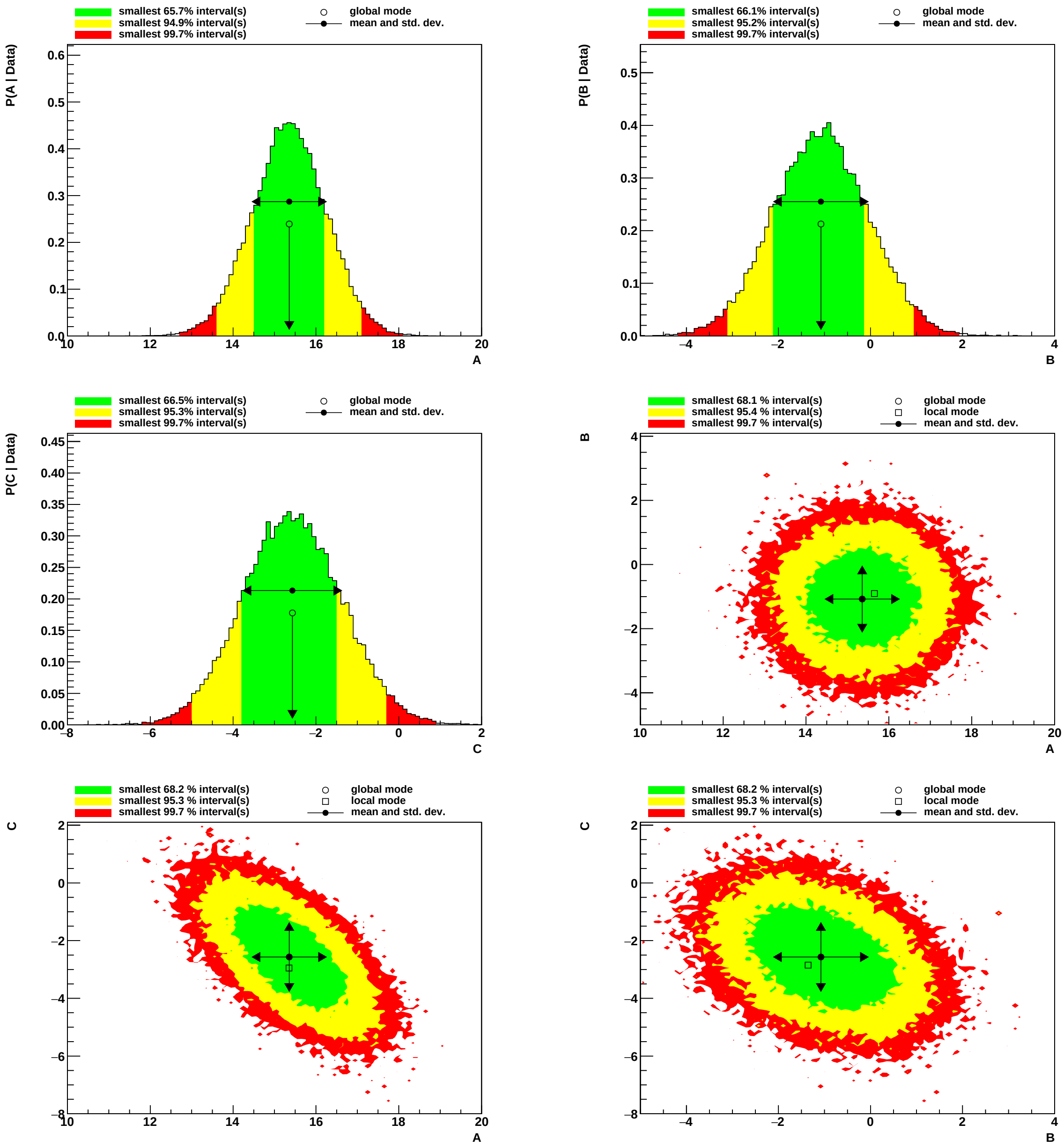


**Figure 4.5:** One and two dimensional marginalized probability distribution for the given model

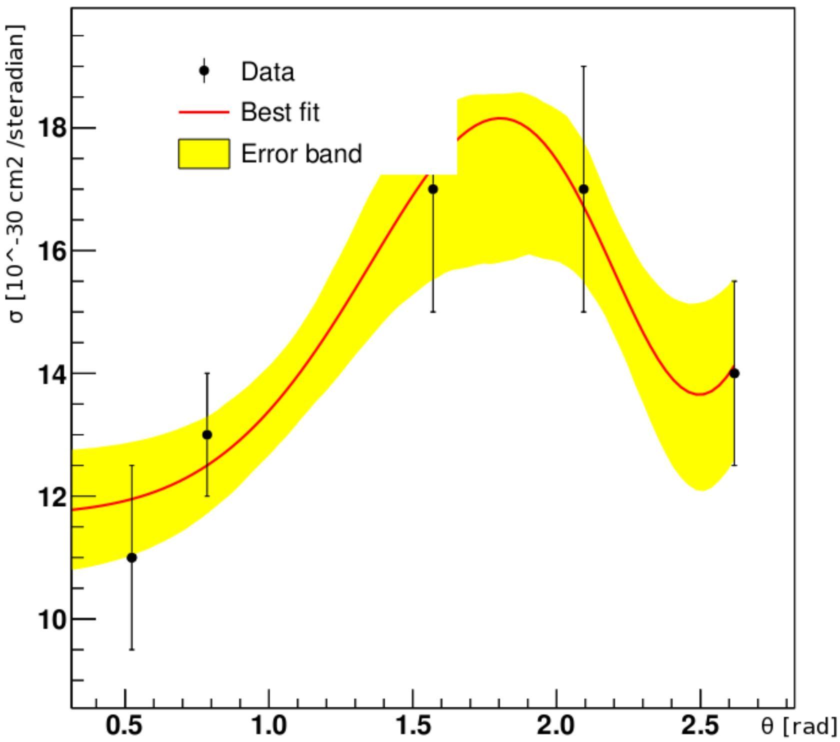


**Figure 4.6:** Given data fitted with the given model with the previous obtained parameters

# Chapter 5

# Model Fitting and Model Selection

## 5.1 Summary

In this chapter one learns how to analyse data with a certain model and extract its parameters. Furthermore one takes a closer look on quantities like the famous $\chi^2$ or the p-value which give information about the goodness of the chosen model. The following exercises take a look at good and bad models, how they describe given data and how the p-value or the $\chi^2$ behave in this two situations. This is done in the Bayesian as well as in the Frequentist interface.

## 5.2 Exercises

### 5.2.1 Problem 1

This exercise provides the data in table 5.1. One is asked to fit this data with a Sigmoid function. This function is defined by:

$$S(E|E_0, A) = \frac{1}{1 + e^{-A(E-E_0)}} \tag{5.1}$$

where $E_0$ is the value where E reaches 50 percent and A is a scaling function. In this exercise E can be identified as the energy and $\epsilon(E|E_0, A) = S(E|E_0, A)$ as the efficiency. First one is asked to find the posterior probability distribution. One starts by defining a prior. From the given data one can see that the efficiency is at around 50 percent for E = 2. Given the functional form of the Sigmoid function one yields:

$$\epsilon(E|E_0, A) = 0.5 \qquad \text{if E} = E_0 \tag{5.2}$$

| Energy ($E_i$) | Trials ($N_i$) | Successes ($r_i$) |
|---|---|---|
| 0.5 | 100 | 0 |
| 1.0 | 100 | 4 |
| 1.5 | 100 | 22 |
| 2.0 | 100 | 55 |
| 2.5 | 100 | 80 |
| 3.0 | 100 | 97 |
| 3.5 | 100 | 99 |
| 4.0 | 100 | 99 |

**Table 5.1:** Data from exercise five

One chooses a Gaussian prior for $E_0$ centred around 2:

$$P_0(E_0) = G(E_0|\mu = 2.0, \sigma = 0.3) \tag{5.3}$$

Furthermore one can assume that the efficiency changes by about 25 percent when one moves away from the centroid by 0.5 units. Therefore:

$$\frac{d\epsilon}{dE} = \frac{Ae^{-A(E-E_0)}}{(1+e^{-A(E-E_0)})^2} = \frac{A}{4} \qquad \text{for E} = E_0 \tag{5.4}$$

Therefore one can estimate A with

$$0.5 \cdot \frac{A}{4} \approx 0.25 \Rightarrow A \approx 2 \tag{5.5}$$

Again one chooses a Gaussian prior for A (centroid on 2):

$$P_0(A) = G(A|\mu = 2.0, \sigma = 0.5) \tag{5.6}$$

So one can give now a probability distribution based one the above defined model:

$$P(r|N, E_0, A) = \prod_{i=1}^{8} \binom{N_i}{r_i} \epsilon(E_i|A, E_0)^{r_i} (1 - \epsilon(E_i|A, E_0))^{N_i - r_i} \tag{5.7}$$

Now with the help of the Bayesian Analysis Toolkit (BAT), which uses Markov Chain Monte Carlo algorithm, the modes of A and $E_0$ are calculated. This takes significantly less computation time then the two dimensional grid space evaluation. The ROOT script from A.12 is used.

One yields the two dimensional mode of

$$(A^*, E_0^*) = (2.86, 1.97) \tag{5.8}$$

In figure the 5.1 probability contour in the (A,$E_0$) parameter space can be found. Now one

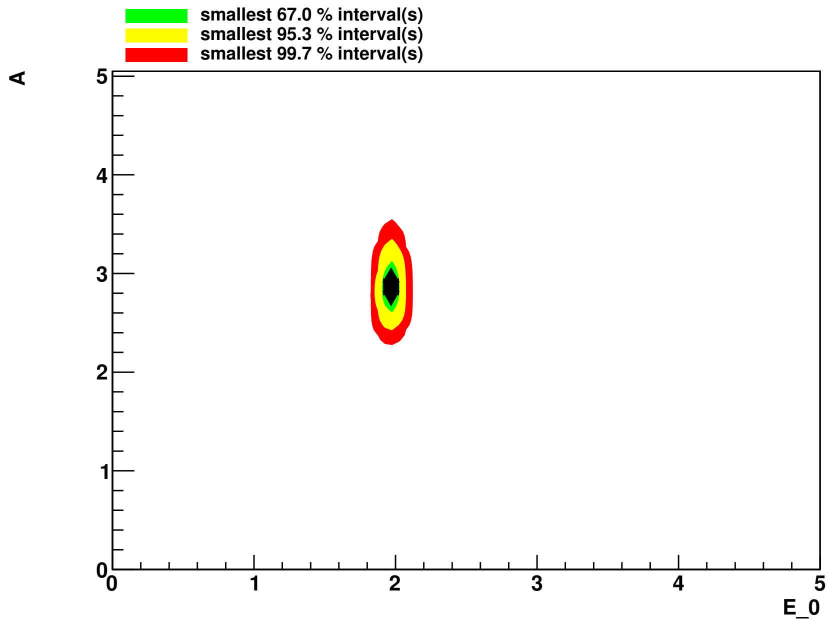


**Figure 5.1:** probability contour in the (A,$E_0$) parameter space

can also draw the data points with the obtained fit function. The result can be seen in figure 5.2.

In the second part of this exercise one is asked to define a suitable test statistic and find the frequentist 68 percent Confidence Level region for (A, $E_0$). The test statistic will drastically reduce the computational power necessary to solve this problem. Since the data is binomial the following test statistic can be used:

$$\gamma(r_i|A, E_0) = \prod_{i=1}^{8} \binom{N_i}{r_i} p_i^{r_i} (1-p_i)^{N_i - r_i} \tag{5.9}$$

So the basic procedure is as follows: First one fixes the value A and $E_0$ and calculate success probability with the sigmoid function from above. After that for each of the eight energies a random success rate is generated. This is based on the previous obtained probability

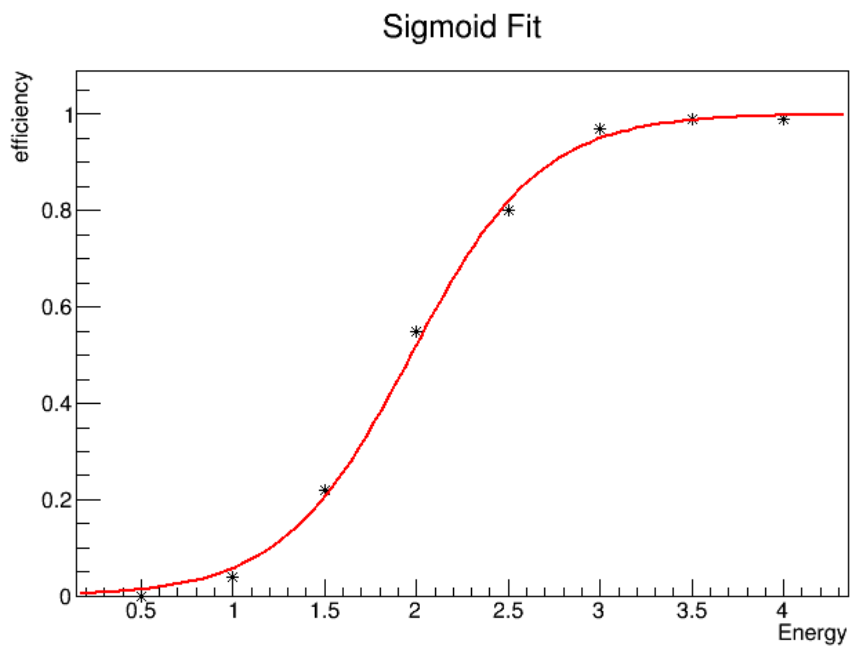


**Figure 5.2:** Given data points fitted with the obtained function

distribution. Then $\gamma$ is calculated. The values are stored in decreasing order. Repeating everything till one yields the desired amount of experimental data. Then the value of $\gamma$ is noted for which 68 percent of the experiments are above this value. Then it is checked if the data is in the accepted range, if it is then the values of A and $E_0$ are in the 68 percent confidence level interval. The script in A.13 uses this instruction. The resulting Confidence Level plot can be found in figure 5.3

### 5.2.2 Problem 2

In this section one should repeat the previous analysis but this time with the following function:

$$\epsilon(E) = sin(A(E - E_0)) \tag{5.10}$$

Starting with the same probability distribution:

$$P(P(r|N, E_0, A) = \prod_{i=1}^{8} \binom{N_i}{r_i} \epsilon(E_i|A, E_0)^{r_i} (1 - \epsilon(E_i|A, E_0))^{N_i - r_i} \tag{5.11}$$

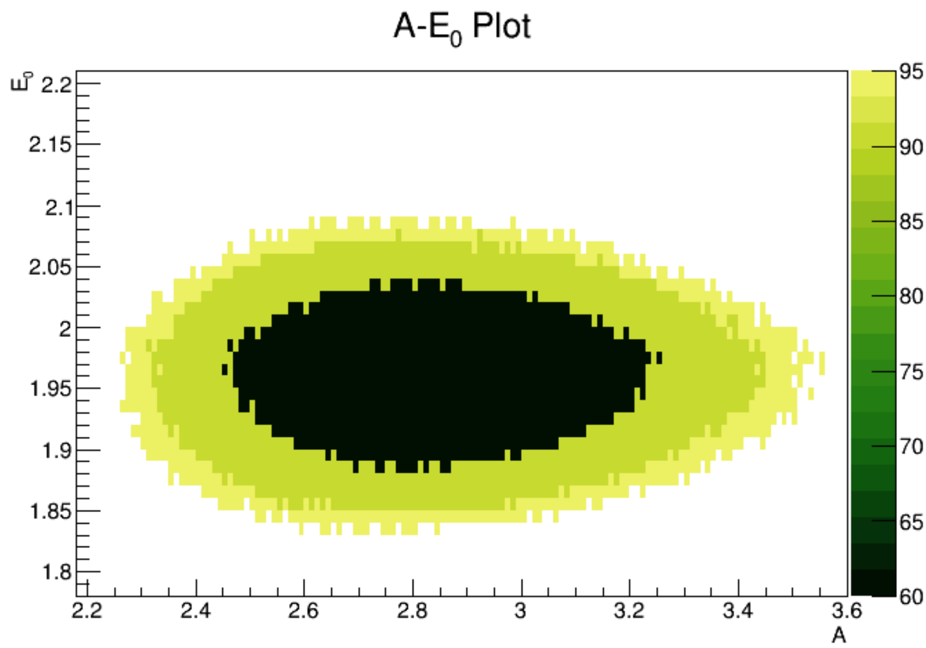


**Figure 5.3:** Confidence Level contour plot in the (A,$E_0$) parameter space. The confidence levels are indicated by the colour.

Again the Bayesian Analysis Tool kit is used to find the two dimensional mode $(A^*, E_0^*)$. The code snippet can be found in A.14. The analysis yields the following modes:

$$(A^*, E_0^*) = (0.5, 8) \tag{5.12}$$

In figure the 5.4 probability contour in the (A,$E_=0$) parameter space can be found. Now one can also draw the data points with the obtained fit function. The result can be seen in figure 5.5.

While the sinus functions seems to fit the given data (more or less), extending the range of the x-axis (fig. 5.6) shows that this function does not provide a physical model (efficiency is oscillating and even smaller then zero)

### 5.2.3 Problem 3

In this problem one is asked to derive the mean, the mode and the variance for the $\chi^2$ distribution for one data point. The $\chi^2$ distribution is one of the best known test statistics. It

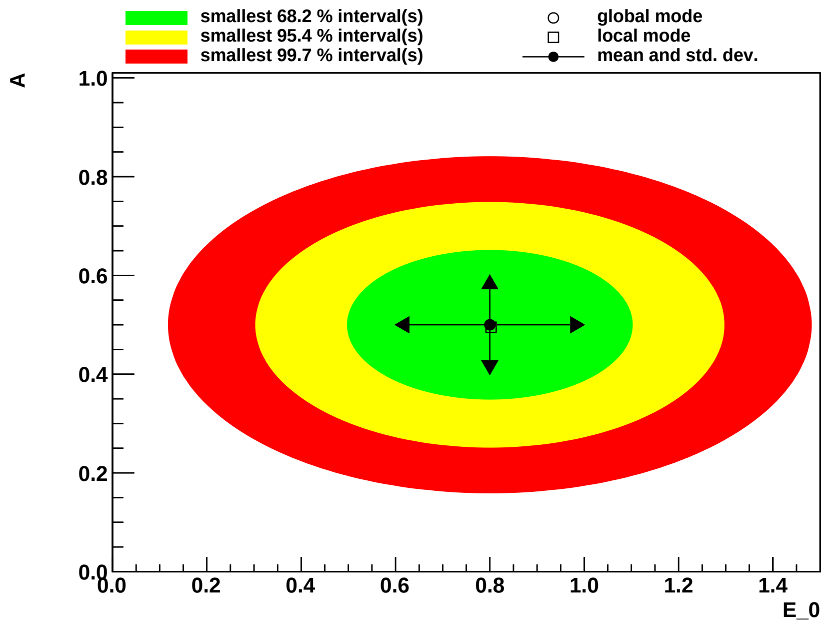


**Figure 5.4:** probability contour in the (A,$E_0$) parameter space

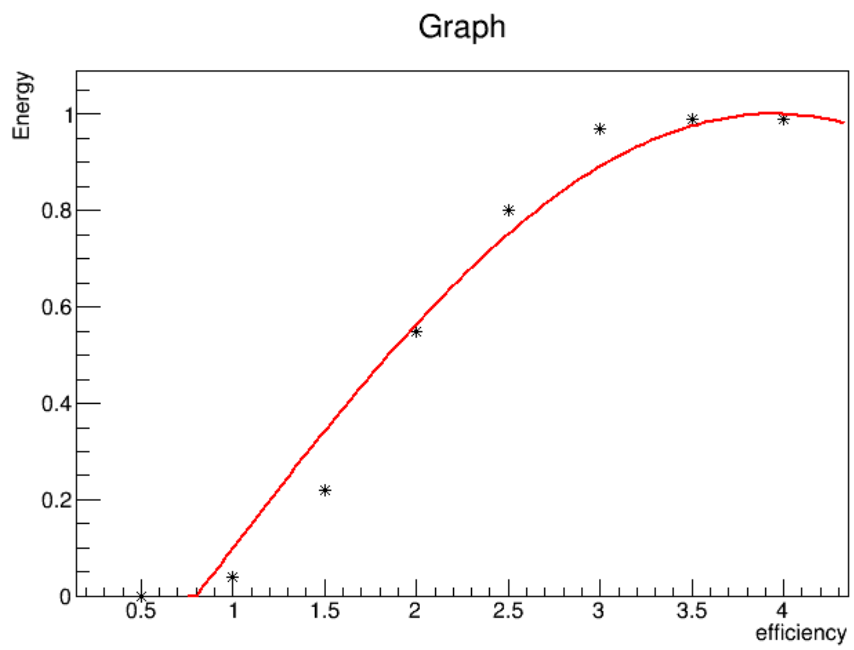


**Figure 5.5:** Given data points fitted with the obtained sinus function

is the weighted sum of the squared distances between the model prediction and the data:

$$\chi^2 = \sum_{i=1} \frac{(y_i - f(x_i|\lambda))^2}{\omega_i^2} \tag{5.13}$$

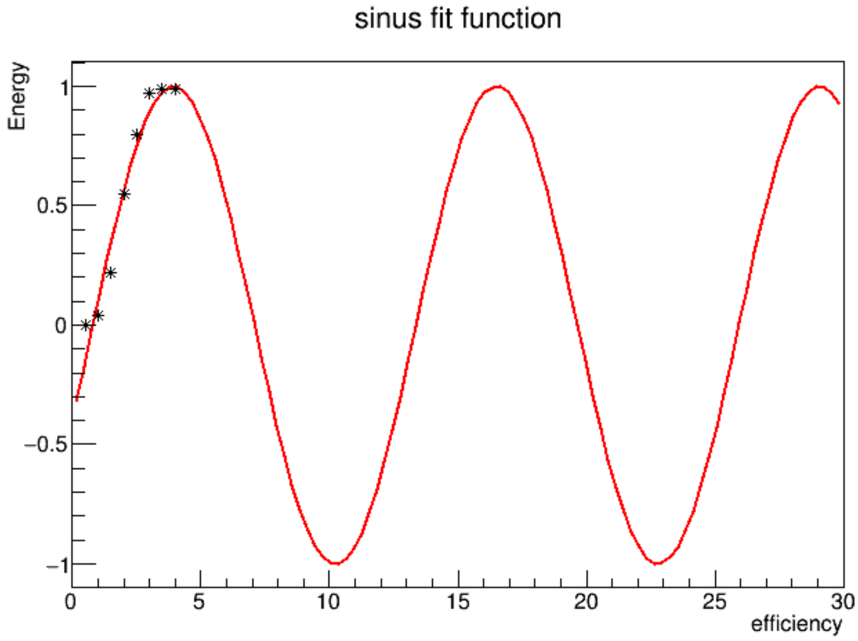


**Figure 5.6:** Given data points fitted with the obtained sinus function for a large energy range shows none physical behaviour

Here f is the model predicting the data at x given parameter value $\lambda$, y is the measured value and $\omega$ is the weight for the data.

Assume that the measurement follows a gauss distribution:

$$P(y) = G(y|f(x|\lambda), \sigma(x|\lambda)) \tag{5.14}$$

Define $\omega_i = \sigma_i = \sigma(x_i|lambda)$. Now work - as it is asked in the problem - with a single measurement:

$$P(\chi^2)\frac{d\chi^2}{dy} = 2P(y) \qquad y \geq f(x|\lambda) \tag{5.15}$$

The factor two is here to take both the negative and positive values into account.

$$\Rightarrow \frac{d\chi^2}{dy} = \frac{2(y - f(x|\lambda)}{\sigma^2} \tag{5.16}$$

$$\left|\frac{d\chi^2}{dy}\right| = \frac{2\sqrt{\chi^2}}{\sigma} \tag{5.17}$$

$$P(\chi^2) = \frac{1}{\sqrt{2\pi\chi^2}}e^{-\chi^2/2} \tag{5.18}$$

This result does not depend on any function parameters. Now one can calculate the mean:

$$E[\chi^2] = \int_0^\infty \chi^2 P(\chi^2) d\chi^2 = \int_0^\infty \sqrt{\frac{\chi^2}{2\pi}} e^{-\chi^2/2} d\chi^2 = 1 \tag{5.19}$$

and the variance:

$$var(\chi^2) = E[\chi^4] - (E[\chi^2])^2 = \int_0^\infty \chi^4 P(\chi^2) d\chi^2 - 1 = \int_0^\infty \sqrt{\frac{\chi^6}{2\pi}} e^{-\chi^2/2} d\chi^2 - 1 = 3 - 1 = 2 \tag{5.20}$$

The mode ($\chi^{2*}$) is defined like this:

$$\max_{\chi^2>0} P(\chi^2) = P(\chi^{2*}) \tag{5.21}$$

But since $P(\chi^2)$ is is decreasing on the positives (see fig. 5.7) the mode is 0.

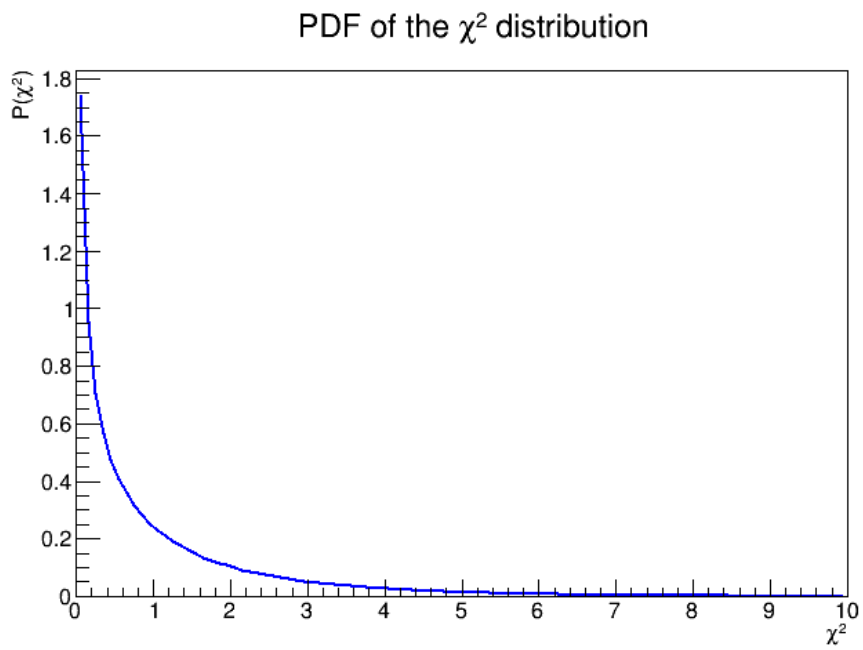


**Figure 5.7:** PDF of the $\chi^2$ distribution

### 5.2.4 Problem 8

In this exercise the data in table 5.2 is given. It is assumed that all the data has the same uncertainty of $\sigma = 4$. Furthermore one can assume that the data can be modelled using a

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| x | 0.10 | 0.15 | 0.20 | 0.25 | 0.30 | 0.35 | 0.40 | 0.45 | 0.50 |
| y | 11.3 | 19.9 | 24.9 | 31.1 | 37.2 | 36.0 | 59.1 | 77.2 | 96.0 |
| x | 0.55 | 0.60 | 0.65 | 0.70 | 0.75 | 0.80 | 0.85 | 0.90 | 0.95 |
| y | 90.3 | 72.2 | 89.9 | 91.0 | 102.0 | 109.7 | 116.0 | 126.6 | 139.8 |

**Table 5.2:** Data from exercise eight

Gauss probability distribution.

In the first part one is asked to use this model:

$$f(x|A,B,C) = A + Bx + Cx^2 \tag{5.22}$$

A quadratic model which is only representing background. In the second part the data will be fitted with a quadratic + Breit-Wigner model representing background and signal:

$$f(x|A,B,C,x_0,\Gamma) = A + Bx + Cx^2 + \frac{D}{(x-x_0)^2+\Gamma^2} \tag{5.23}$$

The data fitted with both functions can be found in figure 5.8. The fit is done in ROOT using $\chi^2$ minimization. One can find the parameter values which minimize the $\chi^2$ respectively

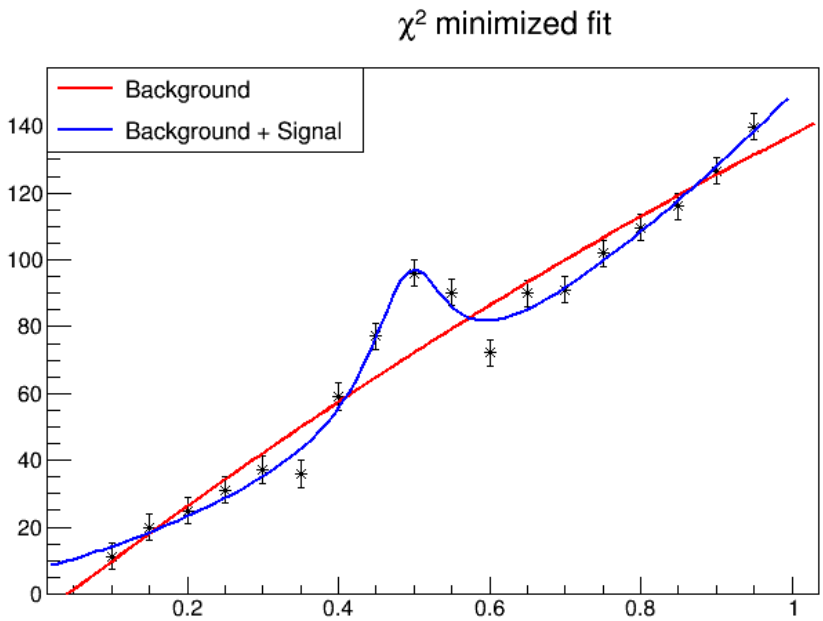


**Figure 5.8:** $\chi^2$ minimized fits for the given data. Fit parameters see table 5.3

maximize the posterior probability distribution since

$$P(\lambda|\vec{D}) \propto e^{-\chi^2(\vec{\lambda})} \tag{5.24}$$

by using the covariance matrix (M):

$$\vec{Y} = M\vec{\lambda} \tag{5.25}$$

Where

$$\vec{Y} = \sum_{i=1}^{n} \frac{C_{ik} y_i}{\sigma_i} \tag{5.26}$$

with

$$C_i k = \frac{\partial f(x_i|\vec{\lambda})}{\partial \lambda_k} \tag{5.27}$$

The covariance matrix for the fits in figure 5.8 are:

$$M_b = \begin{bmatrix} 94.2313 & -376.9 & 318.22 \\ -376.9 & 1765.48 & -1603.83 \\ 318.22 & -1603.83 & 1527.46 \end{bmatrix}$$

for the fit with just the background and for the fit with background + signal the matrix is:

$$M_{b+s} = \begin{bmatrix} 25.26 & -131.4 & 118.1 & 0.1346 & 0.0043 & -0.0019 \\ -131.4 & 872.3 & -829.3 & -1.377 & -0.014 & 0.2122 \\ 118.1 & -829.3 & 804.6 & 1.354 & 0.0078 & -0.2088 \\ 0.1346 & -1.377 & 1.354 & 0.0048 & 3.49 \cdot 10^{-5} & -0.0008546 \\ 0.0043 & -0.01412 & 0.0078 & 3.49 \cdot 10^{-5} & 5.99 \cdot 10^{-5} & -7.71 \cdot 10^{-6} \\ -0.019 & 0.2122 & -0.2088 & -0.00085 & -7.71 \cdot 10^{-6} & 0.0001673 \end{bmatrix}$$

By inverting this matrices one can obtain the fit parameters:

$$\vec{\lambda} = M^{-1}\vec{Y} \tag{5.28}$$

The numerically obtained fit parameters for the fits in fig. 5.8 can be found in table 5.3

Finally one calculates the p-value for the fits. with the help of the p-value one can determine if the used model yields an adequate explanation of the data. In the case of the $\chi^2$

| background | | background + signal | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| $\chi^2$ | 1470.6 | $\chi^2$ | 816.8 |
| A | $-7.33 \pm 9.68$ | A | $6.71 \pm 5.03$ |
| B | $173.82 \pm 41.88$ | B | $56.79 \pm 29.53$ |
| C | $-29.20 \pm 38.96$ | C | $-85.51 \pm 28.37$ |
| | | D | $0.16 \pm 0.07$ |
| | | $x_0$ | $0.49 \pm 0.01$ |
| | | $\Gamma$ | $-0.06 \pm 0.01$ |

**Table 5.3:** Fit-parameters for fig. 5.8

distribution the p-value can be defined as follows:

$$p = F(\chi^2) = \int_{\chi^2}^{\infty} P(\chi^2|n)d\chi^2 = 1 - \frac{\gamma(\frac{n}{2}, \frac{\chi^2}{2})}{\Gamma(\frac{n}{2})} \tag{5.29}$$

Where n is the number of data points and $\gamma$ is the lower-incomplete gamma function defined as:

$$\gamma(s,t) = \sum_{i=0}^{\infty} \frac{t^s e^{-t} t^i}{s(s+1)\cdots(s+1)} \tag{5.30}$$

for the two fits in fig. 5.8 one yields a p-value of p= $9.5796 \cdot 10^{-307}$ for the background fit and p = $1.7593 \cdot 10^{-44}$ for the background+signal fit.

Summarizing: Seeing the fits in the plot and the very low p value one could conclude that both fit models are quiet bad for this data. At least the model with background + signal seems to be a little better (p-value "closer" to one).

For the second part the analysis will be repeated but this time by performing an Bayesian fit. In the Bayesian analysis the $\chi^2$ distribution is not needed if the weights $(1/\omega_i^2)$ in the definition of $\chi^2$ are independent of the parameters and the model depends only linearly on the parameters. Then an an analytical form can be derived:

$$f(x_i|\lambda) = \sum_{k=1}^{K} a_k \lambda_k \tag{5.31}$$

And with that special model:

$$\chi^2 = \sum_i \frac{\left(y_i - \sum_{k=1}^{K} a_k \lambda_k\right)^2}{\omega_i^2} \tag{5.32}$$

The BAT script in A.15 uses the above equation to fit the given data. For the quadratic fit (background) the fitted data can be found in plot 5.9 and in figure 5.10 the one and two dimensional marginalized probability distribution for the fit parameters (A,B,C).

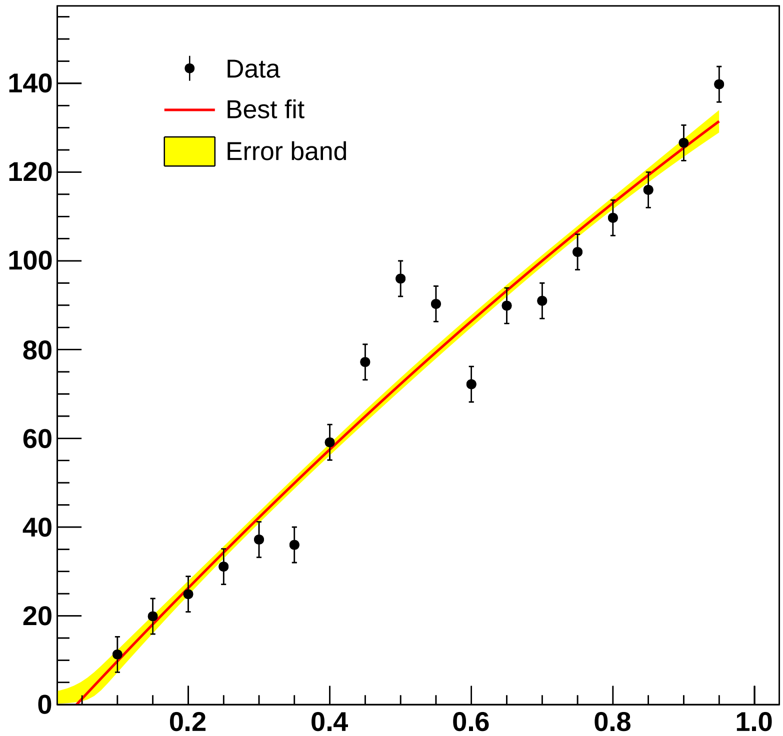


**Figure 5.9:** Bayesian fit for the given data. The intervals for the parameters are: A: $-7.282^{+3.92}_{-3.91}$, B:$173.51^{+6.872}_{-16.869}$, C:$-28.91^{+15.68}_{-15.69}$ (Median + 68 percent central interval)

The same can be repeated for the quadratic+Breit-Wigner fit representing the background and the signal. The fitted data can be found in figure 5.11. The one dimensional marginalized probability distributions can be found in figure 5.12.

Furthermore the Bayes-factor (K) can be stated. It compares two models with each other, to help with the model selection. It is defined as follows:

$$K = \frac{P(D|M_1)}{P(D|M_2)} \tag{5.33}$$

Where D is the given data and M the model.Calculating Bayes-Factor of the background model and the background + signal model with which one can quantitative state if the given

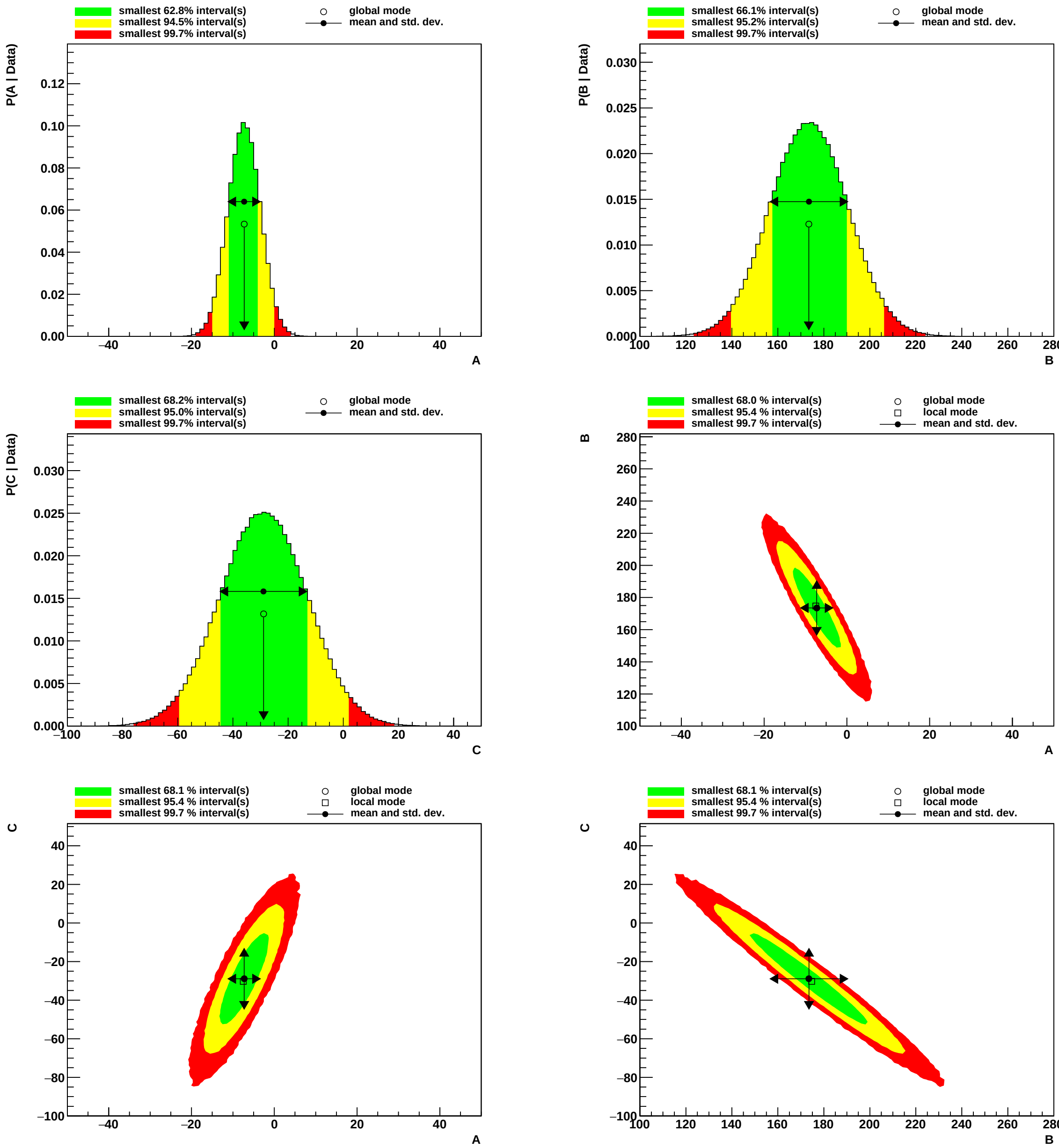


**Figure 5.10:** One and two dimensional marginalized probability distribution for the quadratic fit function

data contains a signal or not.

$$K = \frac{P(D|background + signal)}{P(D|background)} = \frac{e^{-57.402}}{e^{-102.556}} = e^{45.154} \approx 4 \cdot 10^{19} \tag{5.34}$$

It indicates very strongly that the given data indeed contains a signal.

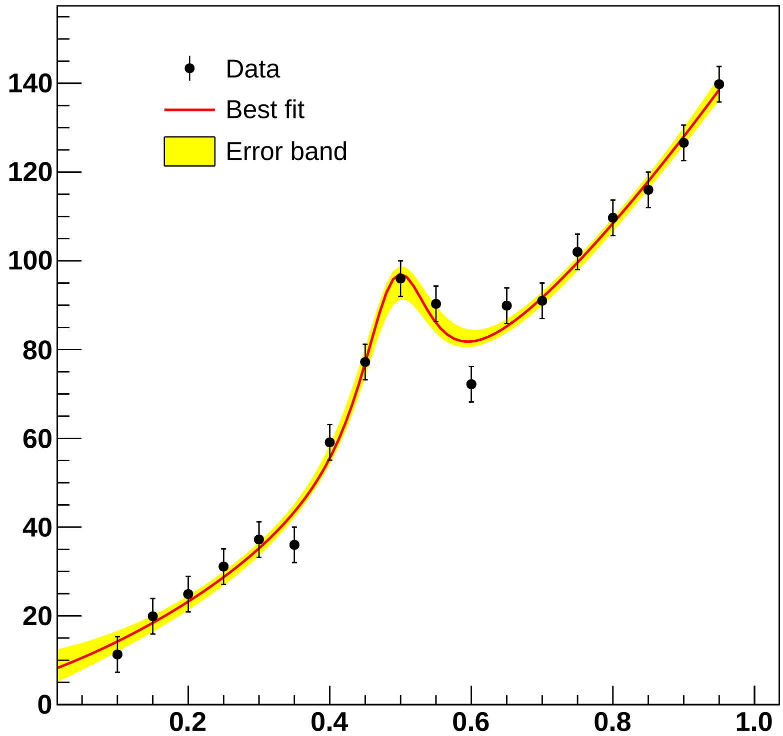


**Figure 5.11:** Bayesian fit for the given data. The intervals for the parameters are: A: $6.993^{+4.014}_{-3.736}$, B:$49.73^{+23}_{-24.67}$, C:$92.85^{+23.81}_{-22.71}$, D:$0.194^{+0.097}_{-0.064}$, $x_0 : 0.493^{+0.007}_{-0.007}$, $\Gamma : 0.069^{+0.015}_{-0.012}$ ( Median + 68 percent central interval)

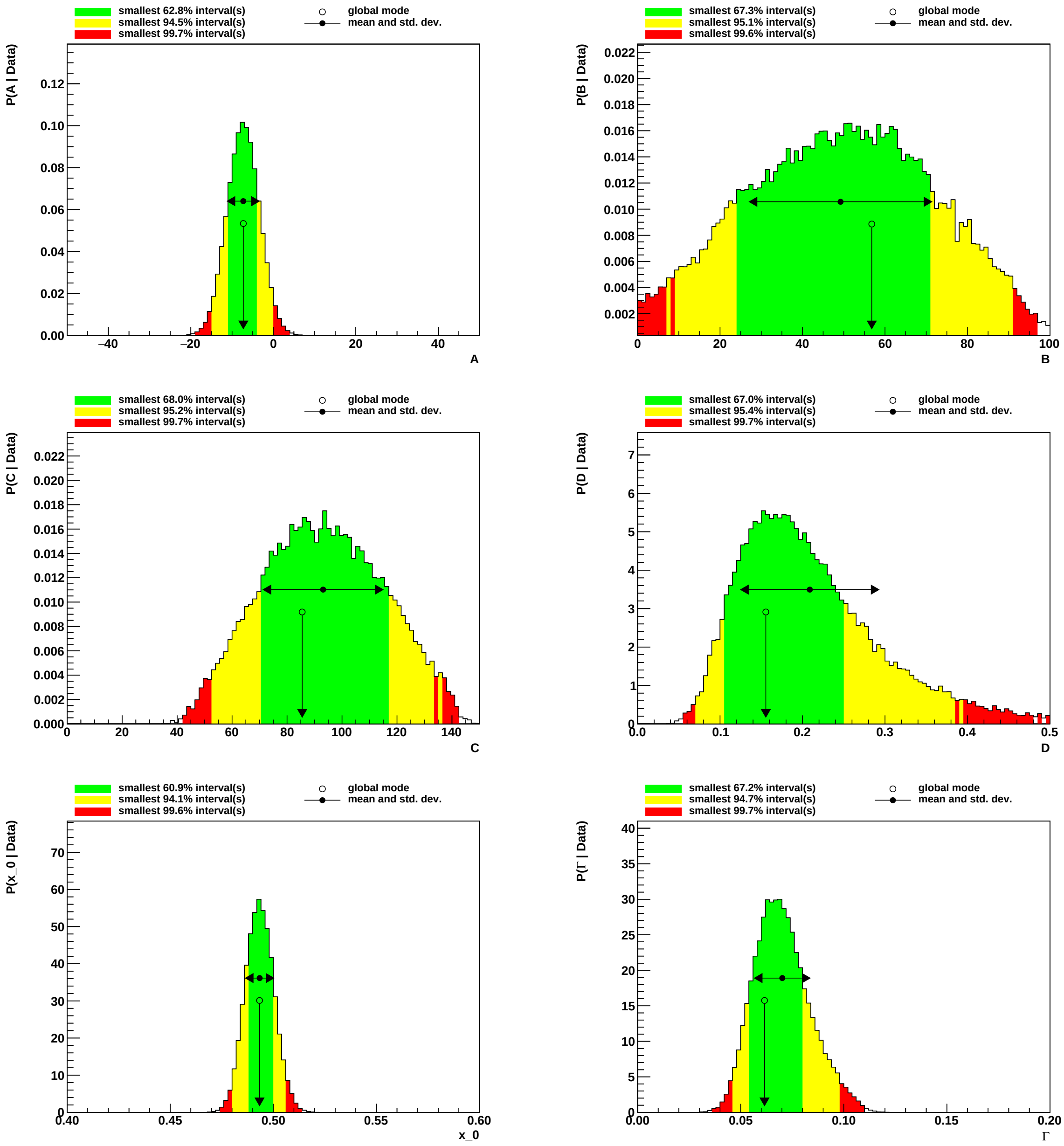


**Figure 5.12:** One dimensional marginalized probability distribution for the quadratic + B.-W. fit function

# Chapter 6

# Maximum Likelihood estimator

## 6.1 Summary

The last chapter of this report deals with the maximum likelihood estimator (MLE). This method is used to estimate the parameters of a statistical model given data. The next two exercises apply this method in different situations by first defining the likelihood of the given model and maximizing it in the parameters an estimator is asked for. In the Bayesian interface the MLE is a special case of maximum posterior estimation, assuming uniform prior distribution. In the Frequentist school the MLE is one of several ways to get the estimate of a parameter without using prior distributions.

## 6.2 Exercises

### 6.2.1 Problem 1

In this problem one works with a Bernoulli distribution:

$$P(x|p) = p^x(1-p)^{1-x} \tag{6.1}$$

In the first part one is asked to calculate the Fischer Information (I). It is a way of measuring the amount of information that (in case of the Bernoulli distribution) the variable x carries about the parameter p. It is defined as follows:

$$I(p) = -E\left(\frac{\partial^2 ln(P(x|p)}{\partial p^2}\right) \tag{6.2}$$

First take the ln of the Bernoulli distribution:

$$ln(P(x|p) = ln(p^x(1-p)^{1-x}) = xln(p) + (1-x)ln(1-p) \tag{6.3}$$

Now one can easily calculate the Fisher-Information:

$$I(p) = \frac{\partial^2}{\partial p^2} ln(P(x|p)) = -E\left(\frac{\partial^2}{\partial p^2} ln(p) + (1-x)\frac{\partial^2}{\partial p^2} ln(1-p)\right) = E\left(\frac{x}{p^2} + \frac{1-x}{(1-p)^2}\right) \tag{6.4}$$

Furthermore one is asked to find the maximum likelihood estimator (MLE) for p. In general it attempts to find the parameter values that maximize the likelihood function, given the observations. It is effectively the same as using the maximum of the posterior probability in a Bayesian analysis with flat priors. For the derivation one starts -as usual- with Bayes Theorem:

$$P(\lambda|D) = \frac{P(D|\lambda)P_0(\lambda)}{\int \cdots d\lambda} \tag{6.5}$$

using the constant prior:

$$P(\lambda|D) \propto P(D|\lambda) \tag{6.6}$$

Since the integral just provides normalization, one can define the likelihood (L) as:

$$L(\lambda) = P(D|\lambda) \propto P(\lambda|D) \tag{6.7}$$

So to find the MLE for p one has to find the maximum of the given Bernoulli distribution:

$$\frac{\partial}{\partial p} P(x|p) \stackrel{!}{=} 0 \tag{6.8}$$

Its easier to work with the log-likelihood, the maximum will be the same:

$$\frac{\partial}{\partial p} ln(P(x|p)) = \frac{x}{p} - \frac{1-x}{1-p} \stackrel{!}{=} 0 \tag{6.9}$$

Thus, the maximum likelihood estimator for p is:

$$\hat{p} = x$$

Finally one can state the expected distribution for $\hat{p} - p_0$. For that one can use the mean value theorem:

$$f'(c) = \frac{f(b) - f(a)}{b-a} \tag{6.10}$$

For the given case one can identify $a = \hat{p}$, $b = p_0$, $c = p_1$ and $f = L_n'$. Thus:

$$\hat{p} - p_0 = \frac{-L_n'(p_0)}{L_n''(p1)} \tag{6.11}$$

where $p_1 \in [p_0, p]$ and:

$$L_n = \prod_{i=0}^{n} p^{x_i}(1-p)^{1-x_i} \tag{6.12}$$

thus

$$L_n'(p_0) = \frac{\partial}{\partial p} L_n\big|_{p_0} = L_n'(p_0) \tag{6.13}$$

Since elements of $L_n$ do not dependent on n the central limit theorem applies. This means $L_n$ is normal distributed:

$$\Rightarrow p_1 \rightarrow p_0 \tag{6.14}$$

$$\Rightarrow \hat{p} - p_0 = \frac{L_n'(p_0)}{L_n''(p_0)} \tag{6.15}$$

$$\Rightarrow \sigma^2 = \frac{E[L_n'(p_0)]}{nI^2(p_0)} = \frac{1}{nI(p_0)} \tag{6.16}$$

$$\Rightarrow \hat{p} - p_0 = N(0, \frac{1}{\sqrt{nI(p_0)}}) \tag{6.17}$$

### 6.2.2 Problem 2

In the last exercise one is given a family of exponential distributions with the PDF:

$$P(x|p) = \lambda e^{-\lambda x} \qquad x \geq 0 \tag{6.18}$$

Now one is asked to generate n=2,10,100 values for x with

$$x = -ln(U) \tag{6.19}$$

where U is is a uniformly distributed random number between zero and one. From this this generated data one has to find the maximum likelihood estimator. The likelihood for this problem is given by:

$$L(\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x} = \lambda^n e^{-\lambda \sum_{i=1}^{n} x} \tag{6.20}$$

The mode of L is the MLE:

$$\frac{d}{d\lambda} ln(\lambda) = \frac{n}{\lambda} - \sum_{i=1}^{n} x \stackrel{!}{=} 0 \tag{6.21}$$

Thus

$$\lambda^* = n \left( \sum_{i=1}^{n} x \right)^{-1} \tag{6.22}$$

For the standard deviation one uses the second derivative:

$$\Delta\lambda = \sigma = \left[ \frac{-d^2}{d\lambda^2} ln(L(\lambda)) \right] -1/2 = \left( \frac{n}{\lambda^2} \right)^{-1/2} = \frac{\lambda^*}{\sqrt{n}} \tag{6.23}$$

This shall be repeated for 100000 experiments. The resulting distribution of the MLE can be found in figure 6.1 (the figure was produced using the script A.16 ). Comparing the three

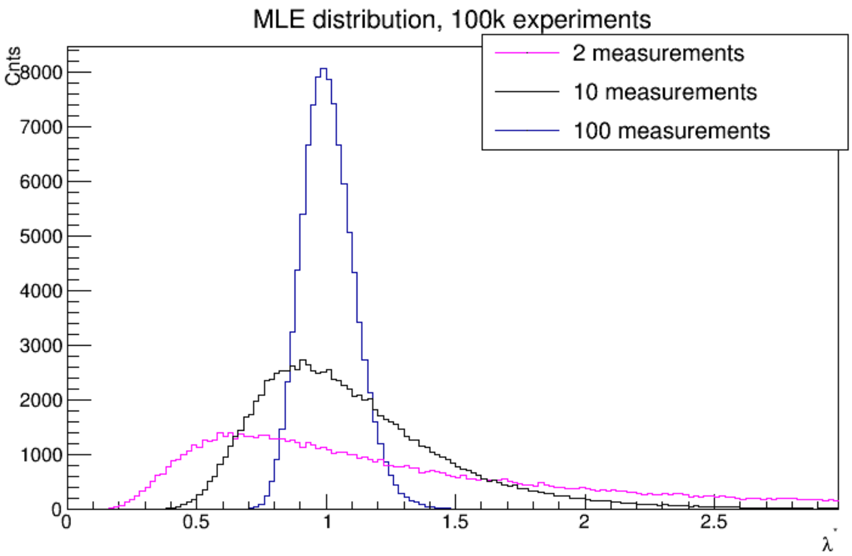


**Figure 6.1:** $\lambda^*$ distribution for different numbers of measurements for 100000 experiments. Maximum of MLE distribution: 0.59, 0.91 and 0.99 for 2, 10 and 100 measurements

curves from figure 6.1 with each other, one can see that the CLT applies. It states that for independent added random variables, their sum tends toward a normal distribution for large enough numbers of measurements. This can be nicely seen since the curves "morphs" into a normal distribution for larger numbers of measurements. Together with the large number theorem which states that for enough measurements the average will be close to the expected value (in this problem the expected value is $\lambda_0 = 1$), one can see that the mean of the normal distribution of the histogram with 100 measurements is 0.99 which is already very close to the expected MLE.

# Appendix A

# Code

## A.1 Find smallest Interval

```
//conf:Confident lvl; eg. 68% -> conf=0.68, eps is step size

vector<double> findSI(TF1* f, double mode, double conf, double eps=0.01)
{
        vector<double> interval;
        double low = mode;
        double high = mode;
        while(true)
        {
                if(f->Integral(low,high) >= conf)
                {
                        interval.push_back(low);
                        interval.push_back(high);
                        break;
                }

                if(f->Eval(low-eps)>f->Eval(high+eps))
                        low-=eps;
                else
                        high+=eps;
                if(low <= 0)
                        low = 0;
        }
        interval.push_back(mode);
        cout << low << endl;
        cout << high << endl;
```

```
        cout << mode << endl;
        return interval;
}
```

## A.2 Problem 2.10

```
TF1* getPostProbDis(int N, int r)
{
        TF1* f = new TF1("f","ROOT::Math::binomial_pdf([0],x,[1])*([1]+1)",0,1);
        f->SetParameters(r,N);
        return f;
}

vector<double> findSI(TF1* f, double mode, double conf, double eps
=0.01)
{
        vector<double> interval;
        double low = mode;
        double high = mode;
        while(true)
        {
                if(f->Integral(low,high) >= conf)
                {
                        interval.push_back(low);
                        interval.push_back(high);
                        break;
                }

                if(f->Eval(low-eps)>f->Eval(high+eps))
                        low-=eps;
                else
                        high+=eps;

                if(low <= 0)
                        low = 0;
                if(high >= 1)
                        high = 1;
                if(low == 0 && high == 1)
                        break;
        }
        interval.push_back(mode);
        cout << low << endl;
```

```
        cout << high << endl;
        cout << mode << endl;
        return interval;
}


void DrawIt()
{
  TCanvas* c = new TCanvas("c","c");
  vector<TF1*> f;
  int N[8] = {100,100,100,100,100,1000,1000,1000};
  int r[8] = {0,4,20,58,92,987,995,998};
  float E[8] = {0.5,1.5,2.0,2.5,3.0,3.5,4.0};
  for(int i = 0; i < 8; i++)
  {
          f.push_back(getPostProbDis(N[i],r[i]));
          f[i]->SetLineColor(i+1);
          if(i== 0)
          {
                  f[i]->Draw();
                  f[i]->GetXaxis()->SetTitle("p");
                  f[i]->GetYaxis()->SetTitle("P");
          }
          else
                  f[i]->Draw("SAME");
  }
  TLegend* legend = new  TLegend(0.1,0.7,0.28,0.9);
  legend->AddEntry(f[0],"E = 0.5","l");
  legend->AddEntry(f[1],"E = 1.0","l");
  legend->AddEntry(f[2],"E = 1.5","l");
  legend->AddEntry(f[3],"E = 2.0","l");
  legend->AddEntry(f[4],"E = 2.5","l");
  legend->AddEntry(f[5],"E = 3.0","l");
  legend->AddEntry(f[6],"E = 3.5","l");
  legend->AddEntry(f[7],"E = 4.0","l");
  legend->Draw();

  TCanvas* c1 = new TCanvas("c1","c1");
  vector<float> prior, low, high,xE;
  float fN[8];
  for(int i = 0; i < 8; i++)
  {
```

```
        cout << i << endl;
        fN[i] = (float)N[i];
        vector<double> interval = findSI(f[i],r[i]/fN[i],0.68,0.0001);
        prior.push_back((float)interval[2]);
        low.push_back(prior[i]-(float)interval[0]);
        high.push_back((float)interval[1]-prior[i]);
        xE.push_back(0.);


  }
  TGraphAsymmErrors* gr =
        new TGraphAsymmErrors(8,&E[0],&prior[0],&xE[0],&xE[0],&low[0],&high[0]);
  gr->SetLineWidth(3);
  gr->SetLineColor(4);
  gr->GetXaxis()->SetTitle("Energy");
  gr->GetYaxis()->SetTitle("p");
  gr->Draw("AP");
}
```

## A.3 Find Central Interval

```
vector<int> getCentralInterval(int N,double p, double confi) // confi = 1-alpha
{
        vector<int> interval;
        int low = 0;
        int high = N;
        if(ROOT::Math::binomial_pdf(low,p,N) <= confi/2)
        {
                double P = 0;
                while(P <= confi/2)
                {
                        P += ROOT::Math::binomial_pdf(low,p,N);
                        low ++;
                }
                low--;
        }
        interval.push_back(low);

        if(ROOT::Math::binomial_pdf(high,p,N) <= confi/2)
        {
                double P = 0;
                while(P <= confi/2)
```

```
                {
                        P += ROOT::Math::binomial_pdf(high,p,N);
                        high--;
                        if(high == 0)
                                break;
                }
                if(high != 0)
                        high++;
        }
        interval.push_back(high);
        return interval;
}
```

## A.4 Problem 2.11

```
vector<double> getConfidenceLevel(int N, int r, double confi)
{
        vector<double> Clvl;
        double high = 0;
        double low = 1;
        for(double p = 1; p >= 0; p-= 0.001)
        {
                vector<int> CintD = getCentralInterval(N,p,confi);
                if(CintD[0] <= r)
                {
                        high = p;
                        break;
                }

        }
        for(double p = 0; p <= 1; p+= 0.001)
        {
                vector<int> CintD = getCentralInterval(N,p,confi);
                if(CintD[1] >= r)
                {
                        low = p;
                        break;
                }

        }

        Clvl.push_back(low);
```

```
        Clvl.push_back(high);

        return Clvl;

}
```

## A.5 Draw function for 2.11

```
void DrawIt()
{
  int N[8] = {100,100,100,100,100,1000,1000,1000};
  int r[8] = {0,4,20,58,92,987,995,998};
  float E[8] = {0.5,1.5,2.0,2.5,3.0,3.5,4.0};

  TCanvas* c1 = new TCanvas("c1","c1");
  vector<float> prior,low, high,xE;
  for(int i = 0; i < 8; i++)
  {
        cout << i << endl;
        vector<double> interval = getConfidenceLevel(N[i],r[i],1-0.90);
        prior.push_back((float)((interval[1]+interval[0])/2));
        low.push_back((float)(prior[i]-interval[0]));
        high.push_back((float)(interval[1]-prior[i]));
        xE.push_back(0.);


  }
  TGraphAsymmErrors* gr =
        new TGraphAsymmErrors(8,&E[0],&prior[0],&xE[0],&xE[0],&low[0],&high[0]);
  gr->SetLineWidth(3);
  gr->SetLineColor(4);
  gr->GetXaxis()->SetTitle("Energy");
  gr->GetYaxis()->SetTitle("p");
  gr->SetTitle("90 percent CL probability range (central interval)");
  gr->Draw("AP");
}
```

## A.6 Get Upper Limit

```
#include "PatrickLib.C" //Previous defined functions

double getUpperLimit(double conf, TF1* f, double low = 0,double step = 0.0001)
```

```
{
        double lim = 0;
        double com = 0;

        while(true)
        {
                com = f->Integral(0,lim);
                if(com >= conf)
                        break;
                lim += step;
        }
        cout << "Upper␣limit␣is:␣" << lim << endl;
        return lim;
}

TF1* f = new TF1("f","(exp(-x)*x**9)/TMath::Factorial(9)");
double lim = getUpperLimit(0.05,f);
```

## A.7 Problem 3.7

```
#include "PatrickLib.C" //previous defined functions

vector<vector<double>> data;

//First define the Poisson function and mode
TF1* f = new TF1("f","(exp(-x)*x**[0])/TMath::Factorial([0])");
int mode = 0;

void DoIt()
{
        TH1* h2 = new TH1F("h2", "h2␣title", 50, 0.0, 50.0);
        TH1* h3 = new TH1F("h3", "h3␣title", 50, 0.0, 50.0);
        int counter = 0;
        //Get some data for different nu and n
        for(int n = 0; n <= 50; n++)
        {
                for(double nu = 0; nu < 50; nu+=0.666)
                {
                        mode = floor(nu);
                        f->SetParameter(0,n);
                        data.push_back(findSI(f, mode, 0.68, 0.01));
```

```
                        if(h2->GetBinContent(n)<data[counter][0])
                                h2->SetBinContent(n,data[counter][0]);
                        if(h3->GetBinContent(n)>data[counter][1]
                || h3->GetBinContent(n) == 0)
                                h3->SetBinContent(n,data[counter][1]);
                        counter++;
                }
        }
        h3->Draw();
        h3->SetLineColor(2);
        h2->Draw("SAME");
        h3->GetXaxis()->SetTitle("n");
        h3->GetYaxis()->SetTitle("#nu");
        h3->SetTitle("68␣percent␣Neyman␣(smallest␣interval)");
}
```

## A.8 Feldman-Cousins

```
void FeldmanCousins()
{
        TFeldmanCousins *f = new TFeldmanCousins(0.68);

        // calculate either the upper or lower limit for 9 observerd
        // events with an estimated background of 3.2.  The calculation of
        // either upper or lower limit will return that limit and fill
        // data members with both the upper and lower limit for you.

        Double_t Nobserved    = 9.0;
        Double_t Nbackground = 3.2;


        Double_t ul = f->CalculateUpperLimit(Nobserved, Nbackground);
        Double_t ll = f->GetLowerLimit();

        cout << "For␣" <<  Nobserved << "␣data␣observed␣with␣and␣estimated␣background"<<endl;
        cout << "of␣" << Nbackground << "␣candidates,␣the␣Feldman-Cousins␣method␣of␣"<<endl;
        cout << "calculating␣confidence␣limits␣gives:"<<endl;
        cout << "\tUpper␣Limit␣=␣" <<  ul << endl;
        cout << "\tLower␣Limit␣=␣" <<  ll << endl;
        cout << "at␣the␣"<< f->GetCL()*100 << "%␣CL" << endl;
}
```

## A.9 Smallest Interval + Background

```
void BaySIBkg(double bkg, double conf,int n, double eps =0.001)
{
        double denum = 0;
        for(int i = 0; i<=n;i++)
        {
                denum += pow(bkg,i)/TMath::Factorial(i);
        }
        denum*=TMath::Factorial(n);

        TF1* pdf = new TF1("pdf","(TMath::Exp(-x)*([1]+x)^([2]))/([3])",0,25);
        pdf->SetParameters(bkg,n,denum);
        pdf->SetLineColor(kBlue);
        pdf->Draw();
        double mode = n-bkg;
        double value = pdf->Eval(mode);
        double low = mode;
        double high = mode;
        while(pdf->Integral(low,high) < conf)
        {
                value -= eps;
        if(pdf->Eval(low-eps) > pdf->Eval(high+eps))
                        low = pdf->GetX(value,0,pdf->GetMaximumX());
        else
                        high = pdf->GetX(value,pdf->GetMaximumX(),50);
        }

        TF1* range = (TF1*)pdf->Clone("range");
        range->SetRange(low,high);
        range->SetFillColor(kBlack);
        range->SetFillStyle(3005);
        range->Draw("SAME FC");

        cout << "calculating credibility limits gives:"<<endl;
        cout << "\tMode = " << mode << endl;
        cout << "\tUpper Limit = " << high << endl;
        cout << "\tLower Limit = " << low << endl;
        cout << "at the "<< conf*100 << "% SI" << endl;
}
```

## A.10 Problem 4.8 - CLT working

```
double randomLn(TRandom* rGen, double l)
{

        double n = rGen->Uniform();
        return -TMath::Log(n)/l;
}
void CLTTest(int n, int N, double l, TH1F* h)
{
        TRandom* gen = new TRandom();
        for(i=0;i<N;i++)
        {
                double x = 0;
                for(int j=0;j<n;j++)
                {
                        x += randomLn(gen,l);
                }
                x = x/n;
                double p = l*TMath::Exp(-l*x);
                h->Fill(x);
        }
 }

void doIt()
{
        double l[3] = {1,0.25,1./8};
        double n[3] = {100,10,4};

        vector<TH1F*> hh;
        TCanvas* c = new TCanvas("c","");
        c->Divide(3);
        int z = 0;
        for(int i=0; i<3; i++)
        {
                cout << i << endl;
                c->cd(i+1);

                auto legend = new TLegend(0.1,0.7,0.48,0.9);
                for(int j=0; j<3; j++)
                {
                        z = j +3*i;
```

```
                    hh.push_back(new TH1F(Form("%d",z),"",5000,0,50));
                    CLTTest(n[j],100000,l[i],hh[z]);
                    hh[z]->SetLineColor(1+j);
                    legend->AddEntry(hh[z],Form("n␣=␣%d",(int)n[j]),"l");
                    hh[z]->Draw("SAME");
                }
                legend->Draw();
                hh[i*3]->SetTitle(Form("CLT␣for␣#lambda␣=␣%.3f",l[i]));
        }

}
```

## A.11 Problem 4.8 - CLT not working

```
void produceDist(int N, TH1F* h)
{
        TF1* c = new TF1("c","3*TMath::Tan(TMath::Pi()*x␣-␣TMath::Pi()/2)␣+␣25");
        c->Draw();
}

void produceDistMean(int n, int N, TH1F* h)
{
        TRandom* gen = new TRandom();
        double U;
        for(int i=0; i<N;i++)
        {
                double x = 0;
                for (int j= 0; j<n;j++)
                {
                        U = gen->Uniform();
                        x += (3*TMath::Tan(TMath::Pi()*U - TMath::Pi()/2) + 25);
                }
                x = x/n;
                h->Fill(x);
        }
}
```

## A.12 Sigmoid Model in BAT

```
// === HEADER ===

#ifndef __SIGMOIDMODEL__H
```

```
#define __SIGMOIDMODEL__H

#include <BAT/BCModel.h>

// ---------------------------------------------------------
class SigmoidModel : public BCModel
{
public:

    SigmoidModel(const std::string& name);

    ~SigmoidModel();


    double LogLikelihood(const std::vector<double>& pars);
    double LogSigmoid(double x,double E, double A);
    double LogSigmoidAnti(double x,double E, double A);

protected:

};
// ---------------------------------------------------------

#endif

// === SOURCE ===

#include "SigmoidModel.h"

#include <BAT/BCGaussianPrior.h>
#include <BAT/BCMath.h>
#include <BAT/BCPositiveDefinitePrior.h>

#include <cmath>

// ---------------------------------------------------------
SigmoidModel::SigmoidModel(const std::string& name)
    : BCModel(name)
{
    // add parameters
    AddParameter("E0", 0., 5., "E_0");
    AddParameter("A", 0., 5., "A");
```

```
    // and set it's priors gausian
    GetParameter("E0").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(2.0, 0.3)));
    GetParameter("A").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(2.0, 0.5)));
}

// ---------------------------------------------------------
SigmoidModel::~SigmoidModel()
{
}

// ---------------------------------------------------------

double SigmoidModel::LogSigmoid(double x,double E, double A)
{
        return log((1./(1+exp(-A*(x-E)))));
}
double SigmoidModel::LogSigmoidAnti(double x,double E, double A)
{
        return log(1-(1./(1+exp(-A*(x-E)))));
}


double SigmoidModel::LogLikelihood(const std::vector<double>& parameters)
{
    // This methods returns the logarithm of the conditional probability
    // p(data|parameters).
    double result = 0;
    int n = 8;
    int count = 0;
    int r[8] = {0,4,22,55,80,97,99,99};
    int N[8] = {100,100,100,100,100,100,100,100};
    for(double i = 0.5; i <=4.0;i+=0.5)
    {
            //calculate binomial part
            result += BCMath::LogBinomFactor(N[count], r[count]);
            //calculate sigmoid part
            result += r[count]*LogSigmoid(i,parameters[0],parameters[1]);
            result += (N[count]-r[count])*LogSigmoidAnti(i,parameters[0],parameters[1]);

            count++;
    }
```

```
    return result;
}

// === Executing script ===

#include "SigmoidModel.h"

#include <BAT/BCGaussianPrior.h>
#include <BAT/BCMath.h>
#include <BAT/BCPositiveDefinitePrior.h>

#include <cmath>

// ---------------------------------------------------------
SigmoidModel::SigmoidModel(const std::string& name)
    : BCModel(name)
{
    // add parameters
    AddParameter("E0", 0., 5., "E_0");
    AddParameter("A", 0., 5., "A");

    // and set it's priors gausian
    GetParameter("E0").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(2.0, 0.3)));
    GetParameter("A").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(2.0, 0.5)));
}

// ---------------------------------------------------------
SigmoidModel::~SigmoidModel()
{
}

// ---------------------------------------------------------

double SigmoidModel::LogSigmoid(double x,double E, double A)
{
        return log((1./(1+exp(-A*(x-E)))));
}
double SigmoidModel::LogSigmoidAnti(double x,double E, double A)
{
        return log(1-(1./(1+exp(-A*(x-E)))));
}
```

```
double SigmoidModel::LogLikelihood(const std::vector<double>& parameters)
{
    // This methods returns the logarithm of the conditional probability
    // p(data|parameters).
    double result = 0;
    int n = 8;
    int count = 0;
    int r[8] = {0,4,22,55,80,97,99,99};
    int N[8] = {100,100,100,100,100,100,100,100};
    for(double i = 0.5; i <=4.0;i+=0.5)
    {
            //calculate binomial part
            result += BCMath::LogBinomFactor(N[count], r[count]);
            //calculate sigmoid part
            result += r[count]*LogSigmoid(i,parameters[0],parameters[1]);
            result += (N[count]-r[count])*LogSigmoidAnti(i,parameters[0],parameters[1]);

            count++;
    }
    return result;
}
```

## A.13 Toy Test for Sigmoid Model

```
void toyTest(int N)
{
        TRandom *eventGenerator = new TRandom();
        TF1* f = new TF1("f","1/(1+TMath::Exp(-[0]*(x-[1])))",0,25);
        TH2F *h2 = new TH2F("h2", "A-E_0 Plot", 400, 0, 4, 300, 0, 3);
        static int trials = 100;
        static int dataPoints = 8;
        double prob;
        double logToyData =0;
        double logToy[N];
        int r;
        double eData[8] = {0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0};
        int rData[8] = {0,4,22,55,80,97,99,99};
        int lvl[3] = {(int)(N*(1-0.68)),(int)(N*(1-0.90)),(int)(N*(1-0.95))};
        int counter =0;

        for(double E = 1.8;E<=2.1;E+=0.01)
```

```
	{
		for(double A = 2.2;A<=3.6;A+=0.01)
		{
			f->SetParameters(A,E);
			for(int i=0; i<N;i++)
			{	logToy[i] = 0;
				logToyData = 0;
				for(int j = 0; j<dataPoints;j++)
				{
					prob = f->Eval(eData[j]);
					r = eventGenerator->Binomial(trials,prob);
					logToy[i] += TMath::Log(
					ROOT::Math::binomial_pdf(r,prob,trials));
					if(N-i <2)
						logToyData += TMath::Log(
						ROOT::Math::binomial_pdf(rData[j],prob,trials));
				}
			}
			sort(logToy, logToy + N);
			counter+=1000;

			if(logToy[lvl[0]] < logToyData)
				h2->SetBinContent(A*100,E*100,60);
			else if(logToy[lvl[1]] < logToyData)
				h2->SetBinContent(A*100,E*100,90);
			else if(logToy[lvl[2]] < logToyData)
				h2->SetBinContent(A*100,E*100,95);
		}
		cout << "Calculated " <<counter<< " data points" <<endl;
	}
}
```

## A.14 SINUS MODEL IN BAT

```
// === SOURCE ===
#include "SinusModel.h"

#include <BAT/BCGaussianPrior.h>
#include <BAT/BCMath.h>
#include <BAT/BCPositiveDefinitePrior.h>

#include <cmath>
```

```
// ------------------------------------------------------------
SinusModel::SinusModel(const std::string& name)
    : BCModel(name)
{
    // add parameters
    AddParameter("E0", 0., 1.5, "E_0");
    AddParameter("A", 0., 1., "A");

    // and set it's priors gausian
    GetParameter("E0").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(0.8, 0.2)));
    GetParameter("A").SetPrior(new BCPositiveDefinitePrior(new BCGaussianPrior(0.5, 0.1)));
}

// ------------------------------------------------------------
SinusModel::~SinusModel()
{
}

// ------------------------------------------------------------


double SinusModel::LogSinus(double x,double E, double A)
{
        return log(abs(sin(A*(x-E))));
}
double SinusModel::LogSinusAnti(double x,double E, double A)
{
        return log(abs(1.-(sin(A*(x-E)))));
}


double SinusModel::LogLikelihood(const std::vector<double>& parameters)
{
    // This methods returns the logarithm of the conditional probability
    // p(data|parameters).
    double result = 0;
    int n = 8;
    int count = 0;
    int r[8] = {0,4,22,55,80,97,99,99};
    int N[8] = {100,100,100,100,100,100,100,100};
    for(double i = 0.5; i <=4.0;i+=0.5)
```

```
    {
            if(0. != abs(sin(parameters[1]*(i-parameters[0]))) && abs(sin(parameters[1]*(i-param
            {
                    //calculate binomial part
                    result += BCMath::LogBinomFactor(N[count], r[count]);
                    //calculate sinus part
                    result += r[count]*LogSinus(i,parameters[0],parameters[1]);
                    result += (N[count]-r[count])*LogSinusAnti(i,parameters[0],parameters[1]);

            }
            count++;
    }
    return result;
}
```

The executed script is similar to the one in A.12. The header file is not given.

## A.15 $\chi^2$ Script

```
#include <TGraphErrors.h>
#include <TF1.h>
#include <TCanvas.h>
#include <TRandom3.h>

#include <BAT/BCAux.h>
#include <BAT/BCLog.h>
#include <BAT/BCGraphFitter.h>


// The macro performs a quadratic fit to the data. The fit function
// is defined using the ROOT TF1 object and the data to fit are stored
// in the TGraphErrors object. The y-errors have to be defined in the
// TGraphErrors for the fit to work.

// ---------------------------------------------------------
void graphFitterSimpleExample()
{
    // open log file
    BCLog::OpenLog("log.txt", BCLog::detail, BCLog::detail);

    // set nicer style for drawing than the ROOT default
```

```
BCAux::SetStyle();

// -------------------------
// define a fit function, also used to create data
TF1 f1("f1", "[0] + [1]*x + [2]*x^2",0.,100.0);//+ [3]/((x-[4])^2+([5])^2) ", 0.0, 100.0);
f1.SetParNames("A", "B","C");//,"D","x_0","#Gamma");

// Parameter limits must be defined for every parameter
f1.SetParLimits(0, 13000, 15000.0);
f1.SetParLimits(1, -13000.0,-11000.0);
f1.SetParLimits(2,  2000.0, 4000.0);
//f1.SetParLimits(3, -24000.0,-20000.0);
//f1.SetParLimits(4, -1.0,0.0);
//f1.SetParLimits(5, -50.0,0.0);
// -------------------------

// -------------------------
// Create data
Double_t x[18]= {0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50
     ,0.55,0.60,0.65,0.70,0.75,0.80,0.85,0.90,0.95};
Double_t y[18]= {11.3,19.9,24.9,31.1,37.2,36.0,59.1,77.2,96.0
     ,90.3,72.2,89.9,91.0,102.0,109.7,116.0,126.6,139.8};
Double_t xE[18];
Double_t yE[18];
for(int i = 0; i<18;i++)
{
     xE[i] = 0;
     yE[i] = 4;
}

// create graph
TGraphErrors graph;

// using sigma=4 as y uncertainty

graph  = TGraphErrors(18,x,y,xE,yE);
// -------------------------

// create a new graph fitter
BCGraphFitter gf(graph, f1);

// set Metropolis as marginalization method
```

```
    gf.SetMarginalizationMethod(BCIntegrate::kMargMetropolis);

    // set precision
    gf.SetPrecision(BCEngineMCMC::kMedium);

    // perform the fit
    gf.Fit();

    // print data and the fit
    TCanvas c1("c1");
    gf.DrawFit("", true);
    c1.Print("fit.pdf");

    // print marginalized distributions
    gf.PrintAllMarginalized("distributions.pdf");

    // print results
    gf.PrintSummary();
}
```

## A.16 MLE DISTRIBUTION SCRIPT

```
double randomLn(TRandom* rGen)
{

        double n = rGen->Uniform();
        return -TMath::Log(n);
}

double generateMLE(int n,TRandom* rGen)
{
        double MLE = 0;
        for(int i= 0; i<n;i++)
        {
                MLE += randomLn(rGen);
        }
        MLE = 1./MLE;
        MLE = MLE*n;

        return MLE;
}
```

```
double generateMLEDev(double MLE, int n)
{
        return MLE/(TMath::Sqrt((double)n));
}

void generateMLEDist(TH1F* hist, int n, TRandom* rGen, int rep = 1000)
{
        for(int i=0; i<rep; i++)
        {
                double MLE = generateMLE(n,rGen);
                hist->Fill(MLE);
        }
}
```