Versuch 252: Aktivierung mit thermischen Neutronen

Matthias Kuntz

24.06.2024

Inhaltsverzeichnis

1 Einleitung

In diesem Versuch soll die Aktivierung radioaktiver Quellen mit thermischen Neutronen untersucht werden. Dazu werden die Präparate ^{116}In und ^{108}Ag sowie ^{110}Ag aktiviert und daraufhin die Zerfallsraten analysiert. Im Endeffekt sollen so die Halbwertszeiten der Präparate bestimmt werden.

1.1 Physikalische Grundlagen

Um eine radioaktive Quelle zu erzeugen werden stabile Isotope durch Kernreaktionen aktiviert. Besonders effektiv ist die Aktivierung mit thermischen Neutronen, da diese nicht die Coulomb-Barriere des Kerns überwinden müssen und viele Kerne einen großen Wirkungsquerschnitt für den Einfach langsamer Neutronen besitzen. Wird ein solches Neutron vom Kern eingefangen so entsteht ein Isotop mit einer um 1 erhöhten Massenzahl, welches radioaktiv sein kann, wie beispielsweise der aus dem stabilen Isotop ^{115}In entstehende β-Strahler ^{116}In.

Es ist zu beachten, dass häufig auch sogenannte Isomere gebildet werden, also Nuklide mit gleicher Anzahl Neutronen und Protonen in einem unterschiedlichen Energiezustand zum primär gebildeten Isotop, die häufig auch verschiedene Halbwertszeiten besitzen.

Allgemein wird bei der Aktivierung eine gewisse Anzahl der Präparatskerne radioaktiv und somit ist die Rate der zerfallenden Kerne ebenso abhängig von der Anzahl der aktivierten Kerne. Die Aktivität $A(t)$ beschreibt die Zahl der Zerfälle pro Sekunde und ist gegeben als

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

wobei Lambda die Zerfallskonstante darstellt, die mit der Halbwertszeit $T_{1/2}$ verbunden ist:

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

Werden keine weiteren Kerne aktiviert so folgen die Zerfälle dem klassischen radioaktiven Zerfallsgesetz:

$$A(t) = A_0 e^{-\lambda t}. \tag{3}$$

Bei der Aktivierung von natürlichem Silber entstehen die beiden radioaktiven Silberisotope ^{108}Ag und ^{110}Ag. Da sich deren Halbwertszeiten unterscheiden

kann durch die Länge der Aktivierungszeit das Verhältnis der beiden Isotope beeinflusst werden, siehe Abbildung 1

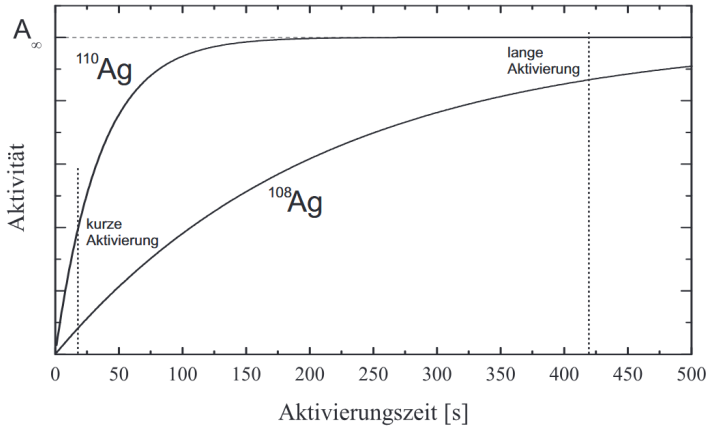


Abbildung 1: Aktivierungsverhältnis bei unterschiedlicher Aktivierungszeit [Quelle: PAP2.2 Skript, S.70, Stand: 31.07.2024]

1.2 Versuchsaufbau

Der Versuchsaufbau, zu sehen in Abbildung 2, besteht aus einem Geiger-Müller-Zählrohr, das auf einer Halterungsschiene angebracht ist, auf welcher ebenso das radioaktive Präparat in verstellbarem Abstand positioniert werden kann. Das Signal des Zählrohrs wird an das Betriebsgerät und über einen externen Zähler an einen Computer gesendet. Am Betriebsgerät können die Zählrohrspannung sowie die Messdauer eingestellt werden.

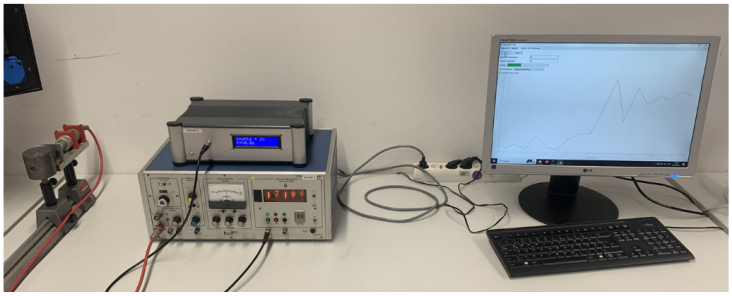


Abbildung 2: Versuchsaufbau [Quelle: PAP2.2 Skript, S.69, Stand: 31.07.2024]

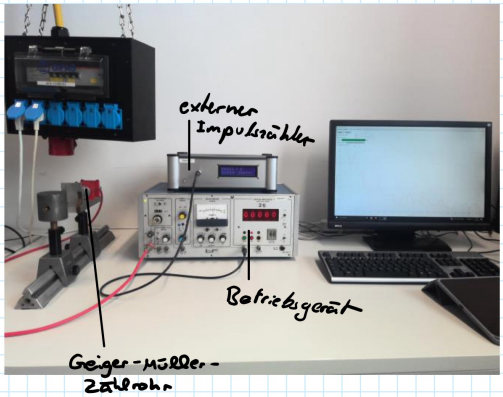
2 Versuchsprotokoll mit Messdaten

Versuch 252 24.6.24

Nicole Schorer
Matthias Kuntz
Tutor: Daniel Härter

Verwendete Instrumente:

- Geiger-Müller Zählrohr mit Betriebsgerät
- Externer Impulszähler
- Neutronenquelle
- Präparatehalterung
- Indium- und Silberbleche



Durchführung:

Es wird zuerst die Halbwertszeit von aktiviertem Silber gemessen. Hierfür beginnen wir mit der Bestimmung der Untergrundzählrate. Die Zählrohrspannung liegt bei (560 ± 10)V . Die Torzeit am Programm wird auf 10s eingestellt bei einem Zeitraum von 8 min. Die Anzahl der Messungen liegt bei 48.

Anschließend wird der Träger mit den Silberblechen in die Neutronenquelle eingelegt und 7 min aktiviert. Es wird möglichst schnell von das Zählrohr gebracht und mit dem Aluminiumblech fixiert. Die Messzeit beträgt 400s und die Torzeit 10s. Dies wird 3 Mal wiederholt bei einer Anzahl von 40 Messungen.

Die Messzeit beträgt 400s und die Totzeit 10s. Dies wird 3 Mal wiederholt bei einer Anzahl von 40 Messungen.

Als letztes wird die Halbwertszeit von aktiviertem Indium gemessen. Das Messintervall beträgt hier 120s bei einem Zeitraum von 50 min bei 25 Messungen.
Hierfür wird das aktivierte Indium-Präparat wieder in die Halterung gesteckt.

t [s]	N_{ug}
0	4
10	3
20	4
30	4
40	9
50	1
60	3
70	5
80	4
90	2
100	5
110	4
120	3
130	4
140	3
150	7
160	2
170	3
180	4
190	3
200	7
210	8
220	4
230	2
240	8
250	5
260	6
270	4
280	1
290	5
300	2
⋮	⋮

t [s]	N_{ug}
⋮	⋮
310	7
320	2
330	4
340	2
350	3
360	2
370	5
380	5
390	6
400	0
410	3
420	3
430	5
440	4
450	1
460	2
470	4

Tabelle 1: Messwerte der Untergrundmessung

t [s]	N_1	N_2	N_3	N_4	t [s]	N_1	N_2	N_3	N_4
					⋮	⋮	⋮	⋮	⋮
0	52	95	102	62	210	11	11	10	8
10	51	85	77	48	220	8	6	3	9
20	36	51	65	29	230	17	12	14	7
30	49	49	52	34	240	10	4	10	5
40	35	47	32	46	250	6	4	7	12
50	33	42	23	25	260	5	6	9	6
60	21	22	31	19	270	7	7	9	8
70	19	23	16	21	280	11	9	10	9
80	16	22	27	17	290	5	8	13	6
90	16	15	16	11	300	4	9	5	6
100	22	9	18	15	310	2	9	7	8
110	15	11	18	9	320	8	5	8	5
120	13	20	14	12	330	7	5	9	8
130	13	17	18	12	340	6	5	4	3
140	14	16	13	6	350	8	6	8	9
150	7	15	14	14	360	3	3	3	8
160	7	16	9	9	370	8	8	8	8
170	14	9	4	5	380	7	10	4	4
180	11	23	19	8	390	4	4	5	3
190	16	8	5	8					
200	7	5	8	11					
⋮	⋮	⋮	⋮	⋮					

Tabelle 2: A1 - Silbermessungen

t [s]	N_{ind}
0	765
120	693
240	694
360	691
480	623
600	609
720	606
840	569
960	591
1080	572
1200	609
1320	543
1440	525
1560	500
1680	460
1800	519
1920	491
2040	504
2160	489
2280	422
2400	430
2520	405
2640	413
2760	431
2880	385

Tabelle 3: A2 - Indiummessung

3 Auswertung

In dieser Evaluation werden alle Fehler, sofern keine spezifische Angabe gemacht wird, mithilfe der Gauss'schen Fehlerfortpflanzung berechnet. Dies bedeutet, dass ein Wert F, der mit der Formel $f(a_1, ..., a_n)$ berechnet wird, den Fehler ΔF annimmt:

$$\Delta F = \sqrt{\sum_n \left(\frac{\partial f}{\partial a_n} \cdot \Delta a_n \right)^2}. \tag{4}$$

Des Weiteren erfolgen Signifikanztests von zwei Werten a und a' über die folgende Formel:

$$\sigma = \frac{|a - a'|}{\sqrt{(\Delta a)^2 + (\Delta a')^2}}. \tag{5}$$

Die Auswertung sowie Berechnung erfolgen über das dem Dokument angehängte Python-Programm. Hierbei erfolgen Fits von Funktionen mithilfe der 'curve_fit'-Funktion des 'SciPy'-Packages und Plots werden mit 'matplotlib' erstellt.

Die Güte eines Fits wird mit der χ^2-Summe bewertet:

$$\chi^2 = \sum_i^N \left(\frac{Funktionswert_i - Messwert_i}{Fehler_i} \right)^2 \tag{6}$$

Auch verwendet wird $\chi^2_{red} = \chi^2/f$, wobei der Freiheitsgrad f die Anzahl der Messwerte minus die Anzahl der Fitparameter ist. Der auf die Freiheitsgrade normierte Wert soll bei einem guten Fit ungefähr 1 sein.

3.1 Zerfall der Silberisotope

Wir beginnen, indem wir aus den Messdaten der Untergrundmessung den Mittelwert der Messrate bestimmen. Der Fehler ergibt sich hierbei einfach aus der Standardabweichung gemäß

$$\sigma_{std} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\overline{x} - x_i)^2}. \tag{7}$$

Beide werden noch mit vier multipliziert, da wir die vier Messreihen der Silbermessungen addieren werden. Somit erhalten wir den vierfachen Mittelwert des Untergrunds:

$$\overline{n}_{bkg} = (15,6 \pm 1,1)\text{counts}/10\text{s}. \tag{8}$$

Nun importieren wir die Daten der vier Silbermessungen und addieren die Raten der vier Messungen aufeinander. Der Fehler ergibt sich statistisch aus $\sqrt{N}$. Wir stellen die resultierenden Raten als Funktion der Zeit dar und fitten die folgende doppelte Zerfallsfunktion:

$$y = \alpha_1 e^{-\lambda_1 t} + \alpha_2 e^{-\lambda_2 t} + n_{bkg}. \tag{9}$$

Somit erhalten wir einmal die Fitparameter für ^{108}Ag und einmal für ^{110}Ag. Wir wiederholen diesen Schritt für drei Werte von n_{bkg}. Einmal nutzen wir einfach den Mittelwert $\overline{n}_{bkg}$. Daraufhin addieren und subtrahieren wir jeweils einmal den Fehler des Mittelwerts, ergo wir verwenden $\overline{n}_{bkg} + 1\sigma$ und $\overline{n}_{bkg} - 1\sigma$. Somit erhalten wir insgesamt drei Plots, dargestellt in den Abbildungen 3 - 5 und drei Sätze an Fit-Parametern, welche in Tabelle 4 eingetragen sind.

$\boldsymbol{n_{bkg}}$	$\boldsymbol{\overline{n}_{bkg}}$	$\boldsymbol{\overline{n}_{bkg} + 1\sigma}$	$\boldsymbol{\overline{n}_{bkg} - 1\sigma}$
$\boldsymbol{\alpha_1}$	273 ± 21	270 ± 22	276 ± 20
$\boldsymbol{\alpha_2}$	67 ± 20	69 ± 22	65 ± 18
$\boldsymbol{\lambda_1}$ [1/s]	0,030 ± 0,004	0,030 ± 0,005	0,029 ± 0,004
$\boldsymbol{\lambda_2}$ [1/s]	0,0059 ± 0,0012	0,0064 ± 0,0013	0,0055 ± 0,0011
$\boldsymbol{\chi^2}$	52,47	52,69	52,31
$\boldsymbol{\chi^2_{red}}$	1,46	1,46	1,45
Wsk. [%]	4,0	4,0	4,0

Tabelle 4: A1 - Silbermessung - Fitparameter und Fitbewertung

Wir machen nun die folgenden vereinfachenden Definitionen: $\lambda_i = \lambda_i(\overline{n}_{bkg})$ und $\lambda_i^{\pm} = \lambda_i(\overline{n}_{bkg} \pm 1\sigma)$. Somit haben wir die Werte unserer Zeitkonstanten gegeben durch die λ_i und können die Fehler folgendermaßen berechnen:

$$\begin{aligned} \xi_i^{\pm} &= |\lambda_i - \lambda_i^{\pm}|, \\ \Xi_i &= \frac{1}{2}(\xi_i^{+} + \xi_i^{-}), \\ \Rightarrow \Delta\lambda_i &= \sqrt{(\Delta\lambda_i')^2 + \Xi_i^2}. \end{aligned} \tag{10}$$

Hierbei bezeichnet $\Delta\lambda_i'$ den ursprünglichen Fehler der Fitparameter, so wie er in Tabelle 4 steht. Wir erhalten somit die folgenden beiden Werte für die Zerfallskonstanten:

$$\begin{aligned} \lambda_1 &= (0,030 \pm 0,004)1/\mathrm{s}, \\ \lambda_2 &= (0,0059 \pm 0,0013)1/\mathrm{s}. \end{aligned} \tag{11}$$

Und somit die folgenden Werte für die Halbwertszeiten gemäß Gleichung 2:

$$\begin{aligned} \boldsymbol{T_1} &= \boldsymbol{(23 \pm 3)}\mathbf{s}, \\ \boldsymbol{T_2} &= \boldsymbol{(117 \pm 25)}\mathbf{s}. \end{aligned} \tag{12}$$

Somit lässt sich identifizieren, dass der Index 1 das Präparat ^{110}Ag und der Index 2 ^{108}Ag bezeichnet. Die jeweiligen Literaturwerte sind gegeben als $T_{1,lit} = 24.6\mathrm{s}$ und $T_{2,lit} = 2,41\mathrm{min} = 144,6\mathrm{s}$, wodurch wir Abweichungen von $0,40\sigma$ für ^{110}Ag und $1,08\sigma$ für ^{108}Ag erhalten, was beides insignifikante Abweichungen sind.

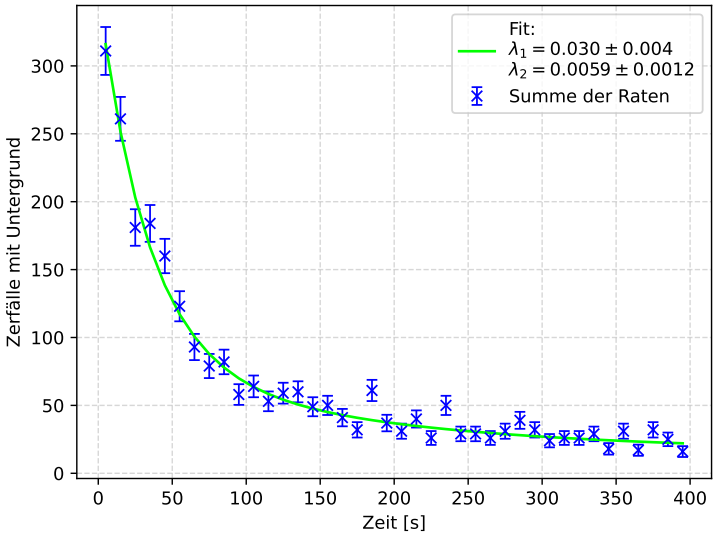


Abbildung 3: A1 - Silberzerfall mit $\overline{n}_{bkg}$

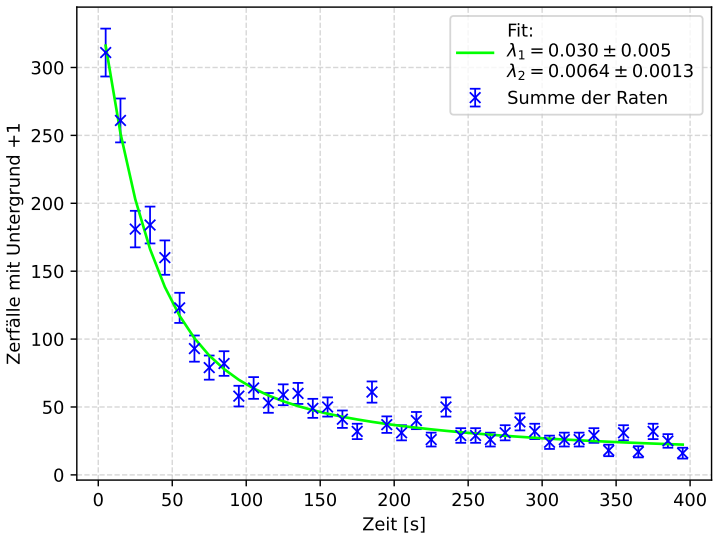


Abbildung 4: A1 - Silberzerfall mit $\overline{n}_{bkg} + 1\sigma$

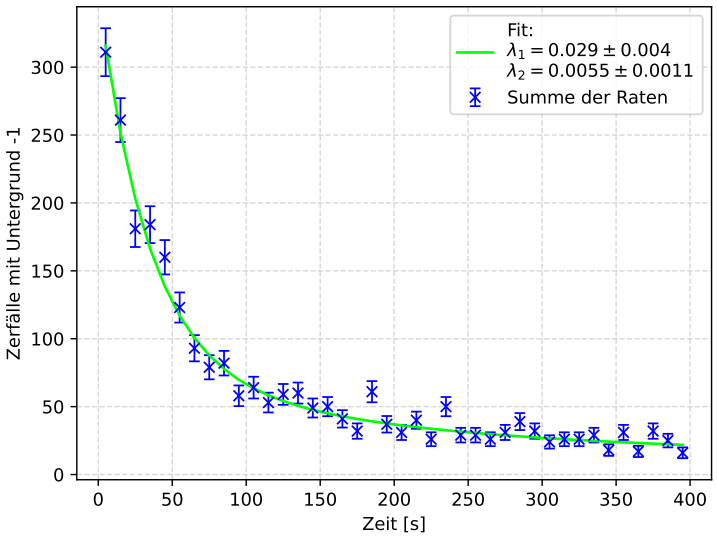


Abbildung 5: A1 - Silberzerfall mit $\overline{n}_{bkg} - 1\sigma$

3.2 Zerfall vom Indiumisotop

Zunächst passen wir den Untergrundwert an die Messdaten der Indium-Messung an. Da wir hier wieder nur eine Messung haben, dafür aber 12-Mal so lange jeweils gemessen haben, müssen wir den Wert von $\overline{n}_{bkg}$ mit dem Faktor 3 multiplizieren.

Nun machen wir genau die gleichen Schritte wie zuvor bei Silber. Da bei Indium allerdings nach kurzer Zeit keine nennenswerten Isomere vorhanden sind wie bei Silber, reicht es eine einfache Exponentialfunktion anzufitten und den ersten stark abweichenden Messwert zu vernachlässigen:

$$y = \alpha e^{-\lambda t} + n_{bkg}. \tag{13}$$

Wir machen wieder unsere drei Fits und erhalten die Diagramme in Abbildung 6 - 8 und die Fitparameter in Tabelle 5.

$\boldsymbol{n_{bkg}}$	$\boldsymbol{\overline{n}_{bkg}}$	$\boldsymbol{\overline{n}_{bkg} + 1\sigma}$	$\boldsymbol{\overline{n}_{bkg} - 1\sigma}$
$\boldsymbol{\alpha}$	675 ± 13	672 ± 13	678 ± 13
$\boldsymbol{\lambda}$ [10^{-5} 1/s]	22,3 ± 1,2	22,5 ± 1,2	22,2 ± 1,2
$\boldsymbol{\chi^2}$	24,81	24,81	24,80
$\boldsymbol{\chi^2_{red}}$	1,13	1,13	1,13
Wsk. [%]	31,0	31,0	31,0

Tabelle 5: A2 - Indiummessung - Fitparameter und Fitbewertung

Wir berechnen analog zu eben die Zerfallskonstante sowie damit die Halbwertszeit und erhalten:

$$\lambda = (22,3 \pm 1,2) \cdot 10^{-5} 1/\text{s} \tag{14}$$

$$\boldsymbol{T = (3,10 \pm 0,17) \cdot 10^3 \text{s}} \tag{15}$$

Der Literaturwert für ^{116}In ist gegeben als $T = 54\text{min} = 3240\text{s}$ womit wir eine Abweichung von $0,83\sigma$ erhalten, was insignifikant ist.

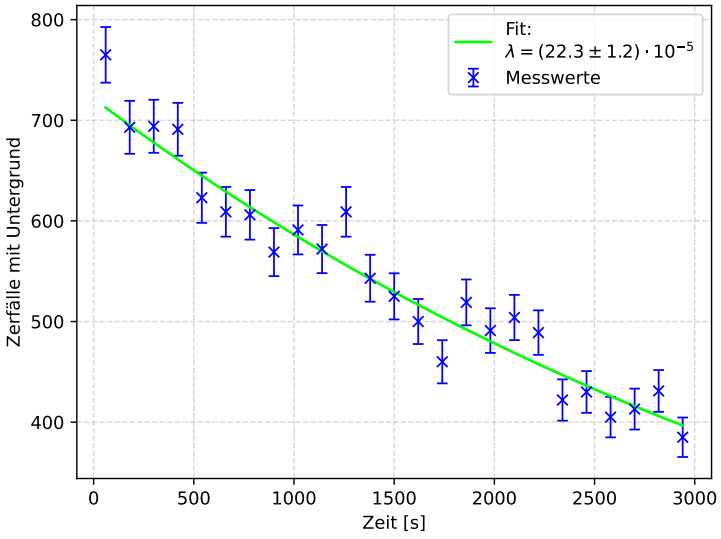


Abbildung 6: A2 - Indiumzerfall mit $\overline{n}_{bkg}$

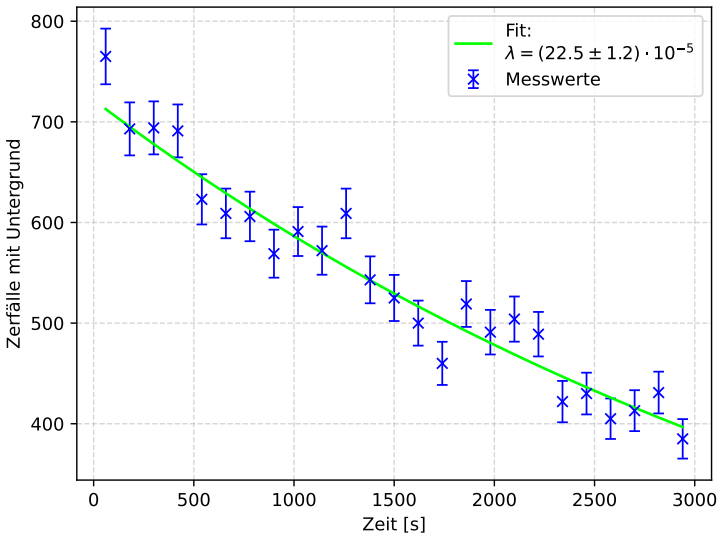


Abbildung 7: A2 - Indiumzerfall mit $\overline{n}_{bkg} + 1\sigma$

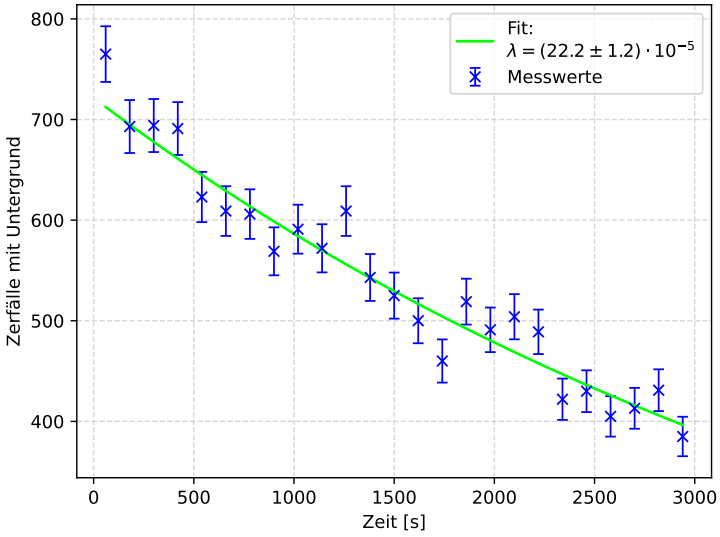


Abbildung 8: A2 - Indiumzerfall mit $\overline{n}_{bkg} - 1\sigma$

4 Zusammenfassung der Endergebnisse

In diesem Versuch wurde der Zerfall von den Präparaten ^{116}In und ^{108}Ag sowie ^{110}Ag nach der Aktivierung mit thermischen Neutronen analysiert. Durch Messung der abnehmenden Zerfallsrate nach der Aktivierung konnten die Halbwertszeiten für die drei Präparate bestimmt werden:

$$\begin{aligned} T_{^{108}Ag} &= (117 \pm 25)\text{s} \\ T_{^{110}Ag} &= (23 \pm 3)\text{s} \\ T_{^{116}In} &= (3,10 \pm 0,17) \cdot 10^3\text{s} \end{aligned} \tag{16}$$

Alle drei Halbwertszeiten lagen hierbei innerhalb insignifikanter Abweichungen von gegebenen Literaturwerten.

5 Diskussion

In diesem Versuch wurden durchweg positive Ergebnisse erzieht. Die drei berechneten Halbwertszeiten liegen alle innerhalb insignifikanter Abweichungen. Als Kritikpunkt an den Werten lässt sich nur nennen, das bei der Halbwertszeit von ^{108}Ag der Fehler etwa 20% des Werts beträgt und somit leicht erhöht ist. Die anderen Werte liegen mit Fehlern von ca. 13 bzw. 5 Prozent allerdings komplett innerhalb akzeptabler Umgebungen.

Des Weiteren lässt sich noch nennen, dass die gemessenen Werte teils stark um den erwarteten Exponentialverlauf schwankten, was an den teilweise leicht erhöhten χ^2_{red}-Werten und den damit verbundenen Fitwahrscheinlichkeiten erkennbar ist. Jedoch halten sich diese Schwankungen insgesamt in Grenzen und treten nicht über die bei einem statistischen Versuch erwarteten Abweichungen hinaus.

Somit lässt sich insgesamt sagen, dass bei diesem Versuch aussagekräftige mit der Theorie übereinstimmende Ergebnisse erzielt wurden und somit eine interessante und lehrreiche Einführung in die Thematik der radioaktiven Aktivierung erfolgen konnte.

252

July 31, 2024

[ ]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
from scipy.optimize import curve_fit
from scipy.stats import chi2
from scipy.stats import norm
import scipy.constants as scp
from scipy.integrate import quad
from tabulate import tabulate
from scipy import signal
import scipy.constants as const
from scipy.special import gamma
```

[ ]:
```
def sigma(x, y, dx, dy, label):
    s = np.abs(x-y)/np.sqrt(dx**2 + dy**2)
    print('Sigmaabweichung {} ='.format(str(label)), s)
    return s
```

---

1 Untergrundmessung

[ ]:
```
ug_t, ug_r = np.loadtxt('./data/untergrund.txt', skiprows=4, unpack=True,␣
↪delimiter=',')
```

[ ]:
```
ug_mw = np.mean(4 * ug_r)
ug_dmw = np.std(4 * ug_r)/np.sqrt(len(ug_r))

print("Mittelwert Untergrund = ({} +/- {})counts/10s".format(ug_mw, ug_dmw))
```

```
Mittelwert Untergrund = (15.583333333333334 +/- 1.1257070823199014)counts/10s
```

2 Zerfall der Silberisotope

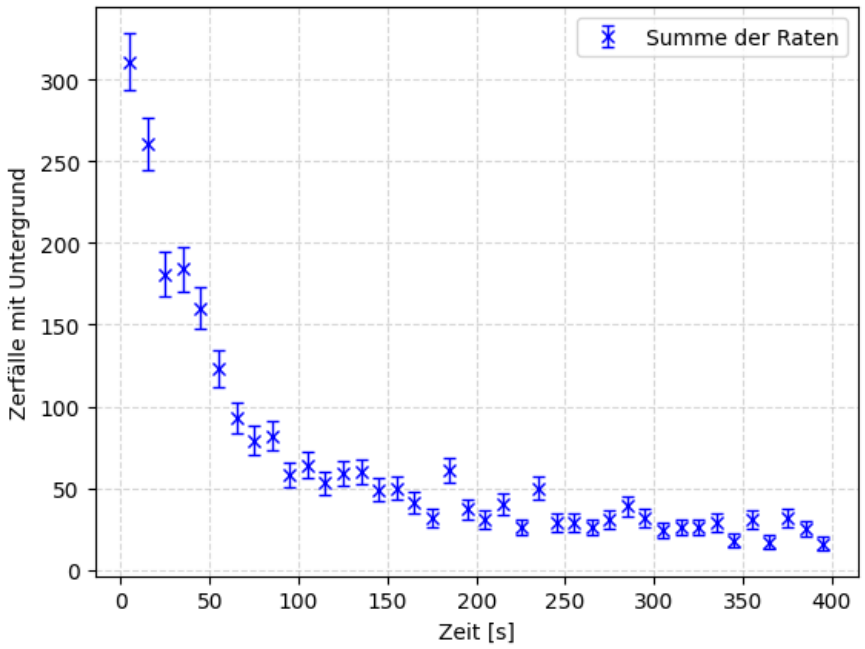
[ ]:
```
n1_t, n1 = np.loadtxt('data/silber1.txt', skiprows=4, delimiter=',',␣
 ↪unpack=True)
n2_t, n2 = np.loadtxt('data/silber2.txt', skiprows=4, delimiter=',',␣
 ↪unpack=True)
n3_t, n3 = np.loadtxt('data/silber3.txt', skiprows=4, delimiter=',',␣
 ↪unpack=True)
n4_t, n4 = np.loadtxt('data/silber4.txt', skiprows=4, delimiter=',',␣
 ↪unpack=True)
```

[ ]:
```
N = n1 + n2 + n3 + n4
dN = np.sqrt(N)
```

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

plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, N, yerr=dN, fmt='x', color='blue', label='Summe der Raten',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
#plt.plot(X, linfit(X, *a1_popt_ac_680), color='lime',
#         label="\n".join([r"Fitgerade: $y = \alpha \cdot x + \beta$",
#                          r'$\alpha ={:.1f}\pm{:.1}$'.
 ↪format(a1_popt_ac_680[0], np.sqrt(a1_pcov_ac_680[0][0])),
#                          r'$\beta ={:.3f}\pm{:.3f}$'.
 ↪format(a1_popt_ac_680[1], np.sqrt(a1_pcov_ac_680[1][1]))]))
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x185be13b040>
```



[ ]:
```
y0 = ug_mw
def exp(x, A1, l1, A2, l2):
    return A1 * np.exp(- x * l1) + A2 * np.exp(-x * l2) + y0
s1_pop, s1_cov = curve_fit(exp, t, N, p0=[500, 0.02, 50, 0.001], sigma=dN,␣
 ↪absolute_sigma=True)

print("A1=",s1_pop[0], ", Standardfehler=", np.sqrt(s1_cov[0][0]))
print("l1=",s1_pop[1], ", Standardfehler=", np.sqrt(s1_cov[1][1]))
print("A2=",s1_pop[2], ", Standardfehler=", np.sqrt(s1_cov[2][2]))
print("l2=",s1_pop[3], ", Standardfehler=", np.sqrt(s1_cov[3][3]))

l1 = s1_pop[1]
dl1 = np.sqrt(s1_cov[1][1])
l2 = s1_pop[3]
dl2 = np.sqrt(s1_cov[3][3])
```

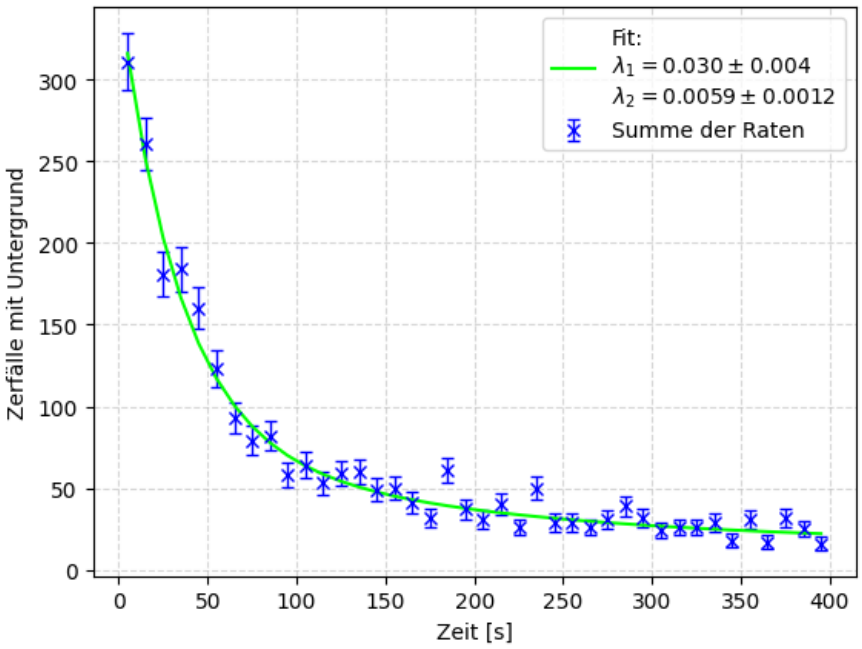
```
A1= 273.20338265849466 , Standardfehler= 21.160044268071253
l1= 0.02977712557206005 , Standardfehler= 0.004253726554796002
A2= 66.94410282445499 , Standardfehler= 19.99547056883678
l2= 0.005907494013643605 , Standardfehler= 0.0011985048945004134
```

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

plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, N, yerr=dN, fmt='x', color='blue', label='Summe der Raten',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
plt.plot(t, exp(t, *s1_pop), color='lime',
        label="\n".join([r"Fit:",
                         r'$\lambda_{} ={:.3f}\pm{:.3f}$'.format('1',␣
 ↪s1_pop[1], np.sqrt(s1_cov[1][1])),
                         r'$\lambda_{} ={:.4f}\pm{:.4f}$'.format('2',␣
 ↪s1_pop[3], np.sqrt(s1_cov[3][3]))]))
plt.legend()
plt.savefig('./plots/Silber_mit_UG.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp(t,*s1_pop) - N)**2 /dN**2)
dof = len(N) - 4 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 52.473667233977125
chi2_red = 1.4576018676104756
Wahrscheinlichkeit = 4.0 %
```



[ ]:
```
y0 = ug_mw - ug_dmw
s2_pop, s2_cov = curve_fit(exp, t, N, p0=[500, 0.02, 50, 0.001], sigma=dN,␣
 ↪absolute_sigma=True)

print("A1=",s2_pop[0], ", Standardfehler=", np.sqrt(s2_cov[0][0]))
print("l1=",s2_pop[1], ", Standardfehler=", np.sqrt(s2_cov[1][1]))
print("A2=",s2_pop[2], ", Standardfehler=", np.sqrt(s2_cov[2][2]))
print("l2=",s2_pop[3], ", Standardfehler=", np.sqrt(s2_cov[3][3]))

l1m = s2_pop[1]
dl1m = np.sqrt(s2_cov[1][1])
l2m = s2_pop[3]
dl2m = np.sqrt(s2_cov[3][3])
```

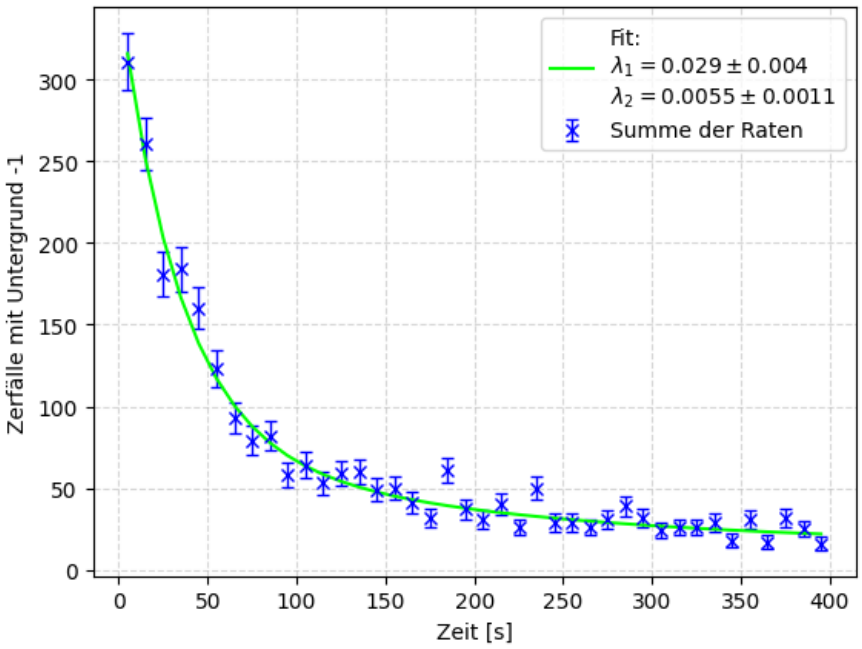
```
A1= 275.8716137602503 , Standardfehler= 20.21726960544724
l1= 0.029493262039445602 , Standardfehler= 0.004020363055628029
A2= 65.14069094119769 , Standardfehler= 18.265484797231323
l2= 0.005497506893630588 , Standardfehler= 0.001106710894171423
```

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

```
plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, N, yerr=dN, fmt='x', color='blue', label='Summe der Raten',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund -1')
plt.plot(t, exp(t, *s2_pop), color='lime',
        label="\n".join([r"Fit:",
                         r'$\lambda_{} ={:.3f}\pm{:.3f}$'.format('1',␣
 ↪s2_pop[1], np.sqrt(s2_cov[1][1])),
                         r'$\lambda_{} ={:.4f}\pm{:.4f}$'.format('2',␣
 ↪s2_pop[3], np.sqrt(s2_cov[3][3]))]))
plt.legend()
plt.savefig('./plots/Silber_mit_UG-1.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp(t,*s2_pop) - N)**2 /dN**2)
dof = len(N) - 4 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 52.306929117782545
chi2_red = 1.4529702532717375
Wahrscheinlichkeit = 4.0 %
```



[ ]:
```
y0 = ug_mw + ug_dmw
s3_pop, s3_cov = curve_fit(exp, t, N, p0=[500, 0.02, 50, 0.001], sigma=dN,␣
 ↪absolute_sigma=True)

print("A1=",s3_pop[0], ", Standardfehler=", np.sqrt(s3_cov[0][0]))
print("l1=",s3_pop[1], ", Standardfehler=", np.sqrt(s3_cov[1][1]))
print("A2=",s3_pop[2], ", Standardfehler=", np.sqrt(s3_cov[2][2]))
print("l2=",s3_pop[3], ", Standardfehler=", np.sqrt(s3_cov[3][3]))

l1p = s3_pop[1]
dl1p = np.sqrt(s3_cov[1][1])
l2p = s3_pop[3]
dl2p = np.sqrt(s3_cov[3][3])
```

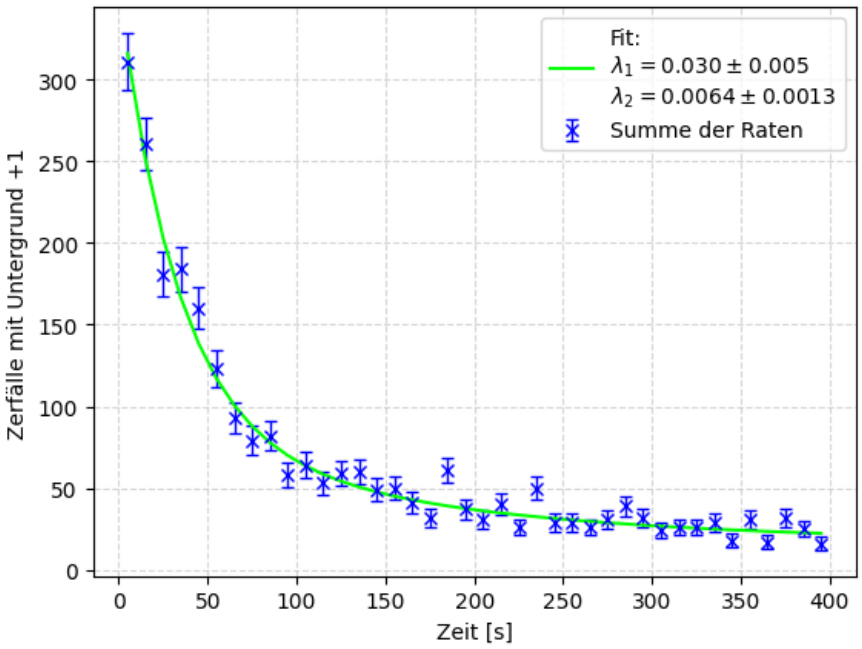
```
A1= 269.9621392034454 , Standardfehler= 22.438095483720453
l1= 0.03010727479762373 , Standardfehler= 0.0045444916051667035
A2= 69.32741599674702 , Standardfehler= 22.15348902417644
l2= 0.006374707676434155 , Standardfehler= 0.0013063042470345188
```

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

```
plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, N, yerr=dN, fmt='x', color='blue', label='Summe der Raten',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund +1')
plt.plot(t, exp(t, *s3_pop), color='lime',
        label="\n".join([r"Fit:",
                         r'$\lambda_{} ={:.3f}\pm{:.3f}$'.format('1',␣
 ↪s3_pop[1], np.sqrt(s3_cov[1][1])),
                         r'$\lambda_{} ={:.4f}\pm{:.4f}$'.format('2',␣
 ↪s3_pop[3], np.sqrt(s3_cov[3][3]))]))
plt.legend()
plt.savefig('./plots/Silber_mit_UG+1.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp(t,*s3_pop) - N)**2 /dN**2)
dof = len(N) - 4 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 52.6853292794104
chi2_red = 1.463481368872511
Wahrscheinlichkeit = 4.0 %
```



[ ]:
```
diff1m = np.abs(l1 - l1m)
diff1p = np.abs(l1 - l1p)

diff1 = np.mean([diff1m, diff1p])

diff2m = np.abs(l2 - l2m)
diff2p = np.abs(l2 - l2p)

diff2 = np.mean([diff2m, diff2p])
```

[ ]:
```
l1 = l1
dl1 = np.sqrt(dl1**2 + diff1**2)

l2 = l2
dl2 = np.sqrt(dl2**2 + diff2**2)

print("Zerfallskonstante 1 = ({} +/- {})".format(l1, dl1))
print("Zerfallskonstante 2 = ({} +/- {})".format(l2, dl2))
```

```
Zerfallskonstante 1 = (0.02977712557206005 +/- 0.004264791028852181)
Zerfallskonstante 2 = (0.005907494013643605 +/- 0.0012762383341207255)
```

[ ]:
```
T1 = np.log(2)/l1
dT1 = T1 * np.sqrt((dl1/l1)**2)

T2 = np.log(2)/l2
dT2 = T2 * np.sqrt((dl2/l2)**2)

print("Halbwertszeit 1 = ({} +/- {})".format(T1, dT1))
print("Halbwertszeit 2 = ({} +/- {})".format(T2, dT2))
```

```
Halbwertszeit 1 = (23.27784053173779 +/- 3.333939175242526)
Halbwertszeit 2 = (117.33353922307714 +/- 25.348406665957704)
```

[ ]:
```
T1_lit = 24.6
T2_lit = 144.6

_ = sigma(T1, T1_lit, dT1, 0, 'Halbwertszeit 1')
_ = sigma(T2, T2_lit, dT2, 0, 'Halbwertszeit 2')
```

```
Sigmaabweichung Halbwertszeit 1 = 0.3965757618136601
Sigmaabweichung Halbwertszeit 2 = 1.0756676400312393
```

[ ]:
```

```

3 Indiumzerfall

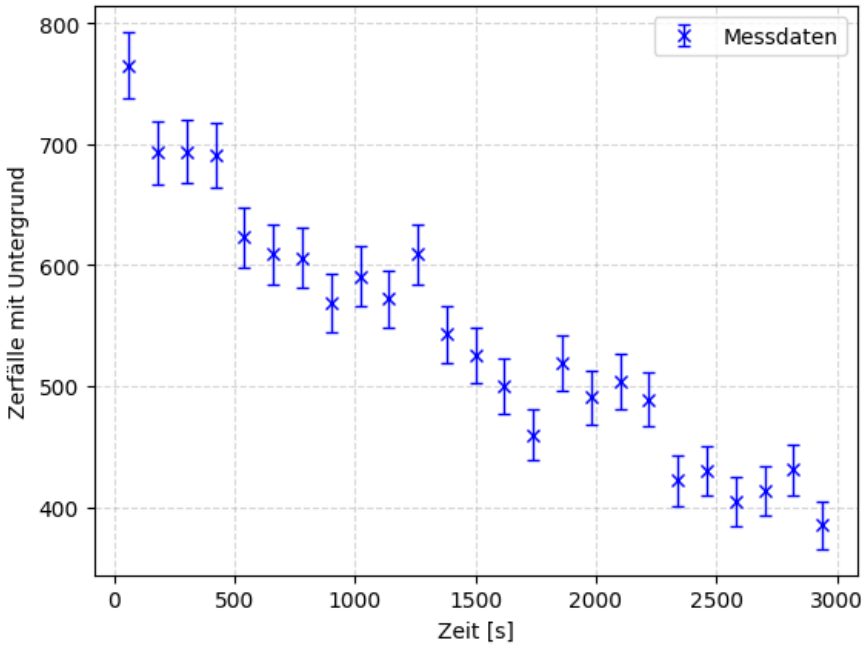
[ ]:
```
ug_i = ug_mw * 3
ug_di = ug_dmw * 3
```

[ ]:
```
T_ind, Ni = np.loadtxt('data/indium.txt', skiprows=4, delimiter=',',␣
 ↪unpack=True)
dNi = np.sqrt(Ni)
```

[ ]:
```
t = np.arange(60,3060,120)

plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, Ni, yerr=dNi, fmt='x', color='blue', label='Messdaten',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
#plt.plot(X, linfit(X, *a1_popt_ac_680), color='lime',
#         label="\n".join([r"Fitgerade: $y = \alpha \cdot x + \beta$",
#                          r'$\alpha ={:.1f}\pm{:.1}$'.
 ↪format(a1_popt_ac_680[0], np.sqrt(a1_pcov_ac_680[0][0])),
#                          r'$\beta ={:.3f}\pm{:.3f}$'.
 ↪format(a1_popt_ac_680[1], np.sqrt(a1_pcov_ac_680[1][1]))]))
plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x185bc8aa790>



[ ]:
```
def exp2(x,A,l):
    return A * np.exp(-x*l) + y0
```

[ ]:
```
y0 = ug_i
i1_pop, i1_cov = curve_fit(exp2, t[1:], Ni[1:], p0=[500, 0.0002], sigma=dNi[1:
 ↪], absolute_sigma=True)

print("A=",i1_pop[0], ", Standardfehler=", np.sqrt(i1_cov[0][0]))
print("l=",i1_pop[1], ", Standardfehler=", np.sqrt(i1_cov[1][1]))

li = i1_pop[1]
dli = np.sqrt(i1_cov[1][1])
```

```
A= 674.8062732895619 , Standardfehler= 12.917666736826769
l= 0.00022342259049030073 , Standardfehler= 1.1838945894853908e-05
```

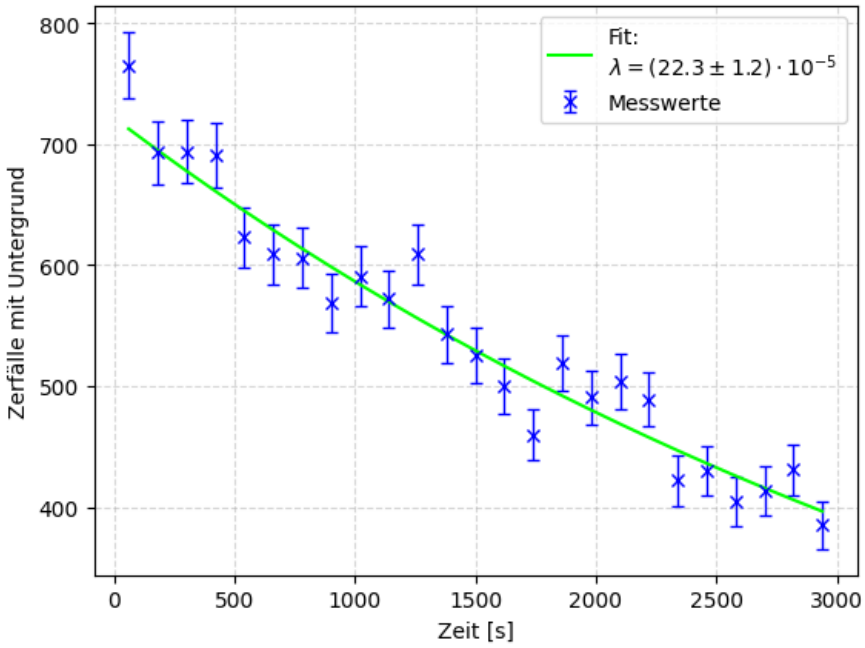
[ ]:
```
t = np.arange(60,3060,120)

plt.grid(alpha=0.5, linestyle='--')
```

```
plt.errorbar(t, Ni, yerr=dNi, fmt='x', color='blue', label='Messwerte',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
plt.plot(t, exp2(t, *i1_pop), color='lime',
        label="\n".join([r"Fit:",
                         r'$\lambda =({:.1f}\pm{:.1f}) \cdot 10^{}$'.
 ↪format(i1_pop[1] * 1e5, np.sqrt(i1_cov[1][1]) * 1e5, '{-5}')]))
plt.legend()
plt.savefig('./plots/Indium_mit_UG.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp2(t[1:],*i1_pop) - Ni[1:])**2 /dNi[1:]**2)
dof = len(Ni[1:]) - 2 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 24.805606637027854
chi2_red = 1.127527574410357
Wahrscheinlichkeit = 31.0 %
```



[ ]:
```
y0 = ug_i - ug_di
i2_pop, i2_cov = curve_fit(exp2, t[1:], Ni[1:], p0=[500, 0.0002], sigma=dNi[1:
 ↪], absolute_sigma=True)

print("A=",i2_pop[0], ", Standardfehler=", np.sqrt(i2_cov[0][0]))
print("l=",i2_pop[1], ", Standardfehler=", np.sqrt(i2_cov[1][1]))

lim = i2_pop[1]
dlim = np.sqrt(i2_cov[1][1])
```

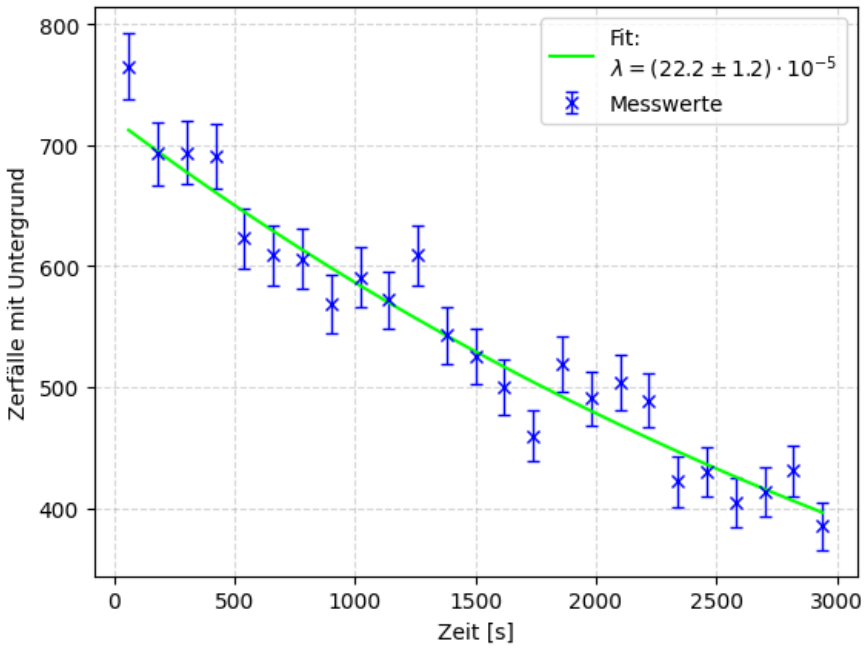
```
A= 678.0136906417098 , Standardfehler= 12.900931727309121
l= 0.00022185611816818426 , Standardfehler= 1.1753511837010672e-05
```

[ ]:
```
t = np.arange(60,3060,120)

plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, Ni, yerr=dNi, fmt='x', color='blue', label='Messwerte',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
plt.plot(t, exp2(t, *i2_pop), color='lime',
        label="\n".join([r"Fit:",
                         r'$\lambda =({:.1f}\pm{:.1f}) \cdot 10^{}$'.
 ↪format(i2_pop[1] * 1e5, np.sqrt(i2_cov[1][1]) * 1e5, '{-5}')]))
plt.legend()
plt.savefig('./plots/Indium_mit_UG-1.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp2(t[1:],*i2_pop) - Ni[1:])**2 /dNi[1:]**2)
dof = len(Ni[1:]) - 2 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 24.804530106461538
chi2_red = 1.1274786412027973
Wahrscheinlichkeit = 31.0 %
```



[ ]:
```
y0 = ug_i + ug_di
i3_pop, i3_cov = curve_fit(exp2, t[1:], Ni[1:], p0=[500, 0.0002], sigma=dNi[1:
 ↪], absolute_sigma=True)

print("A=",i3_pop[0], ", Standardfehler=", np.sqrt(i3_cov[0][0]))
print("l=",i3_pop[1], ", Standardfehler=", np.sqrt(i3_cov[1][1]))

lip = i3_pop[1]
dlip = np.sqrt(i3_cov[1][1])
```

```
A= 671.6011821783825 , Standardfehler= 12.934649820289511
l= 0.00022501119626819482 , Standardfehler= 1.1925658317947085e-05
```

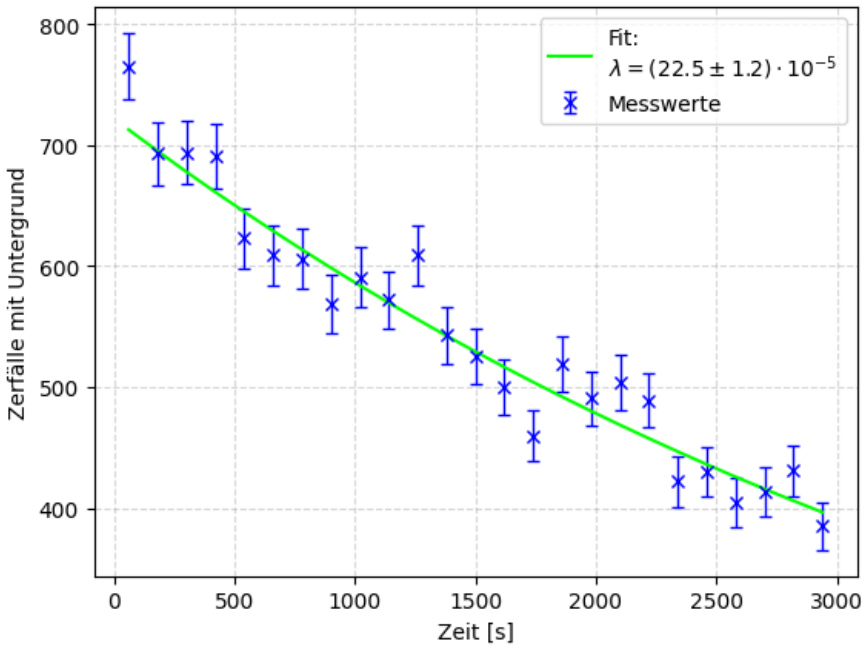
[ ]:
```
t = np.arange(60,3060,120)

plt.grid(alpha=0.5, linestyle='--')
plt.errorbar(t, Ni, yerr=dNi, fmt='x', color='blue', label='Messwerte',␣
 ↪capsize=3, lw=1)
plt.xlabel(r'Zeit [s]')
plt.ylabel(r'Zerfälle mit Untergrund')
plt.plot(t, exp2(t, *i3_pop), color='lime',
```

```
        label="\n".join([r"Fit:",
                         r'$\lambda =({:.1f}\pm{:.1f}) \cdot 10^{}$'.
 ↪format(i3_pop[1] * 1e5, np.sqrt(i3_cov[1][1]) * 1e5, '{-5}')]))
plt.legend()
plt.savefig('./plots/Indium_mit_UG+1.pdf', format='PDF')

#Fitcheck
chi_2 = np.sum((exp2(t[1:],*i3_pop) - Ni[1:])**2 /dNi[1:]**2)
dof = len(Ni[1:]) - 2 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi_2/dof
print("chi2 =", chi_2)
print("chi2_red =",chi2_red)
prob = round(1-chi2.cdf(chi_2,dof),2)*100
print("Wahrscheinlichkeit =", prob,"%")
```

```
chi2 = 24.80693417316548
chi2_red = 1.1275879169620673
Wahrscheinlichkeit = 31.0 %
```



[ ]:
```
diffim = np.abs(li - lim)
diffip = np.abs(li - lip)
```

```
diffi = np.mean([diffim, diffip])

li = li
dli = np.sqrt(dli**2 + diffi**2)

print("Zerfallskonstante Indium = ({} +/- {})".format(li, dli))

Ti = np.log(2)/li
dTi = Ti * np.sqrt((dli/li)**2)

print("Halbwertszeit Indium = ({} +/- {})".format(Ti, dTi))
```

```
Zerfallskonstante Indium = (0.00022342259049030073 +/- 1.1943586955164254e-05)
Halbwertszeit Indium = (3102.4041885775036 +/- 165.84640843625795)
```

[ ]:
```
Ti_lit = 54 * 60
_ = sigma(Ti, Ti_lit, dTi, 0, 'Halbwertszeit Indium')
```

```
Sigmaabweichung Halbwertszeit Indium = 0.8296580717054266
```

4 Tabellen für die Auswertung

[ ]:
```
head = ['Time', 'N_ug']
tab = zip(ug_t, ug_r)

#print(tabulate(tab, headers=head, tablefmt="latex"))
```

[ ]:
```
head = ['Time', 'N1', 'N2', 'N3', 'N4']
tab = zip(n1_t, n1, n2, n3, n4)

#print(tabulate(tab, headers=head, tablefmt="latex"))
```

[ ]:
```
head = ['Time', 'N_In']
tab = zip(T_ind, Ni)

#print(tabulate(tab, headers=head, tablefmt="latex"))
```