Datenanalyse in der Physik

Vorlesung 7

Das Minimierungsprogramm MINUIT und Konfidenzgrenzen

Prof. Dr. J. Mnich

DESY und Universität Hamburg

MINUIT

MINUIT ist ein Programm zur numerischen Berechnung

- des Minimums einer Funktion $F(\vec{a}\,)$, die von mehreren (max. 50) Parametern a_i abhängt
- der Kovarianzmatrix (Fehlermatrix) dieser Parameter
- der Fehlerintervalle (parabolische Näherung oder asymmetrisch) der Parameter
  aus $F_{\min} + \Delta$ für beliebige Werte von Δ
- Konturen für Paare a_i, a_j der Parameter

Das Programm ist von F. James am CERN entwickelt worden

- Ursprünglich in FORTRAN geschrieben
- in der ROOT-Bibliothek auf C++ adaptiert

MINUIT

MINUIT enhält ausgefeilte Algorithmen um auch sehr komplexe Probleme lösen zu können

Informationen und Dokumentation:

- HTML Dokumentation

  `http://wwwinfo.cern.ch/asdoc/WWW/minuit/minmain/tableofcontents2_2.html`

- Übersicht der MINUIT Kommandos

  `http://wwwinfo.cern.ch/asdoc/hbook_html3/node125.html`

- Referenzhandbuch (für die FORTRAN-Version)

  `http://particle-physics.desy.de/e242233/e259021/minuit.pdf`

Benutzung von MINUIT

- Einbinden der ROOT Bibliothek

  siehe Übung 04

  Beispiel: Compilieren und Linken des Programms `MINUIT-template.C`:

  `make MINUIT-template`

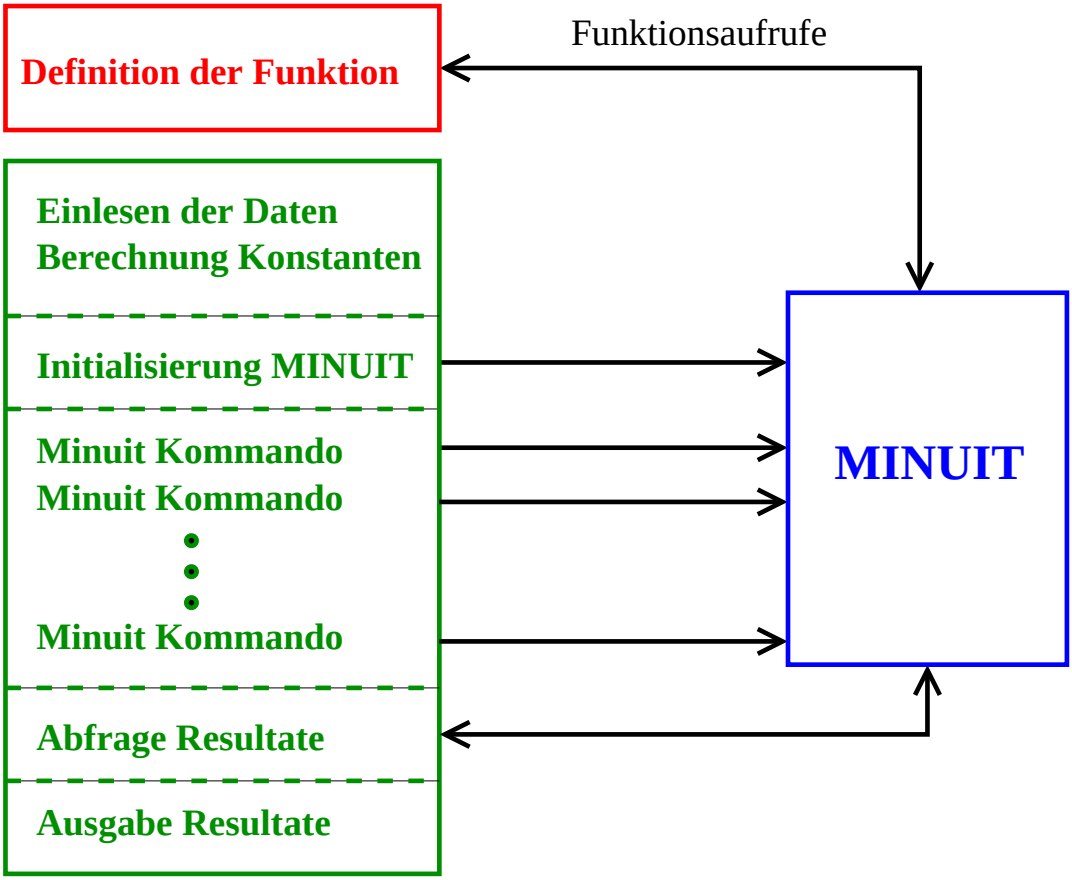
- Aufruf im C Programm

  Das C-Program muss die zu minimierende Funktion definieren und eine Abfolge von MINUIT Kommandos aufrufen

  Typischer Programmablauf:

  - Einlesen der Daten und Definition der Funktion
  - Initialisierung von MINUIT
  - Abfolge von MINUIT Kommandos
  - Ausgabe der Resultate

Blockdiagramm



Beispiel: Radioaktiver Zerfall

Als Beispiel wollen wir hier ein MINUIT-Programm `MINUIT-template.C` entwickeln, das unser Experiment zum radioaktiven Zerfall auswertet

1.Schritt: Einlesen der Daten (hier vereinfachend) und Definition der zu minimierenden Funktion.

```
/* Beispiel MINUIT Fit der Nullpunktsmessung aus der Vorlesung    */
/* Gebe vereinfachend die Daten als globale Variablen ein         */
/* und berechne die Fakultaeten ein fuer alle Mal                 */
#define NDATA 18
int r[NDATA]    = {1,1,  5, 4,2,0,3,2, 4,1,2,1,1,0,1,1,2,1};
int rfak[NDATA] = {1,1,120,24,2,1,6,2,24,1,2,1,1,1,1,1,2,1};

/* Berechne den negativen log Likelihood */
void fcn(int &npar, double *gin, double &f, double *par, int iflag)
{
   int i;
   double mu, lnL;

   mu = par[0];       /* momentaner MINUIT Wert des Parameters */
   lnL = 0.0;         /* log Likelihood, Formel aus Vorlesung  */
   for(i=0;i<NDATA; i++) {
     lnL += r[i]*log(mu) - mu - log((double) rfak[i]);
   }
   f = -lnL;          /* negativer log Likelihood */
}
```

Definition der Funktion

Im C-Programm muss die zu minimierende Funktion enthalten sein

- Der Name (in unserem Beispiel `fcn`) wird in der MINUIT Initialisierung übergeben
  `minuit.SetFCN(fcn);`
- Die Funktionsargumente sind festgelegt und sehen so aus:
  `void fcn(int &npar, double *gin, double &f, double *par, int iflag)`

`npar`	Zahl der Parameter
`gin`	Feld der partiellen Ableitungen (Rückgabe)
`f`	Funktionswert (Rückgabe)
`par`	Feld mit Parameterwerten
`iflag`	Flagge

Für uns wichtig sind nur `f` und `par`

Universität Hamburg

Beispiel: Initialisierung

2.Schritt: Initialisierung von MINUIT

```
main()
{
   double arglist[10];    /* Argumentenliste fuer Minuit */
   int ierflg = 0;        /* Fehlercode fuer Minuit */

   /* MINUIT Parameterdefinition */
   double start   = 1.0;  /* Startwert (geschaetzt)                          */
   double schritt = 0.1;  /* Anfangsschrittweite, Groessenordnung Fehler */
   double l_bnd = 0.1;    /* mu>0 wegen log(mu) in Poisson-Funktion      */
   double u_bnd = 10.;

   /* Initialisiere Minuit mit 1 Parameter */
   TMinuit minuit(1);

   /* Definiere zu minimierende Funktion */
   minuit.SetFCN(fcn);

   /* Definiere Parameter und setze Startwerte und Schrittweiten */
   minuit.mnparm(0,"Poisson mu",start,schritt,l_bnd,u_bnd,ierflg);
}
```

Universität Hamburg

Initialisierung

- Die folgende Header-Datei muss eingebunden werden

```
#include <TMinuit.h>
```

- Im ausführbaren Teil des Programms müssen folgenden Informationen durch Funktionsaufrufe übermittelt werden:

1. Zahl der Parameter, nach denen die Funktion minimiert werden soll:
   `TMinuit minuit(1);`
2. Name der zu minimierenden Funktion `fname`:
   `minuit.SetFCN(fname);`
3. Jeder der Parameter muss durch folgenden Aufruf definiert werden:
   `minuit.mnparm(num,name,start_value,step_size, l_bound,u_bound,err_flag);`

Universität Hamburg

Definition der Parameter

```
minuit.mnparm(num,name,start_value,step_size,
l_bound,u_bound,err_flag);
```

Variable	Typ	Bedeutung
`num`	`int`	Index des Parameters ($\rightarrow$ Funktion `fname`)
`name`	`char`	Name des Parameters
`start_value`	`double`	Startwert für diesen Parameter
`step_size`	`double`	Schrittweite oder ungefährer Fehler
`l_bound`	`double`	untere Grenze des Wertebereiches
`u_bound`	`double`	obere Grenze des Wertebereiches
`err_flg`	`int`	Rückgabe: Fehlercode

- Für `start_value` und `step_size` sollte man sich sinnvolle Werte überlegen (auch wenn das Ergebnis i.a. nicht davon abhängt)
- `l_bound =0.0` bzw. `u_bound=0.0` heisst, dass Wertebereich des Parameters unbeschränkt ist

  Man sollte den Wertebereich nur beschränken, um mathematische Probleme zu vermeiden
  z.B. falls Funktion `fname` Wurzel oder Logarithmen des Parameters bildet

  Sonst nach Möglichkeit den Wertebereich immer unbeschränkt lassen

Universität Hamburg

Beispiel: MINUIT Komamandos

3.Schritt: Ausführung von MINUIT Kommandos

Die Funktion `MINUITergebnisse(&minuit);` (Teil des Beispiel-C-Programms) gibt Resultate aus

```
  ...
   double arglist[10];     /* Argumentenliste fuer Minuit    */
   int ierflg = 0;         /* Fehlercode fuer Minuit         */
  ...
  /* Setze Fehlergrenze auf 1 sigma */
   arglist[0] = 0.5;
   minuit.mnexcm("SET ERR",arglist,1,ierflg);
   /* Minimiere mit Migrad */
   minuit.mnexcm("MIGRAD",arglist,0,ierflg);
   /* Berechne asymmetrische Fehler mit Minos */
   minuit.mnexcm("MINOS",arglist,0,ierflg);
   /* Ausgabe der Ergebnisse */
   MINUITergebnisse(&minuit);
   /* Setze Fehlergrenze auf 2 sigma */
   arglist[0] = 2.0;
   minuit.mnexcm("SET ERR",arglist,1,ierflg);
   minuit.mnexcm("MINOS",arglist,0,ierflg);
   MINUITergebnisse(&minuit);
}
```


MINUIT Kommandos

Nach der Initialisierung werden MINUIT Kommandos aufgerufen, die

- die Minimierung steuern
- Konturen berechnen
- Parameter fixieren oder wieder variabel machen
- Ausgabe des Programms steuern, etc.
- vollständige Liste:

  `http://wwwinfo.cern.ch/asdoc/hbook_html3/node125.html`

Wir stellen hier nur einige wichtige vor

MINUIT Kommandos haben einen Namen und -die meisten- eine Liste von Argumenten, die teilweise optional sind

- Kommandonamen können bis auf die angegebenen Großbuchstaben abgekürzt werden
  Statt `SET ERRordef` reicht auch `SET ERR`
- Obligatorische Argumente werden durch `<...>`, optionale durch `[...]` gekennzeichnet
  Standardwerte werden benutzt, falls optionale Argumente nicht angegeben werden

MINUIT Kommandos

MINUIT Kommandos können

- über eine Datei eingelesen
- oder durch Funktionaufrufe erteilt werden

Die Syntax für den Funktionsaufruf ist:

`minuit.mnexcm(`*`Kommando`*`,`*`Argumentliste`*`,`*`Argumentzahl`*`,`*`Fehlerflagge`*`);`

- *`Kommando`* ist eine Zeichenkette, die den Kommandonamen enthält
- *`Argumentliste`* ist ein Feld vom Typ `double`, dessen Elemente die Argumente enthält
- *`Argumentzahl`* ist die zu verwendende Zahl der Argumente
- *`Fehlerflagge`* Rückgabe einer Fehlerbedingung ($\neq 0$)

Beispiel: 1-σ Fehler einer Log-Likelihood-Funktion werden so definiert:

```
double arglist[10];  /* 10 Argumente sollten ausreichen    */
int ierflg;          /* Rueckgabe Fehlercode               */
 ...
arglist[0] = 0.5;    /* Fmin + 0.5 bestimmt 1-sigma-Fehler */
minuit.mnexcm("SET ERR",arglist,1,ierflg);
```

Universität Hamburg

Wichtige MINUIT Kommandos

- `SET ERRordef <`*`value`*`>`
  Definiert Funktionswert über dem Minimum, für den Fehler und Konturen berechnet werden sollen
- `MIGrad [`*`maxcalls`*`] [`*`tolerance`*`]`
  sucht nach dem Minimum, erechnet die Kovarianzmatrix und daraus die parabolischen Fehler aller Parameter (effizientester Algorithmus)
  `maxcalls` beschränkt die maximale Zahl der Funktionsaufrufe
  mit *`tolerance`* kann die Abbruchbedingung der Minimierung geändert werden
- `MINOs [`*`maxcalls`*`] [`*`parameter`*`] ...[`*`parameter`*`]`
  bestimmt die genauen, asymmetrischen Fehler für alle variablen Parameter bzw. nur die durch ihren Index angegebenen
  wird aufgrufen, nachdem das Minimum und die Kovarianzmatrix bestimmt ist
- `SCAn [`*`parameter`*`] [`*`points`*`] [`*`FROM`*`] [`*`TO`*`]`
  Berechnet und plottet *`points`* Punkte im angegebenen Intervall des Parameters
- `MNContour <`*`Parameter`*`> <`*`Parameter`*`> [points]`
  berechnet Kontur der Funktion in der Ebene der beiden Parameter
  Funktion wird für alle übrigen Paramter minimiert
  Funktionswert der Kontur ist Minimum plus letzter Wert von `SET ERRordef`

Universität Hamburg

Weitere MINUIT Kommandos

- `FIX <parameter> [parameter] ...[parameter]`
  fixiert die aufgelisteten Parameter auf den momentanen Wert
- `RELease <parameter> [parameter] ...[parameter]`
  Parameter werden wieder variabel
- `HESse [maxcalls]`
  berechnet die Kovarianzmatrix aus der inversen Matrix der zweiten Ableitungen $V = G^{-1}$ mit $G_{ij} = \frac{\partial^2 F(\vec{a})}{\partial a_i \partial a_j}$
  wird von `MIGrad` intern aufgerufen
- `IMProve [maxcalls]`
  sucht nach weiteren lokalen Minima der Funktion
  nützlich, falls man den Verdacht hat, dass das gefundene Minimum nicht das globale ist
- `SIMplex [maxcalls] [tolerance]`
  wie `MIGrad`, benutzt aber einen anderen Algorithmus
- `MINImize [maxcalls] [tolerance]`
  versucht `MIGrad` Algorithmus und, falls dieser nicht konvergiert, wechselt zu `SIMplex`

Universität Hamburg

Resultat des Beispiels

Das Resultat der Berechnungen, ausgegeben mit `MINUITergebnisse`, sieht so aus:

```
Ergebnisse der Minimierung mit MINUIT
-------------------------------------

 Minimaler Funktionswert:                  29.296
 Geschaetzte Differenz wahres Minimum:   1.399e-20
 Zahl der variablen Parameter:             1
 Hoechste Parameternummer:                 1
 Fehlerdefinition (Fmin + Delta):           2.000
 Genaue Kovarianzmatrix wurde erfolgreich berechnet

   Parameter     Wert       Fehler    positiv    negativ    L_BND    U_BND
 0 Poisson mu  1.778e+00  6.285e-01 +7.047e-01 -5.568e-01  1.0e-01  1.0e+01

Kovarianzmatrix:
  3.951e-01

Korrelationsmatrix:
  1.000
```

Universität Hamburg

Konfidenzgrenzen

- Eine physikalische Größe (oder ein Parameter) habe den wahren Wert $\hat{x}$

  Wenn wir Messungen dieser Größe durchführen, unterliegen diese Zufallsvariablen einer Wahrscheinlichkeitsverteilung mit Varianz σ^2

- Im allgemeinen liegt unsere Aufgabe darin, eine Aussage über den wahren Wert dieser Größe zu machen

  Aber: Das experimentelle Resultat x_m (bestimmt z.B. aus Mittelwert, Likelihood, ...) ist i.a. nicht gleich dem wahren Wert $x_m \neq \hat{x}$

  Wir geben das Resultat der Messung durch Messwert und Fehler an:

$$x_m \pm \sigma \qquad \text{oder} \qquad x_m \, {}^{+\sigma_r}_{-\sigma_l}$$

  Was können wir daraus für den wahren Wert $\hat{x}$ schließen?

- Über den wahren Wert $\hat{x}$ lassen sich nur Aussagen machen zur Wahrscheinlichkeit, dass er sich in einem bestimmten Intervall befindet

Konfidenzgrenzen:	obere und untere Intervallgrenzen
Konfidenz (oder Konfidenzniveau):	Betrag dieser Wahrscheinlichkeit

Konfidenzgrenzen

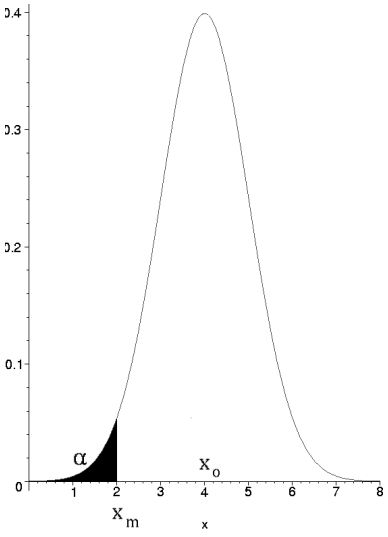
Wir betrachten als Beispiel gaußverteilte Zufallsvariablen (Messwerte)
Der experimentell gefundene Messwert (oder Schätzwert des Parameters) sei x_m

- Durch die Gleichung

$$\alpha = \int_{-\infty}^{x_m} \frac{1}{\sqrt{2\pi\sigma^2}} \, e^{-\frac{(x-x_o)^2}{2\,\sigma^2}} \, \mathrm{d}x$$

  wird die obere Konfidenzgrenze x_o definiert

  Interpretation: Falls x_o der wahre Wert ist, so ist α die Wahrscheinlichkeit den Messwert x_m oder einen kleineren zu finden



- Das ist äquivalent zur Aussage:
  Aus dem beobachteten Messwert x_m folgt, dass der wahre Wert $\hat{x}$ mit der Wahrscheinlichkeit α größer als oder gleich x_o ist

  denn falls $\hat{x} > x_o$ wäre, so wäre auch die Wahrscheinlichkeit einen Wert $x \leq x_m$ zu finden kleiner als α

Konfidenzgrenzen

- Entsprechend definiert man eine untere Konfidenzgrenze x_u

$$\beta = \int_{x_m}^{+\infty} \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-x_u)^2}{2\,\sigma^2}}\, \mathrm{d}x$$

  Das heisst: aus dem beobachteten Messwert x_m folgt, dass der wahre Wert $\hat{x}$ mit der Wahrscheinlichkeit β kleiner als oder gleich x_u ist

- Man kann somit sagen, dass das Intervall $[x_u,\, x_o]$ den wahren Wert mit der Wahrscheinlichkeit $1-\alpha-\beta$ enthält

  oder

  falls das gleiche Experiment sehr oft wiederholt wird, so findet man ein Intervall $[x_u,\, x_o]$, dass $\hat{x}$ enthält in $1-\alpha-\beta$ der Fälle

- Man hat ein Vertrauen (= Konfidenz) von $1-\alpha-\beta$, dass der wahre Wert in diesem Intervall liegt

- Achtung: der wahre Wert ist keine Zufallsvariable, sondern ein fester Wert!

Konfidenzgrenzen

- Beidseitige Konfidenzgrenzen:

  Meist wählt man symmetrische Konfidenzniveaus $\alpha = \beta$

  Falls man bei gaußverteilten Zufallsgrößen $\alpha = \beta = 0,1587 \approx 16\%$ wählt, folgt natürlich $x_u = x_m - \sigma$ und $x_o = x_m + \sigma$

$\Longrightarrow$ Die Angabe eines Messergebnisses $x_m \pm \sigma$ bzw. $x_m\ ^{+\sigma_r}_{-\sigma_l}$ ist gleichbedeutend mit der Aussage, dass der wahre Wert $\hat{x}$ mit

$$1 - 0,1587 - 0,1587 \approx 68\%$$

Wahrscheinlichkeit im Intervall $[x_m - \sigma,\, x_m + \sigma]$ liegt

- Einseitige Konfidenzgrenzen:

  Manchmal ist auch nur eine Aussage über die untere oder obere Grenze des wahren Wertes sinnvoll

  Beispiel: Überprüfung der Radioaktivität eines unbekannten Stoffes, wenn kein Zerfall beobachtet wurde
  Dann sind nur Aussagen möglich, wie z.B. der wahre Wert $\hat{\mu}$ ist mit 95% Wahrscheinlichkeit kleiner als μ_o

- Falls die Messergebnisse nicht gaußverteilt sind, muss in den Definitionen für x_o und x_u natürlich die entsprechende pdf eingesetzt werden

Konfidenzgrenzen für Poisson-verteilte Messgrößen

Im Fall diskreter Zufallsvariablen müssen die Integrale durch Summen ersetzt werden

Für n beobachtete Ereignisse ergibt sich

- obere Konfidenzgrenze μ_o

$$\alpha = \sum_{r=0}^{n} \frac{e^{-\mu_o}\mu_o^r}{r!} = 1 - P(\chi^2_{2n+2} \leq 2\mu_o)$$

- untere Konfidenzgrenze μ_u

$$\beta = \sum_{r=n}^{\infty} \frac{e^{-\mu_u}\mu_u^r}{r!} = P(\chi^2_{2n} \leq 2\mu_u)$$

Diese Definitionen können mit integrierten χ^2-Funktionen gelöst werden:

$P(\chi^2_{2n+2} \leq 2\mu_o)$ ist die χ^2-Wahrscheinlichkeit für $2n+2$ Freiheitsgrade, einen Wert $\leq 2\mu_o$ zu erhalten

Universität Hamburg

Beweis Konfidenzgrenzen Poisson-Verteilung

Aus χ^2-Verteilung für n Freiheitsgrade folgt für $2n+2$ Freiheitsgrade (mit $\Gamma(n+1) = n!$):

$$f_n(\chi^2) = f_n(u) = \frac{\frac{1}{2}\left(\frac{u}{2}\right)^{\frac{n}{2}-1} e^{-\frac{u}{2}}}{\Gamma(\frac{n}{2})} \quad \Longrightarrow \quad f_{2n+2}(u) = \frac{\frac{1}{2}\left(\frac{u}{2}\right)^{n} e^{-\frac{u}{2}}}{n!}$$

Für die obere Konfidenzgrenze lautet die Behauptung dann:

$$\alpha = 1 - \int_{-\infty}^{2\mu_o} \frac{\frac{1}{2}\left(\frac{u}{2}\right)^{n} e^{-\frac{u}{2}}}{n!}\,\mathrm{d}u = \int_{2\mu_o}^{+\infty} \frac{\frac{1}{2}\left(\frac{u}{2}\right)^{n} e^{-\frac{u}{2}}}{n!}\,\mathrm{d}u = \int_{\mu_o}^{+\infty} \frac{u^n e^{-u}}{n!}\,\mathrm{d}u$$

wobei im letzten Schritt die Substitution $\frac{u}{2} \to u$ durchgeführt wurde.
Durch partielle Integration kann man zeigen, dass gilt

$$\int u^n e^{au}\,\mathrm{d}u = \frac{e^{au}}{a}\left[u^n + \sum_{k=1}^{n}\left(\frac{-1}{a}\right)^k n(n-1)\ldots(n-k+1)\,u^{n-k}\right]$$

Mit $a = -1$ folgt somit

$$\alpha = \frac{-e^{-u}}{n!}\left[u^n + \sum_{k=1}^{n}\frac{n!}{(n-k)!}u^{n-k}\right]\Bigg|_{\mu_o}^{+\infty} = e^{-\mu_o}\left[\frac{\mu_o^n}{n!} + \sum_{k=1}^{n}\frac{\mu_o^{n-k}}{(n-k)!}\right] \quad \text{setze } r = n-k$$

$$= e^{-\mu_o}\left[\frac{\mu_o^n}{n!} + \sum_{r=0}^{n-1}\frac{\mu_o^r}{r!}\right] = e^{-\mu_o}\sum_{r=0}^{n}\frac{\mu_o^r}{r!} = \sum_{r=0}^{n}\frac{e^{-\mu_o}\mu_o^r}{r!}$$

Der Beweis für die untere Konfidenzgrenze ist analog

Universität Hamburg

Konfidenzgrenzen für Poisson-verteilte Messgrößen

Wichtiges Zahlenbeispiel:
In einem Zählexperiment wird die Beobachtung $n = 0$ gemacht, z.B. kein radioaktiver Zerfall, kein Higgs-Ereignis, etc. Anzuwenden ist hier natürlich die Poisson-Verteilung:

$$P(r;\mu) = \frac{e^{-\mu}\mu^r}{r!}$$

Die untere Grenze des Konfidenzintervalls ist natürlich $\mu_u = 0$, denn jeder Wert $r < 0$ ist unphysikalisch.
Deshalb gibt man z.B. 95% Konfidenzintervalle so an, dass die Wahrscheinlichkeit $\alpha = 0.05$ ist, dass der wahre Wert $\hat{\mu}$ größer ist als μ_o, also $P(\hat{\mu} \geq \mu_o) \leq 5\%$
Berechne also

$$\alpha \;=\; 0.05 \;=\; \sum_{r=0}^{n} \frac{e^{-\mu_o}\mu_o^r}{r!} \qquad \text{mit } n = 0$$

$$= \frac{e^{-\mu_o}\mu_o^0}{0!} \;=\; e^{-\mu_o} \qquad \Longrightarrow \mu_o = -\ln 0.05 \;=\; 2.996... \;\approx\; 3$$

Das bedeutet, dass der wahre Wert $\hat{\mu}$ mit 95% Wahrscheinlichkeit im Intervall [0...3] liegt. Ein solches Ergebnis gibt man dann an als (CL = Confidence Level)

$$\hat{\mu} < 3 \quad (95\%\ \mathrm{CL})$$

oder mit den entsprechenden Faktoren (Luminosität) umgerechnet

$$\sigma_{\mathrm{Higgs}} < 1.2345\ \mathrm{nb} \quad (95\%\ \mathrm{CL})$$