Simulation des Rutherford-Geiger-Experimentes

Alexander Adam Friedrich Jahns

14. Februar 2020

Zusammenfassung

In dem folgenden Protokoll wird untersucht inwiefern Radioaktiver Zerfall als eine **Possion-Verteilung** modelliert werden kann. Dies geschieht unter Verwendung von Python (*Distribution: Anaconda*). Inspiriert wurde dies durch die Vorlesung : Einführung in die Statistik und angewandte Informatik.

Dozent: Dr. Dominic Hirschbühl
Tutor: Gunnar Jäkel

Inhaltsverzeichnis

1 Einleitung

Wir betrachten einen radioaktiven Kern der mittleren Lebensdauer τ und beobachten ihn per einen Zeitraum $T \ll \tau$. Die Wahrscheinlichkeit, dass er irgendwann in diesem Zeitraum zerfällt, ist dann $W \ll 1$. Wir unterteilen nun die Beobachtungszeit T in n_{int} Zeitintervalle der Länge t, so dass $T = n_{\text{int}} \times t$. Dann ist die Wahrscheinlichkeit, daß der Kern in einem bestimmten Intervall zerfällt, $p \approx W/n$. Wir beobachten nun eine radioaktive Quelle, die N Kerne enthält, die unabhängig voneinander zerfallen, über den Gesamtzeitraum T, und registrieren dabei a_{i} Zerfälle im Zeitintervall $i (i = 1...n_{\text{int}})$.

1.1 Aufgaben

1. Schreiben Sie ein Programm das die Zerfälle des radioaktiven Kerns mit den Eingabeparametern N, n_{int} simuliert. Hinweis: Verwenden Sie gleich verteilte Zufallszahlen im Intervall $[0, T]$.

2. Erstellen Sie ein Histogramm das die Anzahl der Intervall mit k(k = 0, 1, 2, ...) Zerfällen enthält.

3. Untersuchen Sie verschiedene Intervallgrössen und zeigen Sie, dass Sie für $N \to \infty$ eine **Possion-Verteilung** erhalten (halten Sie dabei $\lambda = N/n_{\text{int}}$ fest).

2 Programm

Das Programm ist in 4 Hauptfunktionen unterteilt. Es erfüllt alle 3 Forderungen der Aufgaben wobei diese ebenfalls auf einander aufbauen. Die erste Aufgabe, das generieren von Daten, wird von der Funktion `gen_data_mp()` übernommen welche selber in eine externe Python Datei ausgelagert wurde. Das erstellen einzelner Plots, die zweite Aufgabe, wird von `gen_hist()` durchgeführt und beruht auf `gen_data_mp()` als auch `draw_hist` welche aus gegeben Daten Histogramme erstellt. Zuletzt wird die Hauptaufgabe, das überprüfen der **Possion-Verteilung**, von der Funktion `gen_hist_lam()` verrichtet.

2.1 gen_data_mp

Diese Funktion simuliert das Verhalten von N Kernen mit der mittleren Lebensdauer τ in einem Beobachtungszeitraum T welcher wiederum in n Zeitintervalle unterteilt wurde. Die Funktion bedient sich des Multiprocessings damit sie in der Lage ist das Erstellen von Zufallszahlen auf die Anzahl der verfügbaren logischen Kerne zu verteilen. Da das Multiprocessings Modul nicht in interaktiven Python Umgebungen wie z.B **Jupyter Notebook** musste diese Funktion in ihre eigene Python Umgebung erstellt werden.

2.2 draw_hist

`draw_hist` dient dazu ein Histogramm mit den simulierten Werten zu erstellen und diese mit einer idealen **Possion-Verteilung** zu vergleichen. Sie tut dies unter Verwendung der Module *Scipy* und *Matplotlib*. *Matplotlib* wird verwendet zum erstellen des Histogramms während *Scipy* für die **Possion-Verteilung** als auch die Berechnung des χ^2 und dem $p - Value$ benutzt wird.

2.3 gen_hist

In `gen_hist()` werden die Funktionen `gen_data_mp()` und `draw_hist` zusammengesetzt. Sie erstellt zu beginn unter Verwendung von angegebenen Daten eine Simulation und überreicht diese dann weiter zum Zeichen.

2.4 gen_hist_lam

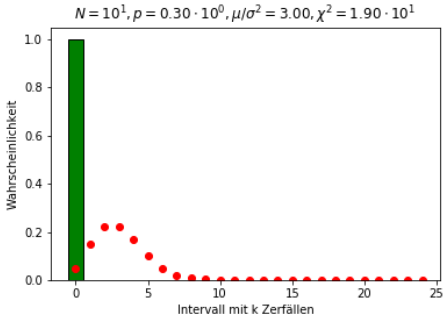
`gen_hist_lam()` ist die Hauptfunktion. Sie bedient sich den vorherigen Funktion um mehrere Graphen zu erstellen welche dir Auswirkung darstellen, wenn $N \longrightarrow \infty \, und \, p \longrightarrow 0$ geht. Die Daten können ebenfalls abgespeichert werden um späteres betrachten der Plots zu ermöglichen oder gar die puren Ergebnisse weiter auszuwerten.

Genauere Beschreibung der Funktionen und ihrer Verhalten lassen sich in den jeweiligen Doc-Strings finden und der Kommentierung des Programms.

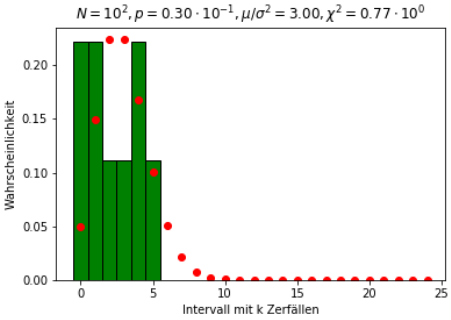
3 Ergebnisse der Simulation

Anzahl der Kerne N	Wahrscheinlichkeit p	χ^2	$p - Value$
10^1	$0,30 \times 10^0$	$1,90 \times 10^1$	$0,00192080$
10^2	$0,30 \times 10^{-1}$	$0,77 \times 10^0$	$0,97920338$
10^3	$0,30 \times 10^{-2}$	$0,57 \times 10^{-1}$	$0,99999612$
10^4	$0,30 \times 10^{-3}$	$0,34 \times 10^{-2}$	$1,00000000$
10^5	$0,30 \times 10^{-4}$	$0,79 \times 10^{-3}$	$1,00000000$
10^6	$0,30 \times 10^{-5}$	$0,95 \times 10^{-4}$	$1,00000000$
10^7	$0,30 \times 10^{-6}$	$0,14 \times 10^{-4}$	$1,00000000$
10^8	$0,30 \times 10^{-7}$	$0,23 \times 10^{-5}$	$1,00000000$
10^9	$0,30 \times 10^{-8}$	$0,13 \times 10^{-6}$	$1,00000000$

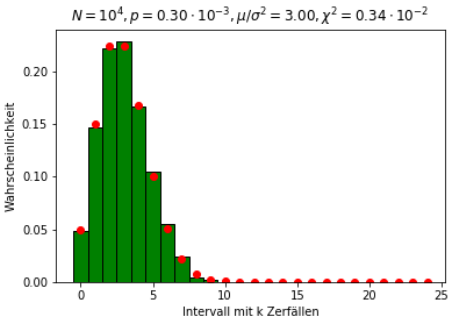
Tabelle 1: Ergebnisse im Rahmen der Aufgabe 3; $\mu = 3$



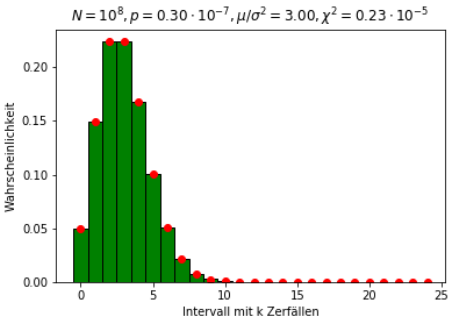
(a) $N = 10$



(b) $N = 100$



(c) $N = 10000$



(d) $N = 10000000$

4 Auswertung

Wie man den Histogrammen 1a bis 1d klar erkennen kann, steigt mit dem Versuchsumfang, also der Anzahl N der Kerne, auch die Annäherung des Histogramms an die Werte der Poissonverteilung mit ihrem jeweiligen Wert μ (in diesem Fall $\mu = 3$).

$$P_\mu(k) = \frac{\mu^k}{k!} \times e^{-\mu} \tag{1}$$

(Pos)

Konkret bedeutet dies, dass für $N \longrightarrow \infty$ *und* $p \longrightarrow 0$ das Histogramm zur Anzahl der Zerfälle in den jeweiligen Abschnitten aus T einer Poissonverteilung (1) abhängig von $\mu = 3$ entspricht. Den Beweis dieser Aussage stellt sowohl der χ^2 Test, welcher in diesem Fall gegen $Null$ konvergiert, als auch das $p - value$ welches in diesem Fall gegen 1 konvergiert da.

Literatur

poissonverteilung.de, https://www.poissonverteilung.de/poissonverteilung.html.