

Versuch 252 - Aktivierung von Indium und Silber mit thermischen Neutronen

Felix Fleischle

25.2.2022

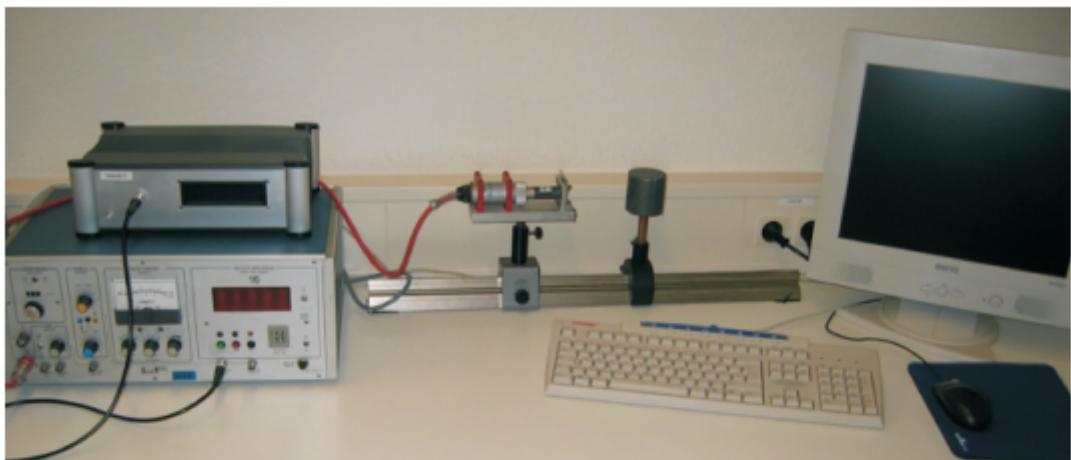


Abbildung 1: Bild des Versuchsaufbaus aus der Praktikumsanleitung

Inhaltsverzeichnis

1 Einleitung	3
1.1 Grundlagen	3
1.1.1 Aktivierung	3
1.1.2 Zerfallsgesetz bei Aktivierung	3
1.2 Durchführung	4
1.2.1 Untergrundmessung	4
1.2.2 Halbwertszeit von Silber	4
1.2.3 Halbwertszeit von Indium	4
2 Auswertung	5
2.1 Zerfall der Silberisotope	5
2.2 Zerfall der Indiumisotope	8
3 Zusammenfassung und Diskussion	11

1 Einleitung

Das Ziel dieses Versuchs ist es, die Halbwertszeit von ^{115}In sowie von ^{108}Ag und ^{110}Ag zu bestimmen.

1.1 Grundlagen

Wir wollen die radioaktiven Isotope von Silber und Indium durch Aktivierung aus den stabilen Isotopen ^{115}In und $^{107}\text{Ag}/^{109}\text{Ag}$ von Silber und Indium erzeugen.

1.1.1 Aktivierung

Bei der Aktivierung werden die stabilen Isotope mit Neutronen beschossen, welche leicht in den Kern eindringen können, da diese keine Ladung haben. Diese erhöhen die Massenzahl der Atome um 1, wodurch die oben genannten radioaktiven Isotope entstehen.

Die Neutronenquelle besteht aus Berylliumspänen und einem α -Strahler, wobei die Kernreaktion



abläuft. Dabei entstehen Neutronen mit einer Energie von 1-10 MeV. Diese werden durch Stöße mit Wasserstoffkernen, sowie mit Protonen abgebremst, bis diese nahezu thermische Energie erreicht haben. Dann treffen die Neutronen auf die stabilen Isotope.

Bei der Bestrahlung von ^{115}In entstehen die beiden Isomere ^{115}In und ^{115m}In . Das letztere befindet sich in einem höheren Energieniveau als ^{115}In , und hat eine deutlich längere Halbwertszeit von ca. 54min, im Vergleich zu ^{115}In mit einer Halbwertszeit von 14s. Beide Isomere sind β^- -Strahler und zerfallen in ^{116}Sn .

Bei der Bestrahlung von natürlichem Silber, welches jeweils ca. zur Hälfte aus ^{107}Ag und ^{109}Ag besteht, entstehen die beiden Isotope ^{108}Ag und ^{110}Ag , welche auch beide β^- -Strahler sind. Sie zerfallen durch β^- -Zerfall in ^{108}Cd und ^{110}Cd .

1.1.2 Zerfallsgesetz bei Aktivierung

Die Aktivität entspricht der Anzahl an gemessenen Zerfälle pro Sekunde. So kann aus dem Zerfallsgesetz die folgende Beziehung herleiten:

$$A(t) = -\frac{d}{dt}N_0e^{-\lambda t} = A_0e^{-\lambda t} \quad (1.2)$$

Bei der Aktivierung ist aber nur ein Teil der Atome aktiviert, und damit fähig zu Zerfallen. So ergibt sich ein Gleichgewicht für die Aktivität mit fortlaufender Zeit

$$A(t) = A_\infty(1 - e^{-\lambda t}) \quad (1.3)$$

Die Halbwertszeit des Isotops ergibt sich nach der Messung aus

$$T_{1/2} = \frac{\ln 2}{\lambda} \quad (1.4)$$

1.2 Durchführung

1.2.1 Untergrundmessung

Wir führen zunächst eine Untergrundmessung durch. Dafür entfernen wir alle Quellen aus dem Raum, und stellen am Geiger-Müller Zählrohr eine Spannung von 500-550 Volt ein. Wir starten dann die Messung am Computer mit einer Torzeit von 10s und messen den Untergrund über einen Zeitraum von 8 Minuten.

1.2.2 Halbwertszeit von Silber

Wir stecken nun ein Silberblech für 7 Minuten in die Neutronenquelle, und stecken es danach so schnell wie möglich vor das Zählrohr. Die Torzeit beträgt weiterhin 10 Sekunden. Wir messen für insgesamt 400 Sekunden. Wir wiederholen die Messung drei Mal und speichern die Messdaten.

1.2.3 Halbwertszeit von Indium

Wir stellen eine Torzeit von 2 Minuten ein und stecken das aktivierte Indium-Präparat vor das Zählrohr. Hier messen wir für insgesamt 50 Minuten.

Versuch Z52 Aktivierung - Messprotokoll

Felix Fleischle 25.2.22 David Hildebrandt

Messaufbau

- Geiger-Müller Zählrohr mit Betriebsgerät
- Externer ~~zählrohr~~ Impulszähler
- PC mit Drucker
- Neutronenquelle
- Präparathalterung
- Indium- und Silberbleche

1. Halbwertszeit von Silber

Die Messergebnisse sind digital abgespeichert

2. Halbwertszeit von Indium

Die Messergebnisse sind digital abgespeichert

2 Auswertung

2.1 Zerfall der Silberisotope

Wir wollen zunächst den Untergrund bestimmen. Dazu importieren wir unsere Messdaten zum Untergrund in Python, und nehmen diesen mal 4, da wir am Ende die Zerfälle von allen vier Messreihen zu Silber addieren. Wir bestimmen dann den Mittelwert, sowie den Fehler des Mittelwertes aus unseren Daten. Wir erhalten

$$u_m = 11,0 \pm 0,9 \quad (2.1)$$

für den Mittelwert der Zerfälle pro Zeitintervall.

Wir importieren nun die Daten für unsere Messreihen für Silber in Python, und addieren die gezählten Zerfälle punktweise, mit dem Fehler \sqrt{N} . Wir plotten unsere Messwerte als Funktion der Zeit mit einer logarithmischen Skala:

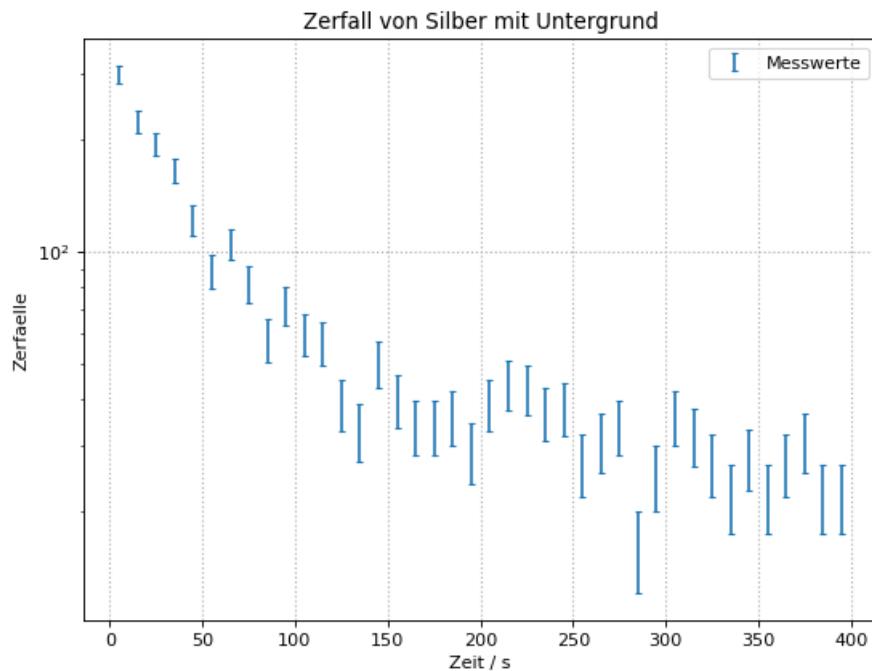


Abbildung 2: Zerfälle der Silberisotope als Funktion der Zeit

Wir definieren nun als fit-Funktion eine Superposition von zwei Exponentialfunktionen plus Untergrund:

$$A(t) = A_1 e^{-\lambda_1 t} + A_2 e^{-\lambda_2 t} + u_m \quad (2.2)$$

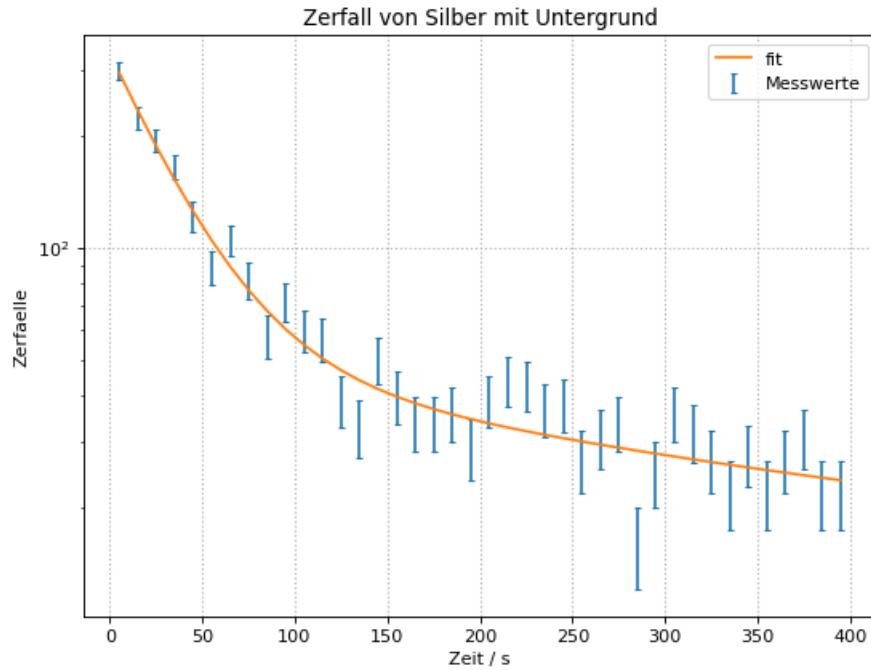


Abbildung 3: Zerfälle der Silberisotope mit fit

Wir lassen und die Fitparameter ausgeben und erhalten

Zerfallsfunktion	λ [1/s]	A
1	$0,028 \pm 0,003$	285 ± 19
2	$0,0029 \pm 0,0010$	39 ± 11

Tabelle 1: Fitparameter Silber

Wir berechnen außerdem die χ^2 -Summe und $\chi^2_{red} = \chi^2/f$ mit der Zahl der Freiheitsgrade f . Zusätzlich berechnen wir noch die Fitwahrscheinlichkeit, dass bei einer erneuten Messung ein χ^2 -Wert größer oder gleich dem aktuellen herauskommt. Wir erhalten

$$\chi^2 = 46,57 \quad (2.3)$$

$$\chi^2_{red} = 1,29 \quad (2.4)$$

$$P = 11,0\% \quad (2.5)$$

Der χ^2_{red} -Wert zeigt, dass der fit die Messwerte gut die Messwerte beschreibt.

Wir wollen nun zusätzlich den Fehler des Untergrunds berücksichtigen. Dazu führen wir zwei weitere fits durch mit $u_m + \Delta u_m$ und $u_m - \Delta u_m$ in der fit-Funktion. Wir erhalten dabei für die Fitparameter

Exponentialfunktion	$\lambda_+ [1/s]$	A_+	$\lambda_- [1/s]$	A_-
1	$0,028 \pm 0,003$	284 ± 19	$0,028 \pm 0,003$	285 ± 19
2	$0,0030 \pm 0,0011$	39 ± 11	$0,0027 \pm 0,0010$	40 ± 11

Tabelle 2: Fitparameter mit niedrigstem und höchstem Untergrund

Wir berechnen die folgenden Differenzen und deren Mittelwert:

$$|\lambda_1 - \lambda_{1,+}| = 4,83 \cdot 10^{-5} \frac{1}{s} \quad (2.6)$$

$$|\lambda_1 - \lambda_{1,-}| = 4,38 \cdot 10^{-5} \frac{1}{s} \quad (2.7)$$

$$|\lambda_2 - \lambda_{2,+}| = 0,000142 \frac{1}{s} \quad (2.8)$$

$$|\lambda_2 - \lambda_{2,-}| = 0,000136 \frac{1}{s} \quad (2.9)$$

$$(\Delta\lambda)_{u,1} = \frac{|\lambda_1 - \lambda_{1,+}| + |\lambda_1 - \lambda_{1,-}|}{2} = 4,60 \cdot 10^{-5} \frac{1}{s} \quad (2.10)$$

$$(\Delta\lambda)_{u,2} = \frac{|\lambda_2 - \lambda_{2,+}| + |\lambda_2 - \lambda_{2,-}|}{2} = 0,000136 \frac{1}{s} \quad (2.11)$$

Damit ergibt sich der Gesamtfehler mit dem Fehler des ersten fits zu

$$\Delta\lambda_1 = \sqrt{(\Delta\lambda)_{1,f}^2 + (\Delta\lambda)_{u,1}^2} = 0,00327 \frac{1}{s} \quad (2.12)$$

$$\Delta\lambda_2 = \sqrt{(\Delta\lambda)_{2,f}^2 + (\Delta\lambda)_{u,2}^2} = 0,00103 \frac{1}{s} \quad (2.13)$$

Unser Endergebnis für die zwei Zerfallskonstanten lautet also

$$\lambda_1 = (0,028 \pm 0,003) \frac{1}{s} \quad (2.14)$$

$$\lambda_2 = (0,0029 \pm 0,0010) \frac{1}{s} \quad (2.15)$$

Wir berechnen nun aus den Zerfallskonstanten die Halbwertszeiten und Lebensdauern der beiden Isotope.

$$T_{1/2,1} = \frac{\ln(2)}{\lambda_1} \pm \frac{\ln(2)\Delta\lambda_1}{\lambda_1^2} = (24,6 \pm 2,8)s \quad (2.16)$$

$$T_{1/2,2} = \frac{\ln(2)}{\lambda_2} \pm \frac{\ln(2)\Delta\lambda_2}{\lambda_2^2} = (240 \pm 90)s \quad (2.17)$$

$$\tau_1 = \frac{1}{\lambda_1} \pm \frac{\Delta\lambda_1}{\lambda_1^2} = (35 \pm 4)s \quad (2.18)$$

$$\tau_2 = \frac{1}{\lambda_2} \pm \frac{\Delta\lambda_2}{\lambda_2^2} = (350 \pm 130)s \quad (2.19)$$

Wie wir sehen können ist der Fehler bei den Zeiten für λ_2 sehr groß. Dies liegt an der bereits sehr großen Standardabweichung, welche sich bereits beim ersten fit für λ_2 ergeben hat. Wir können nun die Halbwertszeiten den Isotopen zuordnen. Aus der Nukleidkarte entnehmen wir die Halbwertszeiten $T_{Ag108} = 144,6\text{s}$ und $T_{Ag110} = 24,6\text{s}$. Wir sehen also dass unser λ_1 ziemlich genau der Zerfallskonstante von ^{110}Ag entspricht, und entsprechend λ_2 ^{108}Ag . Die genauen Abweichungen berechnen wir in der Diskussion.

2.2 Zerfall der Indiumisotope

Wir importieren analog zur Berechnung mit Silber unsere Messwerte für Indium in Python, und plotten diese gegen die Zeit.

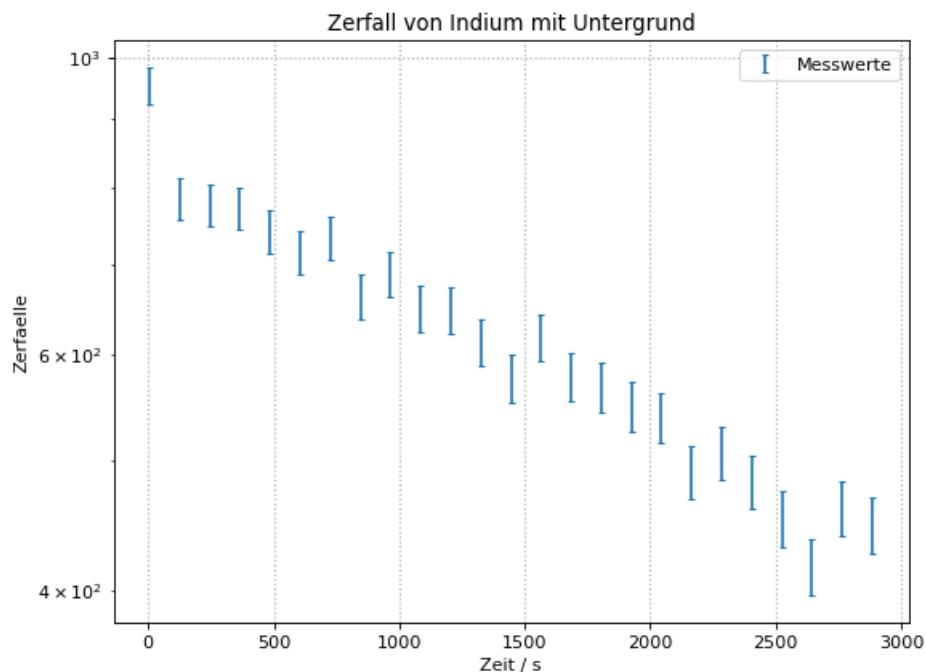


Abbildung 4: Zerfall der Indiumisotope

Wir müssen nun den Untergrund auf unsere Messung mit Indium anpassen. Zuerst müssen wir den Untergrund wieder durch 4 teilen, da wir hier keine 4 Messreihen addieren. Außerdem ist unsere Torzeit hier 12 mal so groß wie bei der Untergrundmessung, also müssen wir die Werte mal 12 nehmen.

Außerdem sehen wir hier, dass der erste Messwert deutlich außerhalb der ansonsten auf einer Geraden liegenden Messwerte liegt. Dies liegt am Zerfall von ^{116}In mit einer sehr viel kürzeren Halbwertszeit als ^{116m}In . Wir vernachlässigen also den ersten Messwert für den fit, da uns nur die Halbwertszeit von ^{116m}In interessiert.

Wir führen also einen fit mit einer einfachen Exponentialfunktion plus Untergrund durch.

$$A(t) = A_3 e^{-\lambda_3 t} + u_m \quad (2.20)$$

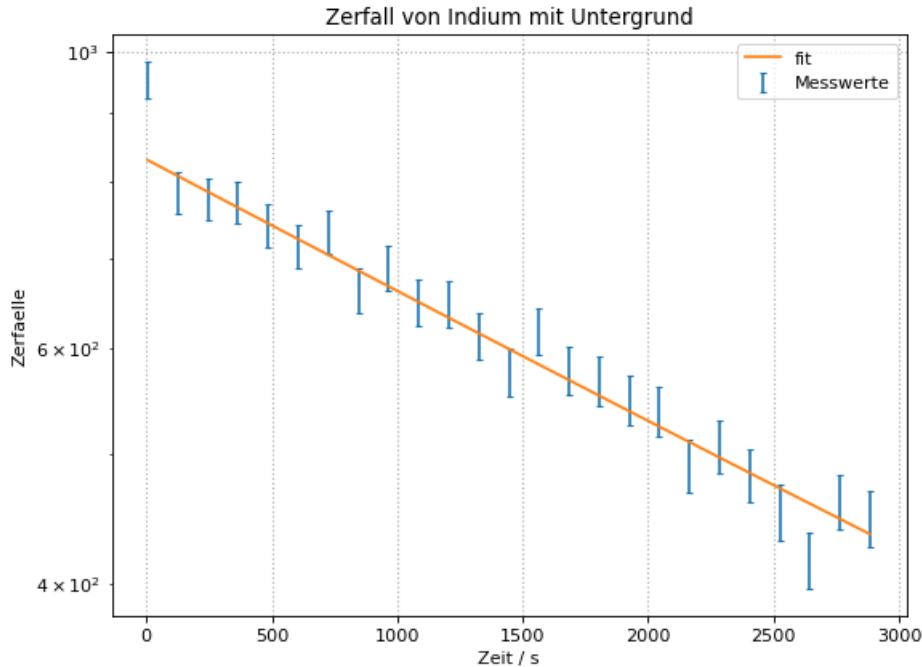


Abbildung 5: Zerfall von Indium mit fit

Unsere Fitparameter ergeben sich zu

$$\lambda_3 = (0,000237 \pm 0,000009) \frac{1}{s} \quad (2.21)$$

$$A_3 = 798 \pm 11 \quad (2.22)$$

Wir berechnen erneut die χ^2 -Summe, χ^2_{red} und die Fitwahrscheinlichkeit:

$$\chi^2 = 14,84 \quad (2.23)$$

$$\chi^2_{red} = 0,74 \quad (2.24)$$

$$P = 79\% \quad (2.25)$$

Dieser fit beschreibt die Messwerte also auch gut.

Wir berücksichtigen erneut den Fehler des Untergrunds mit zwei erneuten fits:

Untergrund	λ_3 [1/s]	A_3	$ \lambda_3 - \lambda_i $ [1/s]	Mittelwert der Differenzen [1/s]
$u_m + \Delta u_m$	$0,000238 \pm 0,000009$	795 ± 11	$1,09 \cdot 10^{-6}$	$1,09 \cdot 10^{-6}$
$u_m - \Delta u_m$	$0,000236 \pm 0,000009$	800 ± 11	$1,08 \cdot 10^{-6}$	"

Tabelle 3: Fitparameter und Differenzen bei Variation des Untergrunds

Damit ergibt sich der Gesamtfehler zu

$$\Delta\lambda_3 = \sqrt{(\Delta\lambda)_{3,f}^2 + (\Delta\lambda)_{u,3}^2} = 8,85 \cdot 10^{-6} \frac{1}{s} \quad (2.26)$$

und damit unser Endergebnis für die Zerfallskonstante

$$\lambda_3 = (0,000237 \pm 0,000009) \frac{1}{s} \quad (2.27)$$

Wir berechnen damit die Halbwertszeit und Lebensdauer von ^{116m}In :

$$T_{1/2,3} = \frac{\ln(2)}{\lambda_3} \pm \frac{\ln(2)\Delta\lambda_3}{\lambda_3^2} = (2930 \pm 110)s \quad (2.28)$$

$$\tau_3 = \frac{1}{\lambda_3} \pm \frac{\Delta\lambda_3}{\lambda_3^2} = (4220 \pm 160)s \quad (2.29)$$

Der Literaturwert beträgt $T_{1/2} = 3240s$.

3 Zusammenfassung und Diskussion

Die Ziele dieses Versuchs waren es, die Halbwertszeit von ^{166}In sowie von ^{108}Ag und ^{110}Ag zu bestimmen.

Wir haben zunächst bei einer Zählrohrspannung von 525V über 8 Minuten eine Untergrundmessung durchgeführt, mit einer Torzeit von 10 Sekunden. Unsere Messdaten haben wir in Python importiert und den Mittelwert und Fehler des Untergrunds bestimmt. Unser Ergebnis war

$$u_m = 11,0 \pm 0,9 \quad (3.1)$$

Danach haben wir die Zerfälle von aktiviertem Silber über einen Zeitraum von 400 Sekunden mit einer Torzeit von 10 Sekunden gemessen, und diese Messreihe insgesamt 4 mal durchgeführt. Wir haben unsere Messdaten in Python punktweise aufaddiert und als Funktion der Zeit geplottet. Wir haben einen fit mit einer Superposition von zwei Exponentialfunktionen durchgeführt. Als Fitparameter haben wir dabei die folgenden Ergebnisse erhalten:

Zerfallsfunktion	λ [1/s]	A
1	$0,028 \pm 0,003$	285 ± 19
2	$0,0029 \pm 0,0010$	39 ± 11

Tabelle 4: Fitparameter Silber

Wir haben außerdem $\chi^2 = 46,57$, $\chi^2_{red} = 1,29$ und $P = 11,0\%$ berechnet, was zeigt dass der fit die Messwerte gut annähert.

Außerdem haben wir noch den Fehler des Untergrunds berücksichtigt, indem wir nochmal zwei fits, einmal mit minimalem Untergrund, und einmal mit maximalem Untergrund, durchgeführt haben. Als Fitparameter haben wir folgende Ergebnisse erhalten:

Exponentialfunktion	λ_+ [1/s]	A_+	λ_- [1/s]	A_-
1	$0,028 \pm 0,003$	284 ± 19	$0,028 \pm 0,003$	285 ± 19
2	$0,0030 \pm 0,0011$	39 ± 11	$0,0027 \pm 0,0010$	40 ± 11

Tabelle 5: Fitparameter mit niedrigstem und höchstem Untergrund

Wir haben dann die Differenzen zwischen den Zerfallskonstanten der verschiedenen fits, sowie den Mittelwert den Abweichung berechnet, woraus wir den folgenden Gesamtfehler der Zerfallskonstanten erhalten:

$$\Delta\lambda_1 = \sqrt{(\Delta\lambda)_{1,f}^2 + (\Delta\lambda)_{u,1}^2} = 0,00327 \frac{1}{s} \quad (3.2)$$

$$\Delta\lambda_2 = \sqrt{(\Delta\lambda)_{2,f}^2 + (\Delta\lambda)_{u,2}^2} = 0,00103 \frac{1}{s} \quad (3.3)$$

Damit lauteten unsere Endergebnisse für die Zerfallskonstanten

$$\lambda_1 = (0,028 \pm 0,003) \frac{1}{s} \quad (3.4)$$

$$\lambda_2 = (0,0029 \pm 0,0010) \frac{1}{s} \quad (3.5)$$

Daraus haben wir die Halbwertszeiten und Lebensdauern berechnet:

$$T_{1/2,1} = \frac{\ln(2)}{\lambda_1} \pm \frac{\ln(2)\Delta\lambda_1}{\lambda_1^2} = (24,6 \pm 2,8)s \quad (3.6)$$

$$T_{1/2,2} = \frac{\ln(2)}{\lambda_2} \pm \frac{\ln(2)\Delta\lambda_2}{\lambda_2^2} = (240 \pm 90)s \quad (3.7)$$

$$\tau_1 = \frac{1}{\lambda_1} \pm \frac{\Delta\lambda_1}{\lambda_1^2} = (35 \pm 4)s \quad (3.8)$$

$$\tau_2 = \frac{1}{\lambda_2} \pm \frac{\Delta\lambda_2}{\lambda_2^2} = (350 \pm 130)s \quad (3.9)$$

Wir konnten mit den Halbwertszeiten aus der Nukleidkarte $T_{Ag108} = 144,6s$ und $T_{Ag110} = 24,6s$ die Halbwertszeiten den Isotopen zuordnen. Wir haben gesehen dass unser λ_1 ziemlich genau der Zerfallskonstante von ^{110}Ag entspricht, und entsprechend λ_2 ^{108}Ag . Für die Abweichungen ergeben sich mit den obigen Literaturwerten:

$$\sigma_{110Ag} = 0,0069 \quad (3.10)$$

$$\sigma_{108Ag} = 1,12 \quad (3.11)$$

Die Werte stimmen also gut überein, hauptsächlich wegen dem großen Fehler von λ_2 , welcher beim fit entstanden ist.

Wir haben die selben Rechnungen für unsere Messung mit Indium durchgeführt, angepasst auf die Torzeit von 120s und Messzeit von 50 Minuten. Dabei haben wir den ersten Messwert vernachlässigt, da wir nur die Halbwertszeit von ^{116m}In bestimmen wollen. Wir haben eine einfache Exponentialfunktion gefittet, und als Fitparameter folgende Ergebnisse erhalten:

$$\lambda_3 = (0,000237 \pm 0,000009) \frac{1}{s} \quad (3.12)$$

$$A_3 = 798 \pm 11 \quad (3.13)$$

Wir haben dann erneut χ^2 , χ^2_{red} und die Fitwahrscheinlichkeit bestimmt:

$$\chi^2 = 14,84 \quad (3.14)$$

$$\chi^2_{red} = 0,74 \quad (3.15)$$

$$P = 79\% \quad (3.16)$$

Auch hier beschreibt der fit die Messwerte gut. Bei der Berücksichtigung des Fehlers des Untergrundes haben wir folgende Fitparameter erhalten:

Untergrund	λ_3 [1/s]	A_3	$ \lambda_3 - \lambda_i $ [1/s]	Mittelwert der Differenzen [1/s]
$u_m + \Delta u_m$	$0,000238 \pm 0,000009$	795 ± 11	$1,09 \cdot 10^{-6}$	$1,09 \cdot 10^{-6}$
$u_m - \Delta u_m$	$0,000236 \pm 0,000009$	800 ± 11	$1,08 \cdot 10^{-6}$	"

Tabelle 6: Fitparameter und Differenzen bei Variation des Untergrunds

und damit den Gesamtfehler und Endergebnis

$$\Delta\lambda_3 = \sqrt{(\Delta\lambda)_{3,f}^2 + (\Delta\lambda)_{u,3}^2} = 8,85 \cdot 10^{-6} \frac{1}{s} \quad (3.17)$$

$$\lambda_3 = (0,000237 \pm 0,000009) \frac{1}{s} \quad (3.18)$$

Für die Halbwertszeit und Lebensdauer haben wir folgende Werte erhalten:

$$T_{1/2,3} = \frac{\ln(2)}{\lambda_3} \pm \frac{\ln(2)\Delta\lambda_3}{\lambda_3^2} = (2930 \pm 110)s \quad (3.19)$$

$$\tau_3 = \frac{1}{\lambda_3} \pm \frac{\Delta\lambda_3}{\lambda_3^2} = (4220 \pm 160)s \quad (3.20)$$

Mit dem Literaturwert von $T_{1/2} = 3240s$ ergibt sich damit eine Abweichung von

$$\sigma_{116mIn} = 2,87 \quad (3.21)$$

Die Abweichung ist also nicht signifikant.

Versuch 252 Aktivierung - Auswertung

Felix Fleischle - 25.2.2022

```
In [1]: import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
```

```
In [2]: unterg =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_Untergrund.dat', usecols=[1])
```

```
In [3]: mittelw_unterg=np.mean(4*unterg)
fehler_unterg=np.std(4*unterg)/np.sqrt(len(unterg))
print('Mittelwert:', mittelw_unterg, 'Fehler:', fehler_unterg)
```

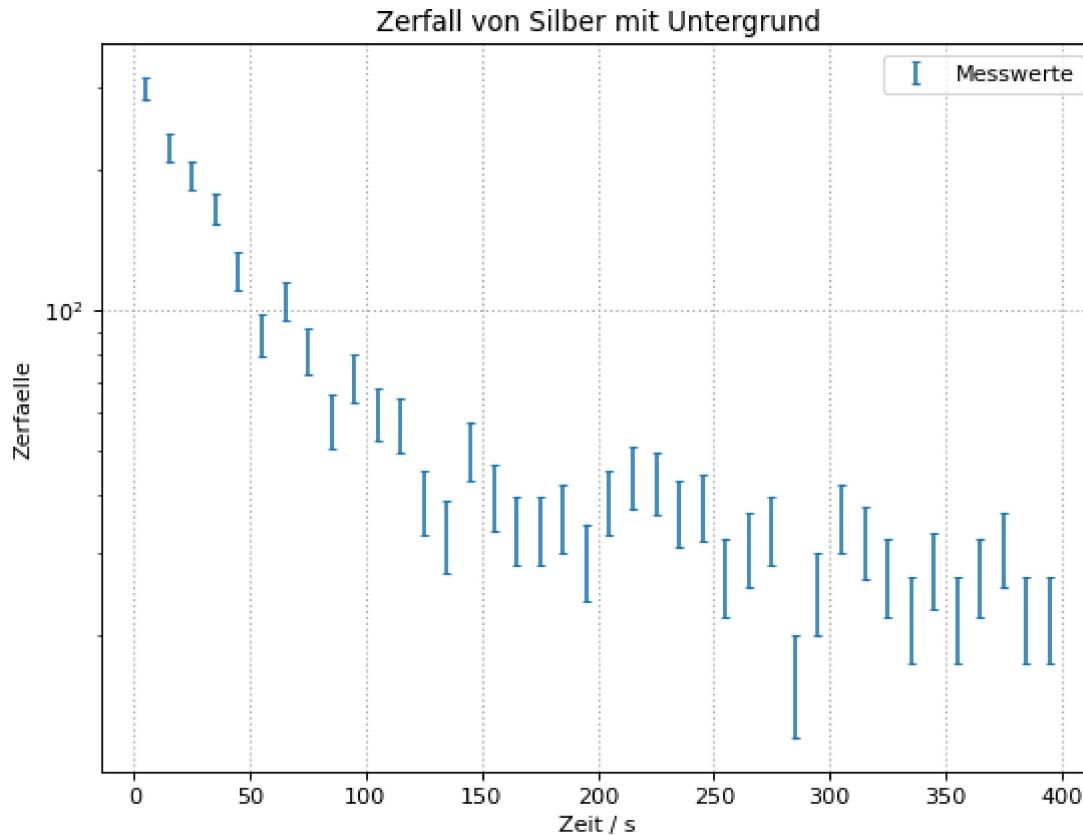
Mittelwert: 11.020408163265307 Fehler: 0.8680020555535112

```
In [4]: n1 =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_silber1.dat', usecols=[1])
n2 =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_silber2.dat', usecols=[1])
n3 =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_silber3.dat', usecols=[1])
n4 =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_silber4.dat', usecols=[1])

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

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

```
In [5]: plt.figure(figsize=(8, 6), dpi=80)
plt.errorbar(t,N, Fehler_N, linestyle='None', label="Messwerte", capsize = 2)
plt.xlabel('Zeit / s')
plt.ylabel('Zerfaelle')
plt.title('Zerfall von Silber mit Untergrund')
plt.yscale('log')
plt.legend()
plt.grid(linestyle=":", linewidth=1)
```

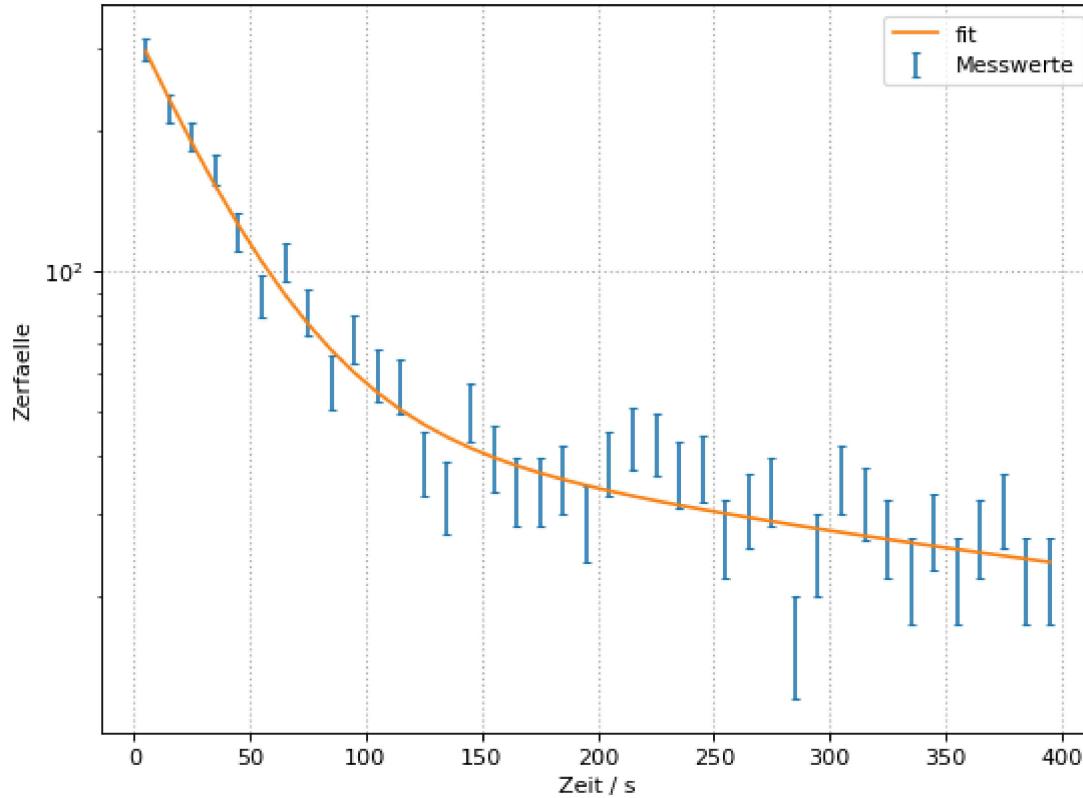


```
In [6]: y0=mittelw_unterg #Untergrund
def fit_func(x, A1,l1,A2,l2):
    return A1*np.exp(-x*l1) + A2*np.exp(-x*l2) + y0

popt, pcov=curve_fit(fit_func,t,N, p0=[500,0.02,50,0.001],sigma=Fehler_N)
```

```
In [7]: plt.figure(figsize=(8, 6), dpi=80)
plt.errorbar(t,N, Fehler_N, linestyle='None', label="Messwerte", capsize = 2)
plt.xlabel('Zeit / s')
plt.ylabel('Zerfaelle')
plt.title('Zerfall von Silber mit Untergrund')
plt.yscale('log')
plt.plot(t,fit_func(t,*popt), label="fit")
plt.legend()
plt.grid(style=":", linewidth=1)
```

Zerfall von Silber mit Untergrund



```
In [8]: 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= 285.1422805327275 , Standardfehler= 18.812901253577955
l1= 0.028199318566941082 , Standardfehler= 0.0032675069585057378
A2= 39.17533164177607 , Standardfehler= 10.967153316851574
l2= 0.0028634023605370296 , Standardfehler= 0.001021005753672669
```

```
In [9]: chi2_=np.sum((fit_func(t,*popt)-N)**2/Fehler_N**2)
dof=len(N)-4 #dof: degrees of freedom, Freiheitsgrad
chi2_red=chi2_/dof
print("chi2=", chi2_)
print("chi2_red=",chi2_red)

from scipy.stats import chi2
```

```
prob=round(1-chi2.cdf(chi2_,dof),2)*100
print("Wahrscheinlichkeit=", prob,"%")
```

```
chi2= 46.57015650893761
chi2_red= 1.2936154585816002
Wahrscheinlichkeit= 11.0 %
```

In [10]:

```
# Wiederholung mit variiertem Untergrund
y0_plus = mittelw_unterg + fehler_unterg
y0_minus = mittelw_unterg - fehler_unterg
def fit_func_plus(x, A1,l1,A2,l2):
    return A1*np.exp(-x*l1) + A2*np.exp(-x*l2) + y0_plus
def fit_func_minus(x, A1,l1,A2,l2):
    return A1*np.exp(-x*l1) + A2*np.exp(-x*l2) + y0_minus
popt_plus, pcov_plus=curve_fit(fit_func_plus,t,N, p0=[500,0.02,50,0.001],sigma=Fehler_N)
popt_minus, pcov_minus=curve_fit(fit_func_minus,t,N, p0=[500,0.02,50,0.001],sigma=Fehler_N)
```

In [11]:

```
print("Plus:")
print("A1=",popt_plus[0], ", Standardfehler=", np.sqrt(pcov_plus[0][0]))
print("l1=",popt_plus[1], ", Standardfehler=", np.sqrt(pcov_plus[1][1]))
print("A2=",popt_plus[2], ", Standardfehler=", np.sqrt(pcov_plus[2][2]))
print("l2=",popt_plus[3], ", Standardfehler=", np.sqrt(pcov_plus[3][3]))
print("Minus:")
print("A1=",popt_minus[0], ", Standardfehler=", np.sqrt(pcov_minus[0][0]))
print("l1=",popt_minus[1], ", Standardfehler=", np.sqrt(pcov_minus[1][1]))
print("A2=",popt_minus[2], ", Standardfehler=", np.sqrt(pcov_minus[2][2]))
print("l2=",popt_minus[3], ", Standardfehler=", np.sqrt(pcov_minus[3][3]))
```

Plus:

```
A1= 284.79994214501886 , Standardfehler= 18.896959441802508
l1= 0.028247581622014118 , Standardfehler= 0.003313801144943892
A2= 38.72219384065353 , Standardfehler= 11.344146721353567
l2= 0.0030056904997335537 , Standardfehler= 0.0010738161021219828
```

Minus:

```
A1= 285.4462610125108 , Standardfehler= 18.74277450611554
l1= 0.028155544226493117 , Standardfehler= 0.003226275942930603
A2= 39.67209005327257 , Standardfehler= 10.634639022857264
l2= 0.002733764853249309 , Standardfehler= 0.000973228441493468
```

In [12]:

```
# Differenzen:
l1_dplus = np.abs(popt[1]-popt_plus[1])
l1_dminus = np.abs(popt[1]-popt_minus[1])
print("Differenzen l1:", l1_dplus , l1_dminus)
l2_dplus = np.abs(popt[3]-popt_plus[3])
l2_dminus = np.abs(popt[3]-popt_minus[3])
```

```

print("Differenzen 12:", l2_dplus , l2_dminus)

# Mittelwert der Differenzen
l1_d = (l1_dplus + l1_dminus)/2
l2_d = (l2_dplus + l2_dminus)/2
print("Mittelwerte", l1_d, l2_d)

#Gesamtfehler
l1_err = np.sqrt( np.sqrt(pcov[1][1])**2 + l1_d**2 )
l2_err = np.sqrt( np.sqrt(pcov[3][3])**2 + l2_d**2 )
print("Gesamtfehler:", l1_err, l2_err)

```

Differenzen 11: 4.826305507303619e-05 4.3774340447964855e-05
 Differenzen 12: 0.00014228813919652406 0.00012963750728772067
 Mittelwerte 4.601869776050052e-05 0.00013596282324212236
 Gesamtfehler: 0.0032678309999794953 0.0010300187563033324

```

In [13]: l1 = popt[1]
          l2 = popt[3]

# Halbwertszeiten
T_l1 = np.log(2)/l1
T_l1_err = np.log(2)*l1_err / l1**2

T_l2 = np.log(2)/l2
T_l2_err = np.log(2)*l2_err / l2**2

print("Halbwertszeit 1:", T_l1, "+-", T_l1_err, "[s]")
print("Halbwertszeit 2:", T_l2, "+-", T_l2_err, "[s]")

#Lebensdauer
tau_l1 = 1/l1
tau_l1_err = l1_err / l1**2

tau_l2 = 1/l2
tau_l2_err = l2_err / l2**2

print("Lebensdauer 1:", tau_l1, "+-", tau_l1_err, "[s]")
print("Lebensdauer 2:", tau_l2, "+-", tau_l2_err, "[s]")

```

Halbwertszeit 1: 24.58028121901296 +- 2.8484448929156336 [s]
 Halbwertszeit 2: 242.07117732135484 +- 87.07747693365111 [s]
 Lebensdauer 1: 35.46184981832612 +- 4.109437321254879 [s]
 Lebensdauer 2: 349.2348870636715 +- 125.62624414530157 [s]

```
In [14]: # Sigma Abweichungen  
T_l1_lit = 24.6  
T_l2_lit = 144.6
```

```
sigma_tl1 = np.abs((T_l1 - T_l1_lit)/T_l1_err)  
sigma_tl2 = np.abs((T_l2 - T_l2_lit)/T_l2_err)  
print("Abweichungen:", sigma_tl1, sigma_tl2)
```

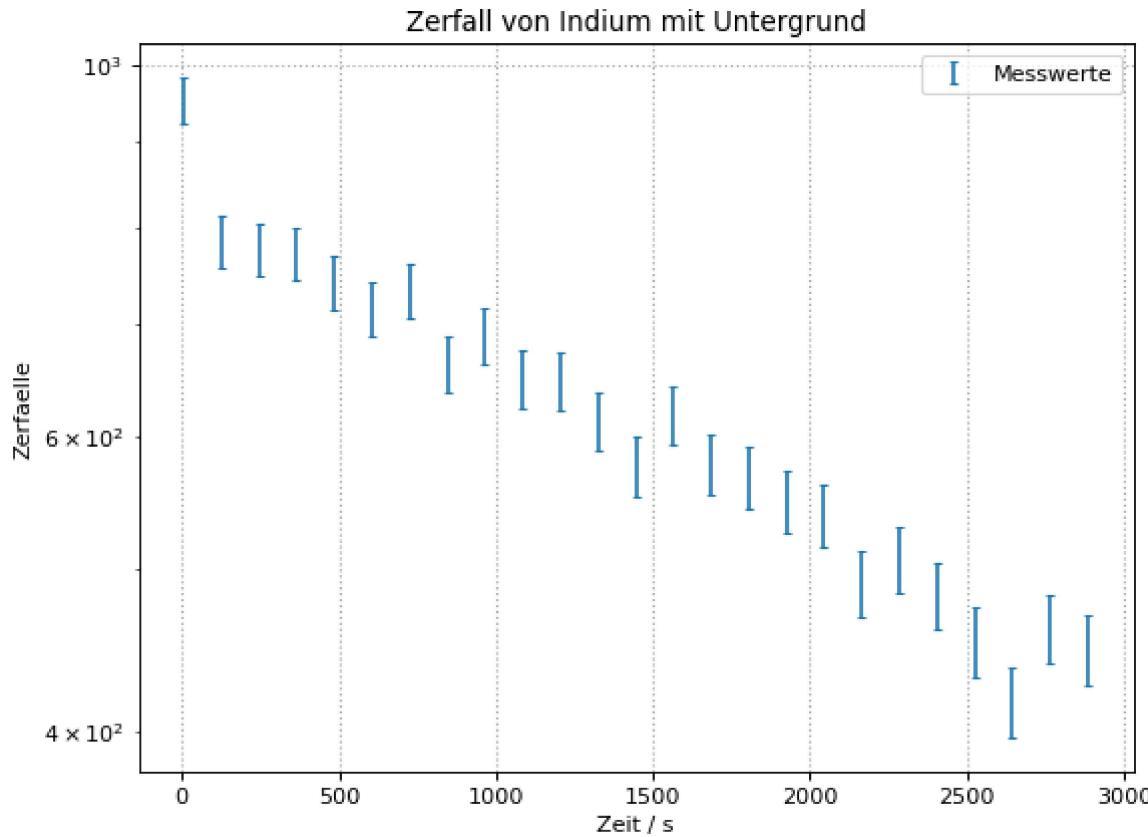
```
Abweichungen: 0.006922647875717715 1.119361524399969
```

```
In [15]: # Indium
```

```
nind =np.loadtxt('C:/Users/fexfl/Documents/.Keine Programme Docs/Studium/PAP2/252/Felix_David_indium.dat', usecols=[1])  
Fehler_nind=np.sqrt(nind)  
  
tind=np.arange(5,3005,120)
```

```
In [16]: plt.figure(figsize=(8, 6), dpi=80)
```

```
plt.errorbar(tind,nind, Fehler_nind, linestyle='None', label="Messwerte", capsize = 2)  
plt.xlabel('Zeit / s')  
plt.ylabel('Zerfaelle')  
plt.title('Zerfall von Indium mit Untergrund')  
plt.yscale('log')  
plt.legend()  
plt.grid(linestyle=":", linewidth=1)
```



```
In [17]: y0=mittelw_unterg *12/4 #Untergrund
nind_corr = nind[1:]
Fehler_nind_corr = Fehler_nind[1:]
tind_corr = tind[1:]

def expfunc(x, A3,l3):
    return A3*np.exp(-x*l3) + y0

poptind, pcovind=curve_fit(expfunc,tind_corr,nind_corr, p0=[500,0.02],sigma=Fehler_nind_corr)
```

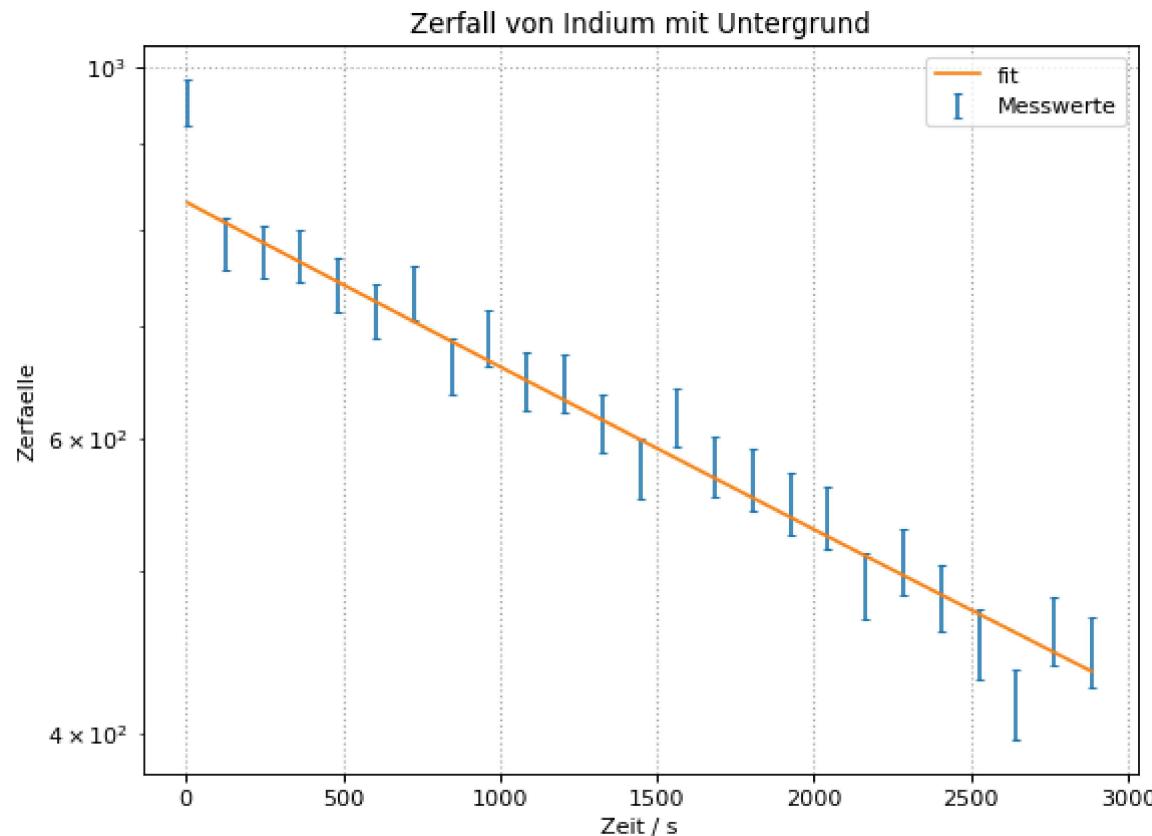
<ipython-input-17-b7769532caa0>:7: RuntimeWarning: overflow encountered in exp
return A3*np.exp(-x*l3) + y0

```
In [18]: plt.figure(figsize=(8, 6), dpi=80)
plt.errorbar(tind,nind, Fehler_nind, linestyle='None', label="Messwerte", capsize = 2)
plt.xlabel('Zeit / s')
```

```

plt.ylabel('Zerfaelle')
plt.title('Zerfall von Indium mit Untergrund')
plt.yscale('log')
plt.plot(tind,expfunc(tind,*poptind), label="fit")
plt.legend()
plt.grid(linestyle=":", linewidth=1)

```



```

In [19]: print("A3=",poptind[0], ", Standardfehler=", np.sqrt(pcovind[0][0]))
print("l3=",poptind[1], ", Standardfehler=", np.sqrt(pcovind[1][1]))

#print(np.log(2)/poptind[1])

```

A3= 797.7878217368338 , Standardfehler= 10.978304078407582
l3= 0.00023684299345154986 , Standardfehler= 8.779103404241653e-06

```

In [20]: chi2_ind=np.sum((expfunc(tind_corr,*poptind)-nind_corr)**2/Fehler_nind_corr**2)
dof_ind=len(nind_corr)-4 #dof:degrees of freedom, Freiheitsgrad

```

```

chi2_red_ind=chi2_ind/dof_ind
print("chi2=", chi2_ind)
print("chi2_red=",chi2_red_ind)

from scipy.stats import chi2
prob_ind=round(1-chi2.cdf(chi2_ind,dof_ind),2)*100
print("Wahrscheinlichkeit=", prob_ind,"%")

chi2= 14.843623005432569
chi2_red= 0.7421811502716285
Wahrscheinlichkeit= 79.0 %

```

In [21]:

```

# Wiederholung mit variiertem Untergrund
def expfunc_plus(x, A3,l3):
    return A3*np.exp(-x*l3) + (y0_plus*12/4)
def expfunc_minus(x, A3,l3):
    return A3*np.exp(-x*l3) + (y0_minus*12/4)
popt_plus_ind, pcov_plus_ind=curve_fit(expfunc_plus,tind_corr,nind_corr, p0=[500,0.02],sigma=Fehler_nind_corr)
popt_minus_ind, pcov_minus_ind=curve_fit(expfunc_minus,tind_corr,nind_corr, p0=[500,0.02],sigma=Fehler_nind_corr)

<ipython-input-21-69387cc9f358>:3: RuntimeWarning: overflow encountered in exp
    return A3*np.exp(-x*l3) + (y0_plus*12/4)
<ipython-input-21-69387cc9f358>:5: RuntimeWarning: overflow encountered in exp
    return A3*np.exp(-x*l3) + (y0_minus*12/4)

```

In [22]:

```

print("Plus:")
print("A3=",popt_plus_ind[0], ", Standardfehler=", np.sqrt(pcov_plus_ind[0][0]))
print("l3=",popt_plus_ind[1], ", Standardfehler=", np.sqrt(pcov_plus_ind[1][1]))
print("Minus:")
print("A3=",popt_minus_ind[0], ", Standardfehler=", np.sqrt(pcov_minus_ind[0][0]))
print("l3=",popt_minus_ind[1], ", Standardfehler=", np.sqrt(pcov_minus_ind[1][1]))

```

Plus:
A3= 795.3063209322307 , Standardfehler= 10.994711369577999
l3= 0.00023793692921360234 , Standardfehler= 8.827303823264e-06
Minus:
A3= 800.2703613107133 , Standardfehler= 10.962117733061122
l3= 0.000235758988490565 , Standardfehler= 8.731470783821266e-06

In [23]:

```

# Differenzen:
l3_dplus = np.abs(poptind[1]-popt_plus_ind[1])
l3_dminus = np.abs(poptind[1]-popt_minus_ind[1])
print("Differenzen l3:", l3_dplus , l3_dminus)

# Mittelwert der Differenzen

```

```

l3_d = (l3_dplus + l3_dminus)/2
print("Mittelwerte", l3_d)

#Gesamtfehler
l3_err = np.sqrt( np.sqrt(pcovind[1][1])**2 + l3_d**2 )
print("Gesamtfehler:", l3_err)

```

Differenzen 13: 1.0939357620524785e-06 1.084004960984848e-06
 Mittelwerte 1.0889703615186633e-06
 Gesamtfehler: 8.846384178331476e-06

```

In [24]: l3 = poptind[1]

# Halbwertszeiten
T_13 = np.log(2)/l3
T_13_err = np.log(2)*l3_err / l3**2

print("Halbwertszeit 3:", T_13, "+-", T_13_err, "[s]")

#Lebensdauer
tau_13 = 1/l3
tau_13_err = l3_err / l3**2

print("Lebensdauer 3:", tau_13, "+-", tau_13_err, "[s]")

```

Halbwertszeit 3: 2926.61045386483 +- 109.31258737238322 [s]
 Lebensdauer 3: 4222.206388404588 +- 157.7047277088788 [s]

```

In [25]: T_13_lit = 3240

sigma_t13 = np.abs((T_13 - T_13_lit)/T_13_err)
print(sigma_t13)

2.8669117955060384

```

In []: