Rechnernutzung in der Physik

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. M. Giffels, Dr. R. Wolf
Dr. A. Mildenberger

WS2015/16 – Blatt 11

`http://comp.physik.kit.edu` Prog.: Di., 19.01.2015 / Ausarb.: Fr., 22.01.2016

Verteilungsdichten und Monte-Carlo-Methode

Aufgabe 21: Zufallszahlen mit beliebiger Verteilungsdichte — Ausarbeitung

Ausgehend von im Intervall $[0, 1)$ gleichverteilten Zufallszahlen (`gRandom->Rndm()` in `ROOT` oder `numpy.random.rand()` in `Python`), sollen Zufallszahlen erzeugt werden, die anderen Verteilungsdichten folgen.
Hinweis: Zu dieser Aufgabe gibt es wieder Vorlagen, `PDFs-pyroot.py` bzw. `PDFs.C`[1]. Da nur die Histogramm-Funktionen aus `ROOT` benötigt werden, dient die `pyroot`-Vorlage auch für Lösungen in `Python` ohne Verwendung von `ROOT`.

a) Streuwinkelverteilung in e^+e^--Streuung

Erzeugen Sie 10'000 Zufallszahlen mit der Verwerfungsmethode gemäß der Verteilungsdichte

$$f(x) = \frac{3}{8}(1 + x^2) \quad (-1 \leq x \leq 1)$$

und füllen Sie die Zufallszahlen in ein Histogramm.
Schreiben Sie eine allgemeine C++- oder Python-Funktion zur Implementierung der Verwerfungsmethode.

b) Cauchy-Verteilung

Verwenden Sie die Transformationsmethode und füllen Sie ein Histogramm mit 10'000 Zufallszahlen, die einer Cauchy-Verteilungsdichte[2] folgen, die gegeben ist durch

$$f(x) = \frac{1}{\pi}\frac{1}{1 + x^2}\,.$$

c) Majoranten-Methode („Importance Sampling")

Füllen Sie ein Histogramm mit 10'000 Zufallszahlen, die gemäß

$$f(x) = \frac{5}{2}\,\sin^2(\pi x)\,\exp(-x),\ 0 \leq x \leq \infty$$

verteilt sind und die wir im Intervall $[0, 2\pi)$ betrachten wollen. Verwenden Sie exponentiell verteilte, mit der Transformationsmethode erzeugte Zufallszahlen als Majorante, $m(x) \propto \exp(-x)$. Sie können die im Aufgabenteil a) erstellte Funktion wiederverwenden bzw. geeignet erweitern, um statt der gleichverteilten Zufallszahlen exponentiell verteilte zu verwenden.

---

[1] Falls Sie in C++ arbeiten und Ihr C++-Makro übersetzen wollen, verwenden Sie „`make SOURCE=PDFs`".
[2] Bei Physikern ist die Cauchy-Verteilung auch als Lorentz- oder Breit-Wigner-Verteilung bekannt.

d) Zufallszahlen gemäß empirischer Verteilung aus Histogramm

In der Datei „`elefant.dat`“ finden Sie die Einträge in 64 Bins einer vorgegebenen Verteilung. Erzeugen und histogrammieren Sie 10'000 Zufallszahlen, die dieser Verteilung folgen. Beispielcode zum Einlesen der Datei in `C++` oder `Python` ist in den Vorlagen enthalten.

Stichpunkte zur Ausarbeitung:

- Beschreiben Sie kurz die Verwerfungsmethode.
- Drucken Sie den kommentierten Code Ihrer Implementierung der Verwerfungsmethode aus.
- Erläutern Sie die Transformationsmethode und geben sie die gewählte Transformationsfunktion für die Cauchy-Verteilung aus Aufgabenteil b) an.
- Fügen Sie einen Ausdruck eines Histogramms aus jeder der vier Teilaufgaben bei.

Aufgabe 22: Zufällige Ereignisse und Poisson-Verteilung Programmtestat

In dieser Aufgabe soll eine Folge von Zeitpunkten erzeugt werden, zu denen ein Ereignis stattfindet, wobei die Eintreffwahrscheinlichkeit pro Zeitintervall konstant ist. Diese Voraussetzungen führen zu einer Poisson-Verteilung der Anzahl von Ereignissen in einem festen Zeitintervall. Die Anwendungsbeispiele dazu sind in der Physik sehr vielfältig, von der Emission eines Photons aus einer Einzel-Photon-Quelle, dem Zerfall eines Kerns in einem radioaktiven Präparat, oder auch der Produktion von Higgs-Bosonen am Large-Hadron-Collider.
Zum Verständnis solcher Poisson-Prozesse gehen wir zunächst von der Verteilung der sogenannten „Wartezeit“ aus, d.h. der zwischen dem Auftreten von zwei Zufallsereignissen verstrichenen Zeit.

a) Verteilung der Wartezeiten

Die Verteilungsdichte der Wartezeiten, t_w, zwischen zwei Ereignissen, die in der Zeit gleichverteilt sind, ist gegeben durch $f(t_w) = \frac{1}{\overline{t_w}} \exp\left(-t_w/\overline{t_w}\right)$, wobei $\overline{t_w}$ die mittlere Wartezeit ist und dem Kehrwert der Ereignisrate R entspricht. Also gilt auch $f(t_w) = R \exp\left(-R\, t_w\right)$ (zur Begründung s. Anhang).
Nutzen Sie diese Verteilung der Wartezeiten und schreiben Sie eine Funktion, die, ausgehend von $t_0 = 0$, eine Folge von N=50'000 Zeitpunkten t_i bestimmt, zu denen solche Ereignisse auftreten. Der Einfachheit halber sei die Rate $R = 1/$Zeiteinheit (ZE). Histogrammieren Sie zur Kontrolle die Zeiten zwischen aufeinanderfolgenden Ereignissen und überlagern Sie zum Vergleich die Exponentialfunktion.

b) Überprüfung der zeitlichen Verteilung

Tragen Sie die im Aufgabenteil a) erzeugten Zeitpunkte t_i in ein Histogramm mit 500 Bins ein, das den Bereich von t_0 bis t_{N-1} abdeckt. Welche Verteilung erhalten Sie? Welche Ereignisrate/Bin lesen Sie ab?

c) Verteilung der Bin-Inhalte

Stellen Sie die Bin-Inhalte des Histograms als Häufigkeitsverteilung dar. Vergleichen Sie mit der Poisson-Verteilung.

d) Detektoreffizienz und gemessene Rate

Als letzer Schritt soll nun das Ansprechverhalten eines Detektors und dessen Einfluss auf die gemessene Ereignisrate untersucht werden. Nehmen Sie dazu an, ein Detektor habe nach dem Registrieren eines Ereignisses zum Zeitpunkt t_i eine Totzeit, die dazu führt, dass die Nachweiseffizienz nach der Funktion $\epsilon(t) = 5./\mathrm{ZE}\,(t - t_i)$ für $(t - t_i) < 0.2\,\mathrm{ZE}$ von Null bis Eins ansteigt. Welche Ereignisrate misst der Detektor? Wie groß ist der auf Grund der Totzeit notwendige Korrekturfaktor auf die gemessene Rate?

ANHANG: Hinweise zu Aufgabe 22

Verteilung der Wartezeiten

Man denke sich die Zeit t_w in n kleine, gleich große Intervalle $\Delta t_i = t_w/n$ zerlegt. Die Wahrscheinlichkeit, ein Ereignis in einem solchen Intervall zu beobachten, ist gegeben durch $p = R\,t_w\,/n$. Ein Zerfall nach der Zeit t_w bedeutet, dass in den Intervallen davor kein Ereignis beobachtet wurde, d.h. $p(t_w) = (1 - Rt_w/n)^n$, für $n \to \infty$ also $p(t_w) = \exp(-Rt_w)$.

Hinweise zur Programmierung

Die Nachweiswahrscheinlichkeit im Detektor von Aufg. 22 d) hängt vom Zeitpunkt des letzten Ereignisses ab, der dazu gespeichert werden muss. In `C++` kann man dazu eine statische Variable verwenden (z.B. `static float tlast`), in `Python` eine globale Variable, die außerhalb einer Funktion definiert ist und innerhalb einer Funktion als `global tlast` deklariert werden muss. Obwohl für diese Übungsaufgabe akzeptabel, sind in längeren Programmen solche statischen bzw. globalen Variablen sehr problematisch. Die saubere Lösung ist es, den Detektor und seine Eigenschaften als eigene Klasse zu definieren, in `Python` z.B.

```
# Class defining detector properties
class SimpleDetector:
  """a class defining a simple detector with dead time"""
  def __init__(self,tau=0.2):
    self.tlast=-1.     # initialize variable storing argument of last call
    self.tfulleff=tau  # time after which full efficiency is regained

  def Efficiency(self,dt):
   # function defining detector characteristics,
   #   here the efficiency depending on time since last hit
    # ...
    return eff

  def Signal(self,t):
   # function returning True if signal at time t is detected
   #   uses random sampling of function "Efficiency()"
    # ...
    #SigSeen= ... # True/False
    return SigSeen
```

Eine Instanz der Klasse kann dann alle notwendigen Daten zum simulierten Detektor und zur Berechnung der Nachweiseffizienz speichern und die notwendigen Funktionen bereit stellen.

Hinweis: Mit dem Rechnernamen `fphctssh.physik.uni-karlsruhe.de` können Sie von überall aus mittels ssh/scp Programm per Netzwerk auf einen Poolrechner zugreifen.
