# PAP 2.2
# Versuch 252: "Aktivierung mit thermischen Neutronen"

Jonas Schweer

15. Januar 2025

## Inhaltsverzeichnis

**1 Einleitung** **2**
- 1.1 Ziel des Versuchs . . . 2
- 1.2 Versuchsaufbau . . . 2
- 1.3 Physikalische Grundlagen . . . 2
  - 1.3.1 Erzeugen eines radioaktiven Kerns . . . 2
  - 1.3.2 Aktivierung eines Stoffes . . . 3
  - 1.3.3 Zerfall . . . 3
  - 1.3.4 Kompensation für die Untergrundaktivität . . . 4
  - 1.3.5 Aktivität der Silberisotope . . . 4

**2 Messprotokoll** **4**

**3 Auswertung** **6**
- 3.1 Auswertung Aufgabe 1 . . . 6
- 3.2 Auswertung Aufgabe 2 . . . 9

**4 Zusammenfassung und Diskussion** **11**

**5 Anhang** **12**

# 1 Einleitung

## 1.1 Ziel des Versuchs

In diesem Versuch wollen wir durch einige Langzeitmessungen die Halbwertszeiten von drei Verschiedenen Isotopen bestimmen. Zuerst schauen wir uns zwei Silberisotope ($^{108}$Ag und $^{110}$Ag) an, und daraufhin ein Indiumisotop ($^{116}$In).

## 1.2 Versuchsaufbau

Der Versuchsaufbau besteht aus einem Geiger-Müller-Zählrohr, welches direkt von einer Präparate-Halterung angebracht ist. In diese Halterung kann dann das jeweilige Plättchen der aktivierten Isotope eingespannt werden, und strahlt somit direkt in das Zählrohr. Dieses ist mit einem Betriebsgerät verbunden, mit dem man das Geiger-Müller-Zählrohr bedienen kann. (Wir benutzen es um die Betriebsspannung einzustellen.) Das Betriebsgerät wiederum ist mit einem externen Impulszähler verbunden, welcher die registrierten Zerfälle an einen nebenstehenden Computer weitergibt. Dieser kann nun die Ergebnisse in ein Histogramm auftragen und uns letztendlich auch die Rohdaten ausgeben, um so die Zerfallsfunktion zu bestimmen. Durch diese erhalten wir die Zerfallskonstante.

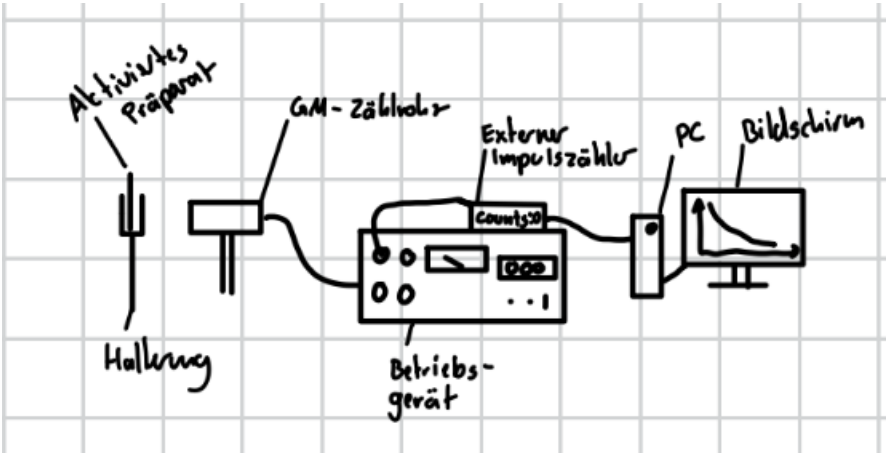



Abbildung 1: Messaufbau

## 1.3 Physikalische Grundlagen

### 1.3.1 Erzeugen eines radioaktiven Kerns

Um diesen Versuch durchführen zu können, müssen wir die Präparate genügend aktivieren. Um das zu schaffen, also um ihren Atomkern in einen instabilen Zustand zu bringen, eignen sich Neutronen besonders gut. Dadurch, dass Neutronen ungeladen sind, Wechselwirken sie weder mit dem Atomkern, noch mit den Elektronen um den Kern herum durch Coulomb-Kräfte. Um zu ermöglichen, dass die Neutronen von dem Atomkern eingefangen werden können, muss ihre Energie dennoch gering genug sein. Wir sprechen von 'thermischer Energie' oder niedriger.

In unserem Versuch stellen wir die Neuronen gemäß folgender Reaktion her:

$$^{9}\mathrm{Be} + \alpha \longrightarrow {}^{12}\mathrm{C} + n \tag{1}$$

Dabei steht $\alpha$ für ein Alpha-Teilchen und $n$ ist das freigegebene Neutron. Da die Energie von diesem allerdings noch zu hoch ist, um von den Atomkernen eingefangen zu werden, wird eine Paraffinummantelung genutzt. Die Neutronen prallen gegen die Wasserstoffatome der Ummantelung und werden so stark abgebremst (elastische Stöße zweier Objekte mit ähnlicher Masse → im Mittel geht die Hälfte der Energie verloren). Wenn die Energie niedrig genug ist, wird also das Neutron von dem Kern eingefangen und je nach Isotop kann sich ein aktivierter Kern bilden. Wird zum Beispiel $^{115}$In bestrahlt, so bildet sich der $\beta$-Strahler $^{116}$In im Grundzustand, sowie der metastabile Zustand $^{116m}$In. Die beiden sind Isomere, welche beide in $^{116}$Sn zerfallen, was wiederum ein stabiles Isotop ist. Die metastabilen Zustände zerfallen jedoch recht schnell wieder, weshalb in unserem Versuch hauptsächlich der Grundzustand betrachtet wird. Bei der Bestrahlung von unseren Silberpräparaten entstehen die zwei instabilen Isotope $^{108}$Ag und $^{110}$Ag, welche in $^{108}$Cd und $^{108}$Cd zerfallen. Durch die unterschiedlichen Halbwertszeiten der beiden Isotope kann durch eine Anpassung der Aktivierungsdauer das Verhältnis zwischen ihnen Angepasst werden, je nach dem welches man genauer untersuchen möchte.

### 1.3.2 Aktivierung eines Stoffes

Die Aktivierung beschreibt den Prozess, bei dem aktiv neue radioaktive Kerne erzeugt werden. Da allerdings während dem Aktivieren bereits Kerne gemäß dem Zerfallsgesetz in einen stabilen Zustand 'zerfallen', kann man die Aktivität $A$(also die Anzahl der Zerfälle pro Sekunde) mit folgender Formel beschreiben:

$$A(t) = A_{\infty}(1 - \exp\left[-\lambda t\right]) \tag{2}$$

Dabei ist $A$ eine Funktion in Abhängigkeit der Bestrahlungsdauer $t$. $A_{\infty}$ ist die Aktivität, bei der die Rate, mit der neue Kerne aktiviert werden mit der Rate, in der die Kerne zerfallen, übereinstimmt und $\lambda$ die Zerfallskonstante.

### 1.3.3 Zerfall

Der Zerfall ist die Rate, in der instabile (aktivierte) Kerne wieder in einen stabilen Zustand übergehen und dabei Energie in Form von Strahlung abgeben. Dieser kann nach dem radioaktiven Zerfallsgesetz modelliert werden:

$$A(t) = A_0 \exp\left[-\lambda t\right] \tag{3}$$

Man sieht also, dass die Aktivität wieder abnimmt und dabei einer Exponentialfunktion folgt. Ausgehend von dieser Formel können wir dann auch die Halbwertszeit bestimmen als:

$$T_{1/2} = \frac{\ln 2}{\lambda} \tag{4}$$

### 1.3.4 Kompensation für die Untergrundaktivität

Da es trotz Abwesenheit von radioaktiven Substanzen im Geiger-Müller-Zählrohr zur Registrierung von Zerfällen kommen kann, muss man für diese Kompensieren, wenn man eine präzise Auswertung erhalten möchte. So müssen wir bei allen $A(t)$ noch ein $A_{ug}$ addieren, welches die Untergrundaktivität ist. So ist gegeben, dass $A(t)$ nicht gegen null konvergiert für $t \to \infty$, sondern eben gegen die gemessene Untergrundaktivität. So folgt für das Zerfallsgesetz:

$$A(t) = A_0 \exp\left[-\lambda t\right] + A_{ug} \tag{5}$$

### 1.3.5 Aktivität der Silberisotope

Da wir es bei den Silberisotopen mit zwei verschiedenen Halbwertszeiten (und somit auch Zerfallskonstanten) zu tun haben, müssen wir die Formel für die Zerfälle zu einem bestimmten Zeitpunkt entsprechend anpassen, um zwei simultane Zerfallsprozesse zu beschreiben. Dies erreichen wir, durch simples addieren der beiden Zerfallsgesetze, wobei allerdings die Untergrundzählrate nur einmal berücksichtigt werden muss. Wir erhalten:

$$A(t) = A_1 \exp\left[-\lambda_1 t\right] + A_2 \exp\left[-\lambda_2 t\right] + A_{ug} \tag{6}$$

schön!

# 2 Messprotokoll

# Messprotokoll: Aktivierung mit thermischen Neutronen

05.11.24
Jonas Schwer

Messgeräte: - Geiger-Müller Zählrohr
- Impulszähler
- Neutronenquelle
- Präparatehalterung
- Indium und Silberbleche

## Messung der Untergrundzählrate

Wir legen am Geiger-Müller-Zähler eine Spannung von (540±5)V an. Zunächst überprüfen wir die Untergrundzählrate. Dazu messen wir sie ungefähr für 8min und speichern die Messwerte.

## Aufgabe 1:

Wir wollen nun die Halbwertszeit von Silber messen. Dafür Aktivieren wir dieses für ca. 7min in der Neutronenquelle und bringen es anschließend vor den GM-Zähler. Nun messen wir für ca. 400s die Klicks. Wir führen die Messung (und Aktivierung) für Silber 4x durch.

## Aufgabe 2:

Wir stellen in dem Programm die Messintervalle auf 120s und messen nun mit dem Zählrohr für 50min die Klicks, die eine Aktivierte Indium-Platte erzeugt.

# 3 Auswertung

## 3.1 Auswertung Aufgabe 1

Um die Halbwertszeit von Silber zu bestimmen, müssen wir zuerst die Untergrundzählrate bestimmen, sodass wir sie herausrechnen können. Dafür laden wir unsere abgespeicherten Werte in Python und lassen uns sowohl den den Mittelwert der registrierten Ereignisse, sowie den Fehler dieser ausrechnen. Für den Fehler nutzen wir:

$$\Delta_{ug} = \frac{\sigma_{ug}}{\sqrt{N_{mess}}} \tag{7}$$

Dabei ist $\Delta_{ug}$ der $1\sigma$-Fehler, den wir für die Untergrundmessung in Betracht ziehen, $\sigma_{ug}$ die Standardabweichung der Untergrundzählrate und $N_{mess}$ die Anzahl der Messungen. Wichtig zu beachten ist außerdem, dass wir alle Werte für die Untergrundzählrate vervierfachen, da wir später die vier separaten Messungen der Silberisotope aufaddieren werden und jede von ihnen 'einmal die Untergrundzählrate mitbringt'. Wir erhalten:

$\boxed{A_{ug} = 14.8}$
$\boxed{\Delta_{ug} = 1.2}$

Nun haben wir von allen vier Messreihen jeweils die Werte, die zum gleichen Zeitintervall gehören, aufaddiert. Die benutzte Aktivität $A_t$ zu einem Zeitpunkt $t$ ergibt sich also aus der der Summe aller Einzelmessungen zum diesem Zeitpunkt:

$$A_t = \sum_{n=1}^{4} A_{n,t} \tag{8}$$

Der Fehler ergibt sich schließlich gemäß statistischer Gesetze als

$$\Delta N = \sqrt{N} \tag{9}$$

wobei $N$ die gezählten Ereignisse sind. Unsere Messung ging jeweils $400s$, wobei eine einzelne Messung $10s$ andauerte. Wir platzieren die Messung als in der Mitte dieses $10s$ Intervalls (e.g. $5s$, $15s$, $25s$, ... ,$395s$). Nun wollen wir die Funktion nach Formel (6) an unsere Messwerte 'heranfitten'. Wir nutzen dafür Python, und lassen uns sowohl die optimalen Parameter, als auch deren Fehler ausgeben. Diese sehen wie folgt aus:

$\boxed{A_1 = 354 \pm 22}$
$\boxed{\lambda_1 = (0.035 \pm 0.004)s^{-1}}$
$\boxed{A_2 = 93 \pm 16}$
$\boxed{\lambda_2 = (0.0056 \pm 0.0007)s^{-1}}$

Die Funktion plotten wir:

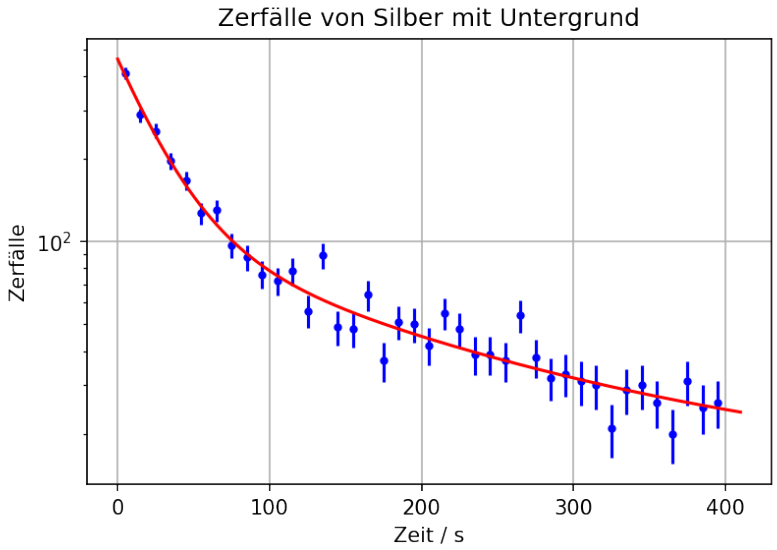



Abbildung 2: Plot der gefitteten Funktion für Silber

Damit können wir nun die Güte des Fits berechnen. Wir nutzen hierfür die Formel

$$\chi^2 = \sum_i^N \left( \frac{f(x_i) - y_i}{\Delta y_i} \right)^2 \tag{10}$$

$f(x_i)$ ist der Wert unserer gefitteten Funktion für Wert $x_i$, $y_i$ ist der tatsächlich gemessene Wert und $\Delta y_i$ der Fehler der Messung. Die reduzierte $\chi^2$-Summe kann man nun mit

$$\chi^2_{red} = \frac{\chi^2}{d} \tag{11}$$

berechnen, wobei d die Anzahl der Messwerte abzüglich der Anzahl der Fitparameter darstellt (hier also 36). $\chi^2$ sowie $\chi^2_{red}$ lauten folgendermaßen:

$\boxed{\chi^2 \approx 43.6}$
$\boxed{\chi^2_{red} \approx 1.2}$

Die Fitwahrscheinlichkeit können wir uns mit Python als

$\boxed{p_{fit} \approx 18\%}$

bestimmen lassen. Wir wiederholen die Schritte nun erneut, jedoch ziehen wir den $1\sigma$-Fehler von der Untergrundmessung ab. Gekennzeichnet wird dies durch den Index '_':

$\boxed{A_{1,-} = 356 \pm 23}$
$\boxed{\lambda_{1,-} = (0.035 \pm 0.004) s^{-1}}$
$\boxed{A_{2,-} = 92 \pm 15}$
$\boxed{\lambda_{2,-} = (0.0054 \pm 0.0007) s^{-1}}$

$\boxed{\chi^2_- \approx 44.7}$
$\boxed{\chi^2_{red,-} \approx 1.2}$
$\boxed{p_{fit,-} \approx 15\%}$

Nun wiederholen wir es, jedoch addieren wir den $1\sigma$-Fehler. Der Index lautet diesmal '$_+$'

$\boxed{A_{1,+} = 351 \pm 23}$
$\boxed{\lambda_{1,+} = (0.035 \pm 0.004)s^{-1}}$
$\boxed{A_{2,+} = 95 \pm 17}$
$\boxed{\lambda_{2,+} = (0.0059 \pm 0.0008)s^{-1}}$
$\boxed{\chi^2_+ \approx 45.1}$
$\boxed{\chi^2_{red,+} \approx 1.3}$
$\boxed{p_{fit,+} \approx 14\%}$

Um nun letztendlich einen Wert für die Zerfallskonstanten zu erhalten, bilden wir zuerst die Differenzen $|\lambda_{1,2} - \lambda_{1,2,\pm}|$. Nun berechnen wir den Mittelwert der beiden Differenzen und addieren ihn quadratisch zu dem ursprünglichen Fehler von $\lambda_{1,2}$. Konkret sieht die Formel so aus:

$$\Delta\lambda_1 = \sqrt{\sigma^2_{\lambda_1} + \left(\frac{\lambda_{Diff,1,-} + \lambda_{Diff,1,+}}{2}\right)^2} \tag{12}$$

$$\Delta\lambda_2 = \sqrt{\sigma^2_{\lambda_2} + \left(\frac{\lambda_{Diff,2,-} + \lambda_{Diff,2,+}}{2}\right)^2} \tag{13}$$

Die neuen Fehler sind:

$\boxed{\Delta\lambda_1 = 0.004s^{-1}}$
$\boxed{\Delta\lambda_2 = 0.0008s^{-1}}$

Wir rechnen nun die Halbwertszeiten gemäß Formel (4). Der Fehler ist nach Gaußscher Fehlerfortpflanzung:

$$\Delta T = \frac{\Delta\lambda}{\lambda} T \tag{14}$$

Wir erhalten also für die Halbwertszeiten:

$\boxed{T_{1/2,^{110}\mathrm{Ag}} = (19.8 \pm 2.3)s}$
$\boxed{T_{1/2,^{108}\mathrm{Ag}} = (123 \pm 17)s}$

✓

Lebensdauer $\tau$ noch berechnen?

## 3.2 Auswertung Aufgabe 2

In Aufgabe 2 gehen wir ganz analog zu Aufgabe 1 vor. Zuerst bestimmen wir die Untergrundzählrate, die für unsere Messungen passt. Das bedeutet vor allem, dass wir sie diesmal nicht vervierfachen, aber dass wir aus unseren Messdaten die ersten 12 aufsummieren. Das kommt daher, dass wir beim Messen des Untergrunds die Torzeit auf $10s$ gestellt hatten, aber bei der Indium-Messung auf $120s$. Also ist der Untergrund, der pro Einzelmessung einfällt äquivalent zu den ersten 12 Einzelmessungen des Untergrunds summiert. Der Fehler berechnet sich wieder nach Formel (7) Die verwendete Untergrundzählrate ist also:

$$\boxed{A_{ug} = 39}$$
$$\boxed{\Delta_{ug} = 0.6}$$

Nun wollen wir die Messwerte plotten. Dafür berechnen wir ihren Fehler wieder mittels Formel (8) und platzieren sie in der Mitte unserer Intervalle. Da wir diesmal in $120s$-Intervallen gemessen haben, wird der erste Messwert also bei $60s$ platziert werden, der zweite bei $180s$ und so weiter. Die Funktion, die wir an die Werte heranfitten wollen, ist wie Formel (5) aufgebaut. Den ersten Messwert werden wir jedoch beim plotten nicht mitberücksichtigen, da dieser sichtbar zu hoch geraten ist. Das ist vermutlich auf das Isomer $^{116m}$In zurückzuführen. Dieses entsteht neben dem Isotop, welches wir untersuchen wollen, ebenfalls bei der Aktivierung. Da es allerdings eine sehr geringe Halbwertszeit hat, sind die Effekte, die es auf die Messwerte hat, nach $120s$ bereits vernachlässigbar klein. Wenn wir also ab dem erst den zweiten Messwert berücksichtigen, untersuchen wir das Isotop, welches wir eigentlich wollen. Ohne Berücksichtigung der $1\sigma$-Fehler des Untergrunds erhalten wir folgenden Plot und Fitparameter:

$$\boxed{A_0 = 635 \pm 13}$$
$$\boxed{\lambda = (19.9 \pm 1.4) \cdot 10^{-5} s^{-1}}$$

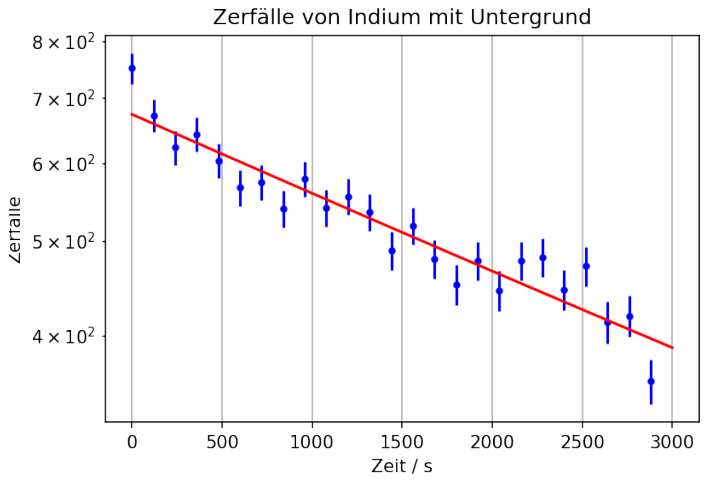



Abbildung 3: Plot der gefitteten Funktion für Silber

Die $\chi^2$-Summe und reduzierte $\chi^2$-Summe lassen sich erneut mit Formel (10) und Formel (11) berechnen und lauten wie folgt:

$$\boxed{\chi^2 \approx 35}$$
$$\boxed{\chi^2_{red} \approx 1.67}$$

Die Fitwahrscheinlichkeit ist dann:

$$\boxed{p_{fit} \approx 2.8\%}$$

Wiederholen wir nun wie in Aufgabe 1 die Rechung, aber ziehen jeweils $1\sigma$ des Untergrundfehlers vom Untergrund selbst ab. Wir erhalten:

$$\boxed{A_{0,-} = 635 \pm 13}$$
$$\boxed{\lambda_- = (19.8 \pm 1.4) \cdot 10^{-5} s^{-1}}$$
$$\boxed{\chi^2_- \approx 35.0}$$
$$\boxed{\chi^2_{red,-} \approx 1.7}$$
$$\boxed{p_{fit,-} \approx 2.8\%}$$

Addieren wir nun stattdessen den Fehler, erhalten wir:

$$\boxed{A_{0,+} = 634 \pm 13}$$
$$\boxed{\lambda_+ = (19.9 \pm 1.4) \cdot 10^{-5} s^{-1}}$$
$$\boxed{\chi^2_+ \approx 35}$$
$$\boxed{\chi^2_{red,+} \approx 1.7}$$
$$\boxed{p_{fit,+} \approx 2.8\%}$$

Mit Formel (12/13) lässt sich jetzt wieder der neue Fehler der Zerfallskonstanten berechnen. Diese lautet dann:

$$\boxed{\Delta\lambda = 1.3 \cdot 10^{-5} s^{-1}}$$

Damit berechnen wir nun gemäß Formel (4) die Halbwertszeit. Der Fehler ist wieder Formel (14). Wir erhalten:

$$\boxed{T_{1/2,^{116}\mathrm{In}} = (3490 \pm 240) s}$$



# 4 Zusammenfassung und Diskussion

Um unser Vorgehen noch einmal Zusammenzufassen haben wir zuerst eine Untergrundmessung durchgeführt, um so hoffentlich mögliche Störeffekte die auftreten könnten herauszurechnen. Anschließend haben wir vier Messungen mit aktiviertem Silber durchgeführt um so über einen Zeitraum den Abfall der Aktivität beobachten zu können. Anschließend haben wir eine sehr lange Messung mit aktiviertem Indium durchgeführt um so ebenfalls den Abfall festzuhalten. Durch betrachten der Untergrundmessung und der $1\sigma$-Grenzen dieser konnten wir so für die beiden Aktivitäten die Abfälle bestimmen. Wir haben so für die drei radioaktiven Isotope $^{108}$Ag, $^{110}$Ag, $^{116}$In sowohl die Zerfallskonstante als auch deren Halbwertszeit errechnet. Diese lauteten nach unseren Messungen:

$$\boxed{\lambda_{^{108}\mathrm{Ag}} = (0.0056 \pm 0.0008) s^{-1}}$$
$$\boxed{\lambda_{^{110}\mathrm{Ag}} = (0.035 \pm 0.004) s^{-1}}$$
$$\boxed{\lambda_{^{116}\mathrm{In}} = (19.9 \pm 1.3) \cdot 10^{-5} s^{-1}}$$
$$\boxed{T_{1/2,^{108}\mathrm{Ag}} = (123 \pm 17) s}$$
$$\boxed{T_{1/2,^{110}\mathrm{Ag}} = (19.8 \pm 2.3) s}$$
$$\boxed{T_{1/2,^{116}\mathrm{In}} = (3490 \pm 240) s}$$

Vergleichen wir die Werte der Halbwertszeit nun mit denen, welche auf der letzten Seite des Versuches im Praktikumsskript in einer Tabelle zu finden waren und somit als unsere Literaturwerte dienen werden. Nutzen wir nun die Formel

$$\sigma = \frac{|x - y|}{\sqrt{\Delta x^2 + \Delta y^2}} \tag{15}$$

um die Abweichungen zu berechnen, erhalten wir folgendes Bild:

$$\boxed{\sigma_{^{108}\mathrm{Ag}} = 1.3}$$
$$\boxed{\sigma_{^{110}\mathrm{Ag}} = 2.1}$$
$$\boxed{\sigma_{^{116}\mathrm{In}} = 1.0}$$

Keine dieser Abweichungen ist signifikant. Dennoch wirkt die Abweichung des $^{110}$Ag-Isotops nicht ganz natürlich. Offenbar kann man es auf statistische Abweichungen schieben (immerhin ist die Abweichung nicht größer als 3 $\sigma$), jedoch glaube ich, dass unsere Messungen eventuell nicht präzise genug waren. Wenn wir uns die Halbwertszeiten genauer anschauen, erkennen wir, dass dieses Isotop nach knapp 2 Messintervallen auf die Hälfte abgefallen ist. Wir haben allerdings insgesamt 40 Messungen gemacht. Ich könnte mir also als mögliche Ursache vorstellen, dass bereits nach den ersten paar Torzeiten die Aktivität dieses Isotops im Vergleich zu dem anderen Isotop (und vor allem im Vergleich mit der Untergrundzählrate) einen so kleinen Effekt gemacht hat, dass Fehler in der Untergrundmessung (diese hat doch deutlich geschwankt) die Halbwertszeit künstlich verfälscht hat. Auch war unser Fehlerbereich", in dem wir die Untergrundzählrate betrachtet haben, so gering, dass es an unseren Ergebnissen nicht wirklich etwas geändert hat, jedoch schien diese doch sehr stark zu schwanken. Möglich wäre auch, dass wir das

Präparat nicht exakt gerade aufgestellt haben oder an dem Tisch gewackelt haben, und so die Intensität der Strahlen variiert haben. Das würde allerdings nicht direkt erklären, warum die Abweichung des $^{108}$Ag-Isotops trotzdem halbwegs gering ist. Wichtig zu erkennen ist auch, dass unsere Fit-Wahrscheinlichkeiten der Funktion nicht gerade die besten waren. Eine Fitwahrscheinlichkeit von $p_{fit} \approx 14\%$ bzw. von $p_{fit} \approx 2.8\%$ deutet nicht gerade darauf hin, dass unsere Werte genau zu unseren Messwerten passen, da es doch sehr viel Schwankung gab. Trotzdem waren die letztendlichen Ergebnisse nicht so alarmierend wie diese Fitwahrscheinlichkeit vermuten lässt. An der grundsätzlichen Versuchsdurchführung ist kein grober Fehler aufgefallen, sodass wir einen systematischen Fehler am Aufbau ausschließen. Auch die weiteren Fehlerquellen bleiben reine Spekulation, da eine $2.1\sigma$-Abweichung sehr gut auch wirklich nur von statistischer Auftreten kann, und die anderen beiden Halbwertszeiten durchaus relativ akkurat sind.
Alles in allem bin ich sehr zufrieden mit unseren Ergebnissen, wir konnten erfolgreich die Halbwertszeiten und Zerfallskonstanten der drei Isotope bestimmen und so auch das Zerfallsgesetz zeigen. Wenn ich den Versuch erneut durchführen würde, würde ich vermutlich die Torzeit in der ersten Versuchsreihe auf $5s$ verringern. Auch würde ich versuchen, die Untergrundmessung länger und genauer durchzuführen und auszuwerten. Dann werden die Abweichungen hoffentlich geringer.

✓ gute Diskussion

# 5 Anhang

gutes Protokoll, einige kleinere Sachen habe ich dir direkt im Protokoll angemerkt

+

# Versuch_252_Halbwertszeit

January 15, 2025

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.stats import chi2
%matplotlib inline
```

### 0.0.1 Untergrund

[3]:
```
bkg = np.loadtxt("./Messwerte/untergrund.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")
```

Since we add the four measurments from the activation of silver together, we have to multiply the background by four, before we can subtract it.

[4]:
```
mean_bkg = np.mean(4*bkg)
fehler_bkg = np.std(4*bkg)/np.sqrt(len(bkg))
print("Mittelwert: ", mean_bkg, "Fehler: ", fehler_bkg)
```

```
Mittelwert:  14.75 Fehler:  1.2312834879822672
```

### 0.0.2 Silber

[5]:
```
n1 = np.loadtxt("./Messwerte/silber1.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")
n2 = np.loadtxt("./Messwerte/silber2.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")
n3 = np.loadtxt("./Messwerte/silber3.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")
n4 = np.loadtxt("./Messwerte/silber4.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")

N = n1 + n2 + n3 + n4
Fehler_N = np.sqrt(N)

t = np.arange(5,405,10)
```

define fit function as a superposition of two exponentials, since we have two different decaying isotopes. (larger values for the $l_i$ correspond to smaller half-life, i.e $^{110}Ag$)

[6]:
```
y0 = mean_bkg
def fit_func(x,A1,l1,A2,l2):
    return A1*np.exp(-l1*x) + A2*np.exp(-l2*x) +y0

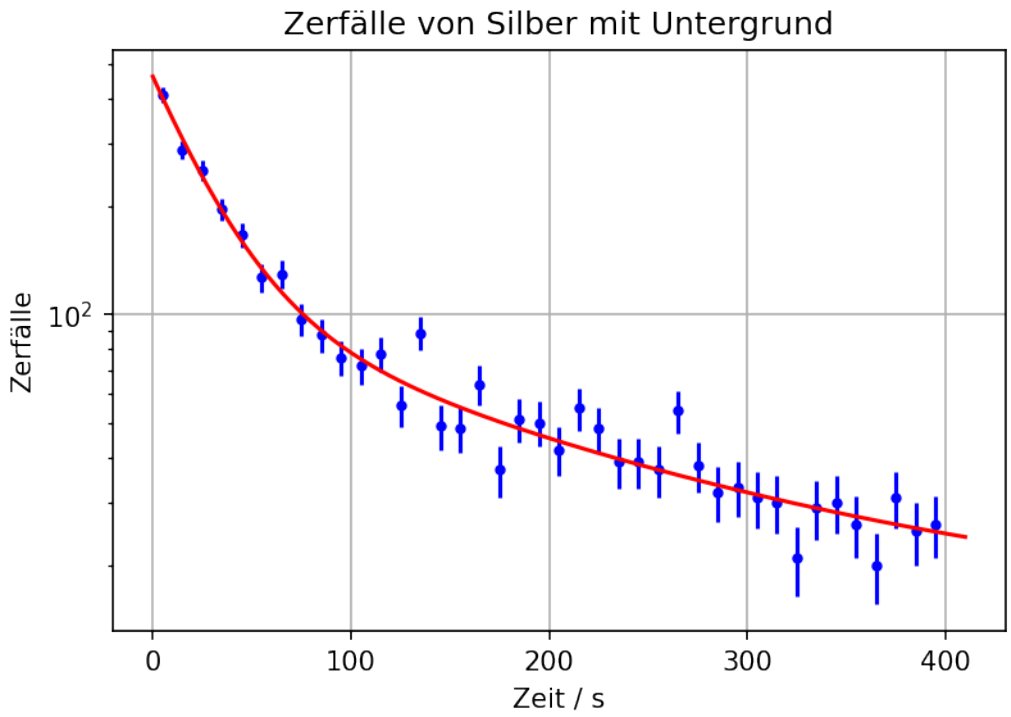
popt, pcov = curve_fit(fit_func,t,N,sigma=Fehler_N,p0=[500,0.02,50,0.001],␣
↪absolute_sigma =True)

print("A1=", popt[0], ", Standardfehler=", np.sqrt(pcov[0][0]))
print("l1=", popt[1], ", Standardfehler=", np.sqrt(pcov[1][1]))
print("A2=", popt[2], ", Standardfehler=", np.sqrt(pcov[2][2]))
print("l2=", popt[3], ", Standardfehler=", np.sqrt(pcov[3][3]))
```

```
A1= 353.8082539212489 , Standardfehler= 22.950111729567237
l1= 0.03507425331954153 , Standardfehler= 0.004062506770707304
A2= 93.13796456368725 , Standardfehler= 15.885144802500522
l2= 0.005636185469890369 , Standardfehler= 0.0007255719156083614
```

[7]:
```
t_var = np.linspace(0,410,1000)

plt.figure(dpi=150)
plt.semilogy(t,N,'b.')
plt.errorbar(t, N, Fehler_N,  fmt="None", color='blue')
plt.semilogy(t_var,fit_func(t_var,*popt),'r-')
plt.grid()
plt.xlabel("Zeit / s")
plt.ylabel("Zerfälle")
plt.title("Zerfälle von Silber mit Untergrund")
plt.savefig('Plot1Therm.png')
```




calculate the fit probability

[8]:
```
ddof = len(N)-4
chi_2 = np.sum( (fit_func(t,*popt)-N)**2/Fehler_N**2 )
chi2_red = chi_2/ddof
print("chi2 = ", chi_2, "\n"
      "chi2_red = ", chi2_red)

prob = np.round( 1-chi2.cdf(chi_2,ddof) ,2)
print( "Fit w.keit = ", prob )
```

```
chi2 =  43.591011823772206
chi2_red =  1.2108614395492279
Fit w.keit =  0.18
```

We repeat these calculations, but we add/subtract the $1\sigma$ error of the background from the mean. We first subtract the error

[9]:
```
y0 = mean_bkg - fehler_bkg
def fit_func(x,A1,l1,A2,l2):
    return A1*np.exp(-l1*x) + A2*np.exp(-l2*x) +y0

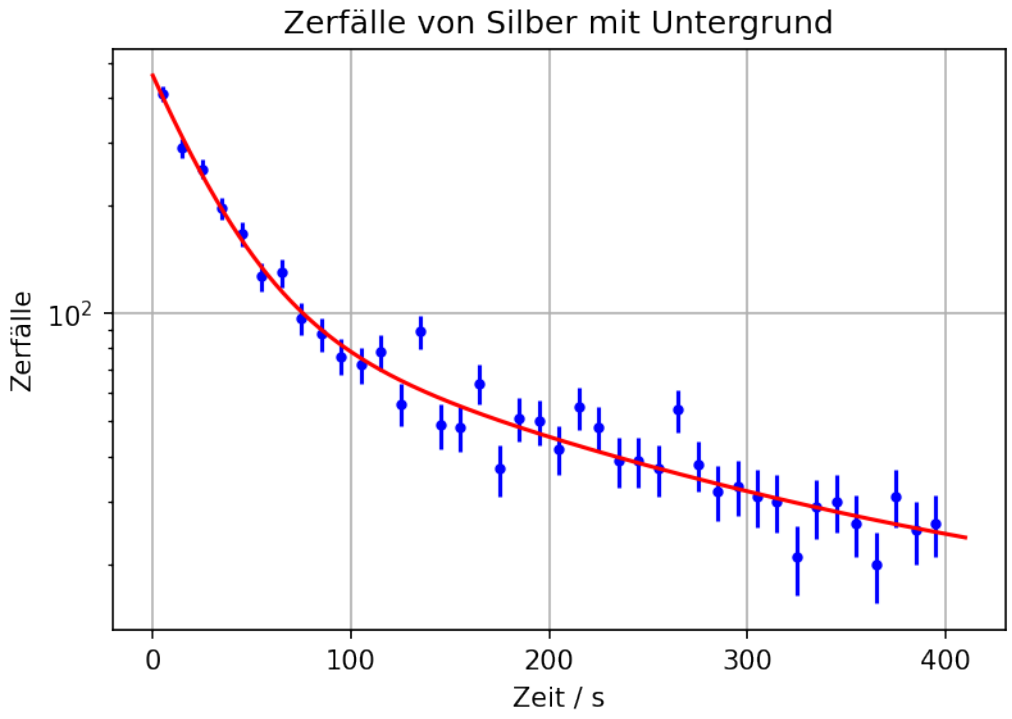
popt_min, pcov_min = curve_fit(fit_func,t,N,sigma=Fehler_N,p0=[500,0.02,50,0.
↪001], absolute_sigma =True)
```

[10]:
```
print("A1=", popt_min[0], ", Standardfehler=", np.sqrt(pcov_min[0][0]))
print("l1=", popt_min[1], ", Standardfehler=", np.sqrt(pcov_min[1][1]))
print("A2=", popt_min[2], ", Standardfehler=", np.sqrt(pcov_min[2][2]))
print("l2=", popt_min[3], ", Standardfehler=", np.sqrt(pcov_min[3][3]))

plt.figure(dpi=150)
plt.semilogy(t,N,'b.')
plt.errorbar(t, N, Fehler_N,  fmt="None", color='blue')
plt.semilogy(t_var,fit_func(t_var,*popt_min),'r-')
plt.grid()
plt.xlabel("Zeit / s")
plt.ylabel("Zerfälle")
plt.title("Zerfälle von Silber mit Untergrund")
```

```
A1= 355.88434725370274 , Standardfehler= 22.618033375064613
l1= 0.0347743225290583 , Standardfehler= 0.0039182611374851675
A2= 91.79912919094875 , Standardfehler= 15.040166304187538
l2= 0.005352180979614844 , Standardfehler= 0.0006864309226616838
```

[10]:
```
Text(0.5, 1.0, 'Zerfälle von Silber mit Untergrund')
```




[11]:
```
ddof = len(N)-4
chi_2 = np.sum( (fit_func(t,*popt)-N)**2/Fehler_N**2 )
chi2_red = chi_2/ddof
```

```
print("chi2 = ", chi_2, "\n"
      "chi2_red = ", chi2_red)

prob = np.round( 1-chi2.cdf(chi_2,ddof) ,2)
print( "Fit w.keit = ", prob )
```

```
chi2 =  44.74093780130406
chi2_red =  1.2428038278140017
Fit w.keit =  0.15
```

[12]:
```
y0 = mean_bkg + fehler_bkg
def fit_func(x,A1,l1,A2,l2):
    return A1*np.exp(-l1*x) + A2*np.exp(-l2*x) +y0

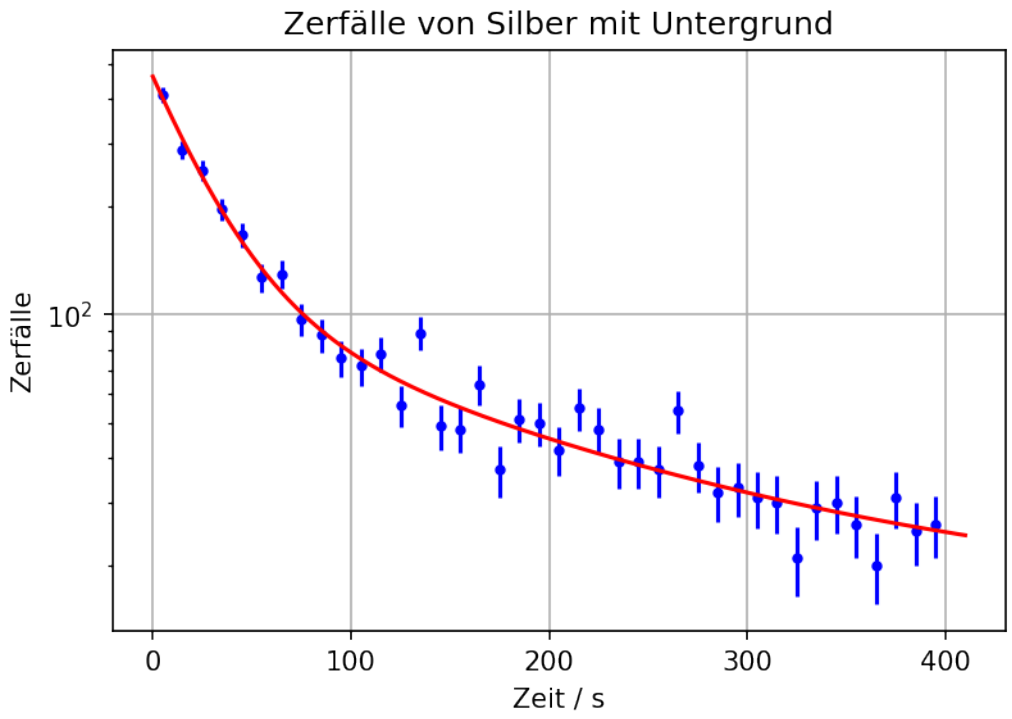
popt_pl, pcov_pl = curve_fit(fit_func,t,N,sigma=Fehler_N,p0=[500,0.02,50,0.
↪001], absolute_sigma =True)
```

[13]:
```
print("A1=", popt_pl[0], ", Standardfehler=", np.sqrt(pcov_pl[0][0]))
print("l1=", popt_pl[1], ", Standardfehler=", np.sqrt(pcov_pl[1][1]))
print("A2=", popt_pl[2], ", Standardfehler=", np.sqrt(pcov_pl[2][2]))
print("l2=", popt_pl[3], ", Standardfehler=", np.sqrt(pcov_pl[3][3]))

plt.figure(dpi=150)
plt.semilogy(t,N,'b.')
plt.errorbar(t, N, Fehler_N,  fmt="None", color='blue')
plt.semilogy(t_var,fit_func(t_var,*popt_pl),'r-')
plt.grid()
plt.xlabel("Zeit / s")
plt.ylabel("Zerfälle")
plt.title("Zerfälle von Silber mit Untergrund")
```

```
A1= 351.4223502051352 , Standardfehler= 23.354048448370797
l1= 0.0354106259475449 , Standardfehler= 0.004230087929842473
A2= 94.82157209602728 , Standardfehler= 16.863802084562074
l2= 0.005948828580329911 , Standardfehler= 0.0007695351601472311
```

[13]:
```
Text(0.5, 1.0, 'Zerfälle von Silber mit Untergrund')
```




[14]:
```
ddof = len(N)-4
chi_2 = np.sum( (fit_func(t,*popt)-N)**2/Fehler_N**2 )
chi2_red = chi_2/ddof
print("chi2 = ", chi_2, "\n"
      "chi2_red = ", chi2_red)

prob = np.round( 1-chi2.cdf(chi_2,ddof) ,2)
print( "Fit w.keit = ", prob )
```

```
chi2 =  45.10431582170402
chi2_red =  1.2528976617140006
Fit w.keit =  0.14
```

[15]:
```
diff_min = popt[1:4:2]-popt_min[1:4:2] # arrays truncated to the decay rates to␣
 ↪calculate both differences at once
diff_pl = popt_pl[1:4:2]-popt[1:4:2]

mean_diff = np.mean([diff_min,diff_pl],axis=0)
print(mean_diff)
delta_l1 = np.sqrt(pcov[1][1] + mean_diff[0]**2)
delta_l2 = np.sqrt(pcov[3][3] + mean_diff[1]**2)
print(delta_l1,delta_l2)
```

```
[0.00031815 0.00029832]
```

```
0.004074945615850243 0.0007845072941530554
```

[16]:
```
T_Ag108 = np.log(2)/popt[1]
T_Ag110 = np.log(2)/popt[3]
print("T_Ag108=", T_Ag108)
print("Fehler=", T_Ag108 / (popt[1]/delta_l1))
print("T_Ag110=", T_Ag110)
print("Fehler=", T_Ag110 / (popt[3]/delta_l2))
```

```
T_Ag108= 19.76227902116907
Fehler= 2.295992206102216
T_Ag110= 122.98161305423257
Fehler= 17.117955575303437
```

### 0.0.3 Indium

[17]:
```
N_i = np.loadtxt("./Messwerte/indium.txt", skiprows = 4, usecols=[1],␣
 ↪delimiter=",", encoding="latin-1")
t_i = np.arange(0,120*25,120)
bkg_120 = np.sum(bkg[0:13])
mean_bkg_120 = np.mean(bkg_120)
fehler_bkg_120 = np.std(bkg[0:13])/np.sqrt(12)
print(bkg_120, fehler_bkg_120)

Fehler_N_i = np.sqrt(N_i)

y0 = mean_bkg_120
def fit_func(x,A,l):
    return A*np.exp(-l*x) +y0

popt, pcov = curve_fit(fit_func,t_i,N_i,sigma=Fehler_N_i,p0=[50,0.001])

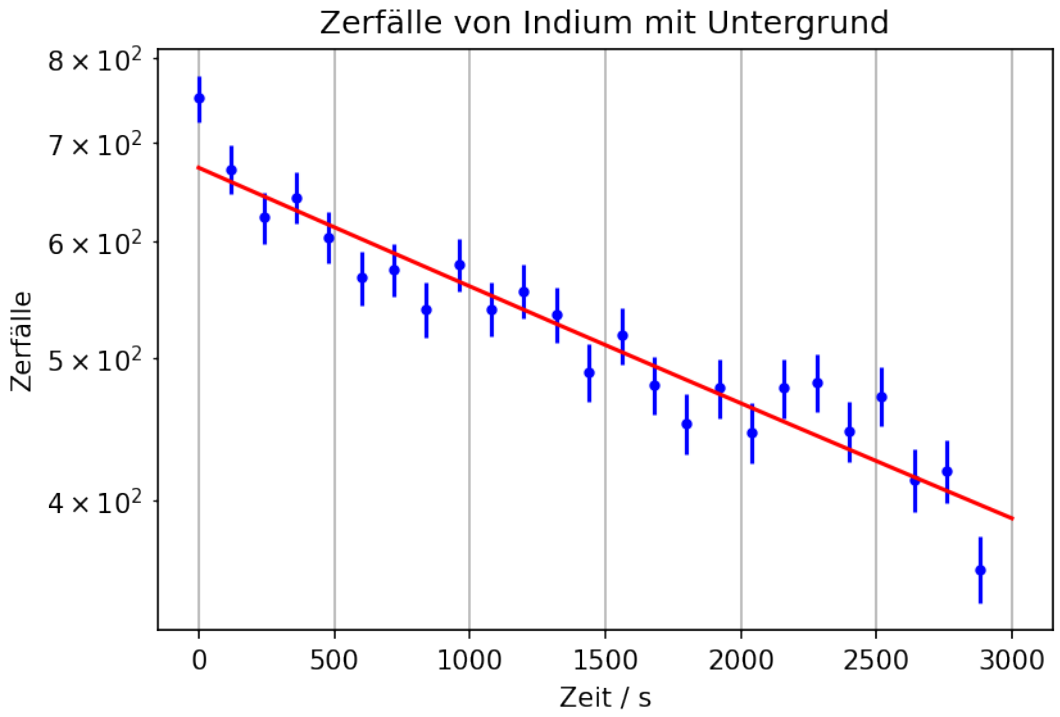
print("A=", popt[0], ", Standardfehler=", np.sqrt(pcov[0][0]))
print("l=", popt[1], ", Standardfehler=", np.sqrt(pcov[1][1]))
```

```
39.0 0.6201736729460423
A= 634.6804439398444 , Standardfehler= 13.376770865944431
l= 0.0001986294919130756 , Standardfehler= 1.3702874365616592e-05
```

[18]:
```
t_var = np.linspace(0,120*25,1000)

plt.figure(dpi=150)
plt.semilogy(t_i,N_i,'b.')
plt.errorbar(t_i, N_i, Fehler_N_i,  fmt="None", color='blue')
plt.semilogy(t_var,fit_func(t_var,*popt),'r-')
plt.grid()
plt.xlabel("Zeit / s")
plt.ylabel("Zerfälle")
```

```
plt.title("Zerfälle von Indium mit Untergrund")
plt.savefig('Plot2Therm.png')
```




[19]:
```
y0 = mean_bkg_120 - fehler_bkg_120
def fit_func(x,A,l):
    return A*np.exp(-l*x) + y0

popt_min, pcov_min = curve_fit(fit_func,t_i,N_i,sigma=Fehler_N_i,p0=[50,0.001])

print("A=", popt[0], ", Standardfehler=", np.sqrt(pcov[0][0]))
print("l=", popt[1], ", Standardfehler=", np.sqrt(pcov[1][1]))
```

```
A= 634.6804439398444 , Standardfehler= 13.376770865944431
l= 0.0001986294919130756 , Standardfehler= 1.3702874365616592e-05
```

[20]:
```
y0 = mean_bkg_120 + fehler_bkg_120
def fit_func(x,A,l):
    return A*np.exp(-l*x) + y0

popt_pl, pcov_pl = curve_fit(fit_func,t_i,N_i,sigma=Fehler_N_i,p0=[50,0.001])

print("A=", popt[0], ", Standardfehler=", np.sqrt(pcov[0][0]))
print("l=", popt[1], ", Standardfehler=", np.sqrt(pcov[1][1]))
```

```
A= 634.6804439398444 , Standardfehler= 13.376770865944431
l= 0.0001986294919130756 , Standardfehler= 1.3702874365616592e-05
```

[26]:
```
diff_min = popt[1]-popt_min[1]
diff_pl = popt_pl[1]-popt[1]

mean_diff = np.mean([diff_min,diff_pl],axis=0)
print(mean_diff)
delta_l = np.sqrt(pcov[1][1] + mean_diff**2)
print(delta_l)
```

```
2.612484720538482e-07
1.370536452065478e-05
```

[28]:
```
chi2_ = np.sum((fit_func(t_i,*popt)-N_i)**2/Fehler_N_i**2)
dof_i = len(N_i)-4
chi2_red = chi2_/dof_i
print("chi2=", chi2_)
print("chi2_red=", chi2_red)
```

```
chi2= 35.026438006791835
chi2_red= 1.6679256193710397
```

[29]:
```
prob = round(1-chi2.cdf(chi2_, dof_i),3)
print("Wahrscheinlichkeit=", prob)
```

```
Wahrscheinlichkeit= 0.028
```

[30]:
```
T_In116 = np.log(2)/popt[1]
print("T_In116=", T_In116)
```

```
T_In116= 3489.6488627342455
```

[31]:
```
deltaT = T_In116 * delta_l/popt[1]
print(deltaT)
```

```
240.78453432177776
```