Blatt05

December 4, 2023

---

1 Rechnernutzung in der Physik

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik
Prof. G. Quast, Prof. M. Steinhauser
Dr. A. Mildenberger, Dr. Th. Chwalek
Ilias Seite zum Kurs
WS 2023/24 – Blatt 05
Abgabe: Montag 11.12.2023 bzw. Dienstag 12.12.2023

Auf dem letzten Übungsblatt im Statistik-Block sollen viele der Konzepte und Techniken aus dem Block der Veranstaltung "Rechnernutzung in der Physik" in einer Projektaufgabe angewendet und vertieft werden. In der Projektaufgabe sollen mit Hilfe eines Ensemble-Tests aus vielen Pseudo-experimenten (auch:Toy-MC) die Eigenschaften einer (einfachen) Parameteranpassung an einem nahezu realistischen Beispiel untersucht werden. Zum Abschluss des Statistik-Blocks erwarten wir, dass Sie aus den Beispielen in Vorlesung und Tutorium, aus den weiteren hier angegebenen Tutorials und aus der Online-Dokumentation der verwendeten Software-Werkzeuge eigenständig Elemente zur Lösung der Aufgaben zusammenstellen.

Das Blatt darf innerhalb einer Tutoriumsgruppe zu maximal dritt bearbeitet werden. Die Abfrage erfolgt jedoch gleichmäßig. Jede teilnehmende Person muss in der Lage sein jeden Programmschritt vollständig erklären zu können. **Anmerkung:** die Bearbeitung dieses Blattes ist **verpflichtend** für den Erhalt des Übungsscheins.

Gemessen werden soll die Lebensdauern von Teilchen, die nach Durchgang durch einen Detektor in einem Absorber gestoppt werden. Die Zerfälle werden über die Zerfallsprodukte wieder vom selben Detektor registriert. Die theoretische Lebensdauer der Teilchen beträgt $ =$2µs, dies entspricht in etwa der Lebensdauer des Myons. Wegen der Überlappung der Detektorsignale können Lebensdauern kleiner als $t_{min}=$1µs nicht zuverlässig gemessen werden. Die Messelektronik ist nur bis zum Zeitpunkt $t_{max}=$10µs aktiv. Die Zahl der so registrierten Zerfälle ist mit $N = 50$ registrierten Ereignissen nur sehr klein, so dass in einem ungebinner Maximum-Likelihood-Fit (d.h. alle Datenpunkte werden in der Likelihood-Funktion berücksichtigt, nicht nur die Einträge in Bins eines Histogramms) eine Exponentialfunktion an die im Intervall $[t_{\min}, t_{\max}]$ gemessenen Lebensdauern angepasst werden soll.

Es stellen sich zwei Fragen: 1. Ist der Schätzwert für die Lebensdauer $\hat{\tau}$ *erwartungstreu* (auch: unverzerrt, engl.: unbiased)? 2. Wie gut ist die *Abdeckung* (engl.: coverage) des Konfidenzintervalls

für $\hat{\tau}$?

Zur Erläuterung: Die Unsicherheit auf die gemessene Lebensdauer soll aus einem Scan der negativen Log-Likelihood (NLL) gewonnen werden. Das Intervall $[\hat{\tau} - \Delta^-, \hat{\tau} + \Delta^+]$ mit den asymmetrischen Unsicherheiten Δ^+ und Δ^- heißt Konfidenzintervall. Zur Erinnerung: in der frequentistischen Statistik liegt der wahre Wert eines Parameters in einem Bruchteil α aller aus Daten konstruierten Konfidenzintervalle. Eine häufige Wahl ist $\alpha \approx 0,683$, dies entspricht einer Standardabweichung der Gaußverteilung. Ein wichtiger Test bei der Bestimmung von Konfidenzintervallen mithilfe von Ensemble-Tests ist die Überprüfung der Abdeckung, also die Frage, ob der wahre Wert wirklich in in einem Bruchteil α aller Konfidenzintervalle liegt.

Hinweise: Lesen Sie alle Aufgaben zuerst durch, planen Sie die Programmstruktur sorgfältig und achten Sie auf einen modularen Aufbau der einzelen Teile, damit Sie die notwendigen Schritte in einer Schleife ausführen und die gesamte Studien ggf. mit unterschiedlichen Parametereinstellungen wiederholen können. In den Codebeispielen zur Vorlesung finden Sie eine Reihe von Lösungsansätzen. Auch in den Tutorials zu Likelihood-Anpassungen in Jupyter *negLogLFits.ipynb* und in Beispiel-Python-Skripten zu PhyPraKit wie *PhyPraKit/examples/toyMC_Fit.py* von Prof. Quast finden Sie eine Vorlage zum Aufsetzen der Programmstruktur. Wenn Sie die Maximum-Likelihood-Anpassung nicht selbst programmieren möchten, können Sie die Funktion *PhyPraKit.phyFit.mFit()* dazu verwenden, die in der PhyPraKit-Dokumentation beschrieben ist.

[ ]:
```
# notwendige Importe: numpy, matplotlib.pyplot, Fit Pakete
import numpy as np
import matplotlib.pyplot as plt

# ->  eigenen Code hier einfügen
```

1.1 Aufgaben

1) Schreiben Sie eine Funktion, die 50 exponentiell verteilte Zufallszahlen im sensitiven Detektorintervall $[t_{\text{min}}, t_{\text{max}}]$ erzeugt.

[ ]:
```
# Parameter für die Ausführung der Studie als globale Variable
Nexp = 3000
N = 50
tau = 2.
tmin = 1.
tmax = 10.

npar = 1            # Zahl der angepassten Parameter
pnams = ["tau"]     # Liste mit Namen der Parameter
true_vals= np.array([tau, tmin, tmax]) # die "Wahren Werte"
```

[ ]:
```
# definieren Sie hier die Verteilungsdichte
## def exponentialDecayPDF(t, tau= tau, tmin=tmin, tmax=tmax):

# ->  eigenen Code hier einfügen
```

```
[ ]: # Funktion zur Erzeugung der Daten
     ## def generateExpData(N, tau, tmin, tmax):

     # ->  eigenen Code hier einfügen
```

2) Setzen Sie eine Anpassung mit einer ungebinnten negativen log-Likelihood-Funktion auf. Dazu können Sie das Beispiel aus der Vorlesung und aus einem früheren Übungsblatt nutzen. Alternativ können Sie die Funktion `mfit()` aus dem Paket `PhyPrakit.phyFit` nutzen.

```
[ ]: # Schleife zur wiederholten Ausführung von Datenerzeugung, Anpassung und zum␣
      ↪Speichern der Ergebnisse
     ## def MC_loop():

     # ->  eigenen Code hier einfügen
```

3) Führen Sie 1) und 2) in einer Schleife aus; das Ziel ist es, 3000 Pseudoexperimente zu simulieren. Beginnen Sie zum Testen des Codes aber zunächst mit einer kleineren Anzahl! Denken Sie daran, in der Schleife die zur Bestimmung von Erwartungstreue und Abdeckung notwendigen Daten in einem Array zu speichern.

```
[ ]: # run MC loop
     # ->  eigenen Code hier einfügen
```

4) Analysieren Sie die in der Monte Carlo-Schleife gewonnenen Daten und geben Sie Erwartungstreue und Abdeckung für die geschätzte Lebensdauer $\hat{\tau}$ und deren Unsicherheitsintervall $[\hat{\tau} - \Delta^-, \hat{\tau} + \Delta^+]$ an. Denken Sie daran, dass Sie auch die statistische Signifikanz der so bestimmten Eigenschaften sicherstellen müssen. Schätzen Sie daher die Unsicherheiten der ermittelten Erwartungstreue und Abdeckung ab.

```
[ ]: # Berechnung und Ausgabe der Ergebnisse

     # ->  eigenen Code hier einfügen
```

5) Bewerten Sie Ihre Ergebnisse:
   a) Wird eine statistisch signifikante Verzerrung nachgewiesen?

   b) Ist sie ggf. im Bezug auf die Unsicherheiten der Einzelmessungen relevant?

   c) Benötigen Sie auf Grund der festgestellten Unter- oder Überabdeckung eine Korrektur des für eine Einzelmessung relevanten Konfidenzintervalls ?

Antworten *Hier Antworten eintippen*

6) **Freiwillig**: In der Vorlesung haben Sie die "Bootstrap-Methode" kennen gelernt, um die Eigenschaften von Funtionen von Zufallszahlen zu untersuchen. Diese Methode lässt sich auch auf das hier behandelte Problem anwenden. Schreiben sie dazu eine Funktion, die aus einem einzigen initialen, mit der unter 1) geschriebenen Funktion erzeugten Daten neue Datensätze durch "Ziehen mit Zurücklegen" erzeugt. Dazu können Sie die Methode `rng.choice(data, size=N)` verwenden. Führen Sie nun wiederum die Schritte 2) - 4) durch

und vergeleichen Sie das Ergebnis. Sie sollten diese Studie mit mehreren verschiedenen initialen Datensätzen wiederholen, da im Einzelfall, abhängig vom initialen Datensazt, größere Fluktuationen auftreten.

[ ]:
```
# Funktion zum Re-Sampling der Daten (für Bootstrap)
#def resampleData(N, dT0):
```

[ ]:
```
# Einen Datensatz erzeugen
# run MC loop

# Daten mit Resampling ("bootstrapping")
```