Versuch 252

Frauendorfer Andreas

06.12.2021

1 Einleitung

1.1 Aktivierung

Eine Neutronenquelle setzt aus Berylliumspäne (Be) und Heliumkernen Kohlenstoff und Neutronen mit der Energie 1 bis 10MeV frei.

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

Diese Neutronen sind sehr schnell und werden durch einen Paraffinblock abgebremst. Stoßen die Neutronen auf Wasserstoffkerne, also ein Proton, verlieren sie im Mittel die Hälfte ihrer kinetischen Energie. Stoßen sie auf schwerere Kohlenstoffkerne, werden sie nur ein bisschen abgebremst, der Stoß ist elastisch. Die langsamen Neutronen können nun von Atomen eingefangen werden. Dabei entstehen Isotope mit um 1 erhöhter Massenzahl.
In unserem Fall bestrahlen wir ^{115}Indium, das durch hinzufügen eines Neutrons zum radioaktiven β-Strahler ^{116}Indium wird. Dabei werden zwei Isomere erzeugt. Zum Einen wird ^{116}In im Grundzustand gebildet, zum anderen das metastabile ^{116m}In. Beide emittieren ein Elektron und ein Elektron-Antineutrino um zum stabilen ^{116}Sn zu werden.

Aktivierung heißt, dass eine festgelegte Anzahl an radioaktiven Kernen pro Sekunde erzeugt wird (durch Neutronenaufnahme). Dem wirkt jedoch der Zerfall der Kerne entgegen.

$$A(t) = A_\infty(1 - e^{-\lambda t}) \tag{2}$$

Erst nimmt die Anzahl der Atomkerne zu, bis ein Gleichgewicht eintritt, bei dem pro Sekunde gleichviele Kerne zerfallen und gebildet werden. Hört die Aktivierung auf, dann zerfallen die Kerne nach dem Zerfallsgesetz:

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

wobei die Zerfallskonstante λ eine charakteristische Größe für ein Isotop ist.
Die Halbwertszeit ist:

$$T_{1/2} = \frac{ln2}{\lambda} \tag{4}$$

Bei Silber werden zwei unterschiedliche Isotope erzeugt, da es zu 49% aus ^{109}Ag und zu 51% aus ^{107}Ag besteht. Durch Neutronabsorption entstehen die radioaktiven Isotope ^{110}Ag und ^{108}Ag, die zu ^{110}Cd und ^{108}Cd zerfallen.
Man kann aufgrund der unterschiedlich langen Halbwertszeiten der Silberisotope und mit Variation der Aktivierungszeiten das Verhältnis zwischen ^{108}Ag und ^{110}Ag beeinflussen.

Messprotokoll

Mo, 22. Nov 2021
14:28 Uhr

Torzeit: 10 s

Messintervall	counts
1	4
2	7
3	4
4	4
5	6
6	4
7	2
8	1
9	6
10	5
11	4
12	2
13	2
14	2
15	5
16	6
17	3
18	3
19	3
20	0
21	7
22	3
23	5
24	5
25	3
26	3
27	2
28	3
29	2
30	6
31	5
32	3
33	9
34	4
35	4

36	3
37	9
38	6
39	4
40	3
41	3
42	5
43	3
44	6
45	3
46	0
47	6
48	5

Messprotokoll

Mo, 22. Nov 2021
14:40 Uhr

Torzeit: 10 s

Messintervall	counts
1	56
2	50
3	38
4	40
5	39
6	26
7	22
8	27
9	23
10	13
11	20
12	10
13	11
14	18
15	14
16	11
17	18
18	15
19	14
20	13
21	7
22	17
23	7
24	18
25	11
26	7
27	12
28	14
29	7
30	7
31	8
32	10
33	8
34	5
35	4

36	5
37	5
38	10
39	5
40	4
41	4
42	12

Messprotokoll

Mo, 22. Nov 2021
14:55 Uhr

Torzeit: 10 s

Messintervall	counts
1	81
2	61
3	39
4	41
5	36
6	40
7	19
8	17
9	19
10	20
11	17
12	8
13	20
14	17
15	15
16	10
17	12
18	21
19	7
20	8
21	19
22	4
23	3
24	5
25	7
26	9
27	6
28	7
29	7
30	8
31	4
32	7
33	5
34	4
35	9

36	11
37	6
38	6
39	5
40	10
41	5
42	6
43	4

Messprotokoll

Mo, 22. Nov 2021
15:11 Uhr

Torzeit: 10 s

Messintervall	counts
1	66
2	53
3	51
4	39
5	45
6	37
7	22
8	32
9	23
10	18
11	13
12	13
13	15
14	10
15	13
16	12
17	17
18	8
19	10
20	9
21	12
22	10
23	12
24	7
25	9
26	7
27	9
28	1
29	6
30	6
31	5
32	14
33	5
34	8
35	9

36	6
37	7
38	7
39	10
40	7

Messprotokoll

Mo, 22. Nov 2021
15:26 Uhr

Torzeit: 10 s

Messintervall	counts
1	77
2	50
3	66
4	33
5	33
6	21
7	20
8	23
9	17
10	16
11	24
12	20
13	16
14	8
15	16
16	16
17	12
18	16
19	9
20	14
21	19
22	11
23	12
24	10
25	4
26	5
27	12
28	5
29	5
30	9
31	8
32	5
33	9
34	11
35	7

36	6
37	13
38	7
39	8
40	5

Messprotokoll

Mo, 22. Nov 2021
16:19 Uhr

Torzeit: 120 s

Messintervall	counts
1	904
2	777
3	719
4	764
5	633
6	705
7	661
8	625
9	661
10	614
11	593
12	586
13	603
14	549
15	577
16	541
17	589
18	450
19	489
20	537
21	472
22	481
23	483
24	433
25	427

2 Auswertung

2.1 Silber

In diesem Versuchteil haben wir Silber durch thermische Neutronen aktiviert und dann die Zerfälle pro Zeit gemessen. Da immer Untergrundstrahlung U existiert, haben wir diese zuerst gemessen. Dabei ist zu beachten, dass sich im Raum auch andere Radioaktivitätsversuche befanden. Wir haben eine Spannung im Plateaubereich des Geiger-Müllerzählrohrs gewählt: $(520 \pm 10)V$
Der Fehler der Hintergrundstrahlung ist:

$$4\Delta U_g = \frac{\Delta(4U)}{\sqrt{n}} \tag{5}$$

Der Fehler wird 4 mal so groß benötigt, da wir die Messreihen später addieren. Die Untergrundstrahlung mit Standardabweichung als Fehler ergibt sich also zu (siehe Pyhton skript):

$$4U_g = (16,1 \pm 1,2)\frac{1}{10s} \tag{6}$$

Da zwei verschiedene Isotope bei der Aktivierung von Silbe produziert werden, muss die Fitfunktion aus zwei unabhängigen e-Funktionen bestehen.

Der Index 1 steht im Folgenden für das Isotop ^{108}Ag und der Index 2 für das Isotop ^{110}Ag.

$$f(t) = A_1 e^{-\lambda_1 t} + A_2 e^{-\lambda_2 t} + 4U_g \tag{7}$$

mit den Zerfallskonstanten λ_1 und λ_2.

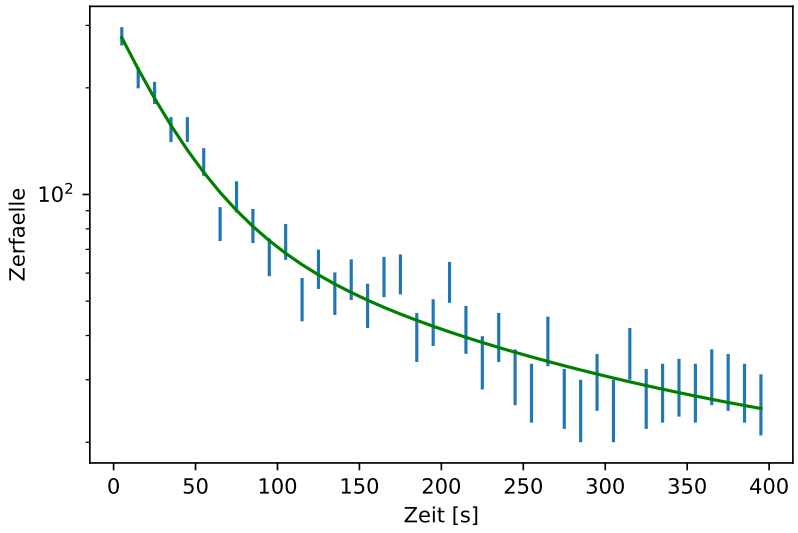


Figure 1: Silbermessung ohne Hintergrundfehler

Die besten Fitparameter sind:

$$A_1 = 220 \pm 20 \tag{8}$$

$$\lambda_1 = 0,028 \pm 0,005)\frac{1}{s} \tag{9}$$

$$A_2 = 72 \pm 20 \tag{10}$$

$$\lambda_2 = (0,0053 \pm 0,0011)\frac{1}{s} \tag{11}$$

Die χ^2-Summe ist: 35,57
Außerdem haben wir die χ^2_{red}-Summe berechnet zu: 0,987
Die Fitwahrscheinlichkeit ist: 49.0%
Da der ideale Wert für die χ^2_{red}-Summe 1 ist, passt unsere Fitfunktion sehr gut zu unseren Daten. Die Fitwahrscheinlichkeit ist physikalisch.
Wir haben bis jetzt den Fehler der Hintergrundstrahlung vernachlässigt.
Im Folgenden wollen wir die Zerfallskonstante unter Berücksichtigung dieses Fehlers ausrechnen. Die korrigierte Hintergrundstrahlung ist:

$$(16,08 - 1,12))\frac{1}{10s} = 14,96\frac{1}{10s} \tag{12}$$

$$(16,08 + 1,12))\frac{1}{10s} = 17,20\frac{1}{10s} \tag{13}$$

Dafür untersuchen wir die Abweichung an den Rändern des 1σ-Bereichs also des Fehlers des Mittelwerts der Untergrundstrahlung. mit der korrigierten Hintergrundstrahlung ergeben sich bem Fitten andere Werte für die Zerfallskonstanten. Der Unterschied zwischen diesen und den ursprünglich berechneten Zerfallskonstanten ist im folgenden Dargestellt. Dabei steht λ_1^- für den abgezogenen Fehler, λ_1^+ für den draufaddierten Fehler zum Mittelwert der Untergrundspannung. Diese Werte haben wir durch 2 extra Fits für die jeweilige Korrektur erhalten.

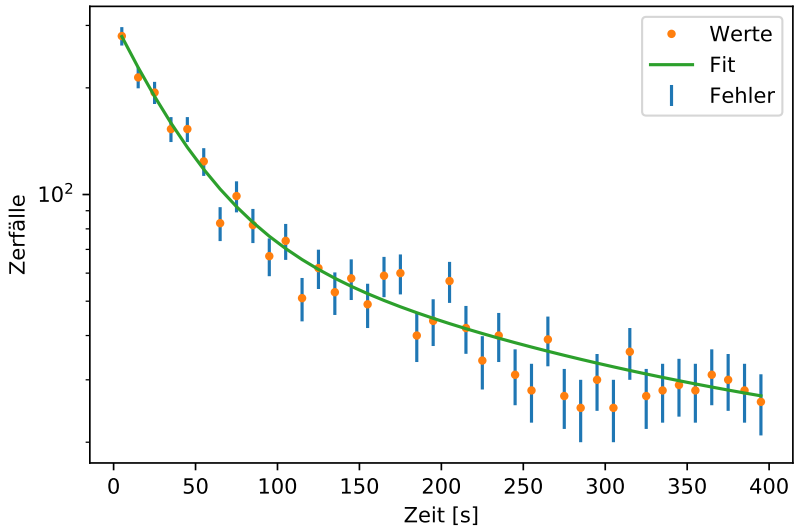


Figure 2: Silbermessung mit positivem Hintergrundstrahlungsfehler

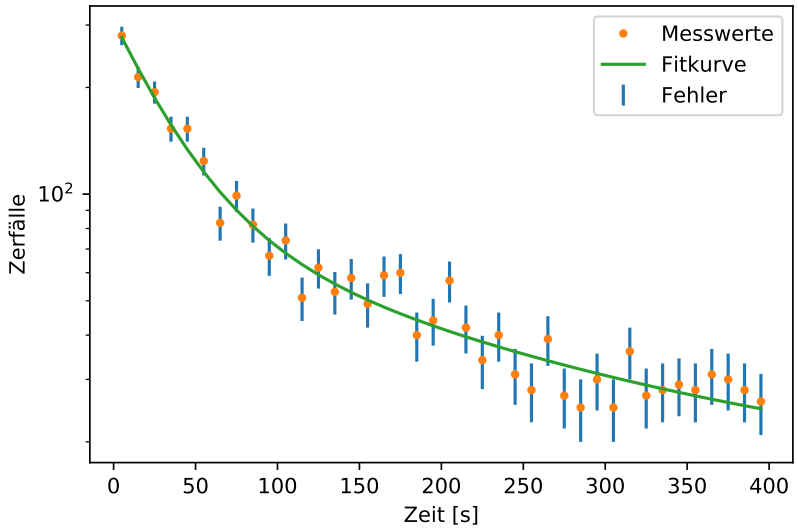


Figure 3: Silbermessung mit negativem Hintergrundstrahlungsfehler

$$|\lambda_1 - \lambda_1^-| = 0,0004 \tag{14}$$
$$|\lambda_1 - \lambda_1^+| = 0,0004 \tag{15}$$
$$|\lambda_2 - \lambda_2^-| = 0,0003 \tag{16}$$
$$|\lambda_2 - \lambda_2^+| = 0,0003 \tag{17}$$

Wenn wir den Mittelwert beider Werte berechnen, bekommen wir die Fehler beider Zerfallskonstanten resultierend von der Hintergrundstrahlung:

$$|\lambda_1 - \lambda_1^\pm| = 0,0004 \tag{18}$$
$$|\lambda_2 - \lambda_2^\pm| = 0,0003 \tag{19}$$

Quadratische Addition mit den Fehlern des Fits führt zu totalen Fehlern von

$$\Delta\lambda_1 = 0,005 \tag{20}$$
$$\Delta\lambda_2 = 0,0012 \tag{21}$$

Da wir nun alle nötigen Parameter für die Halbwertszeit kennen, können wir nun Gleichung 4 ausführen. Der Fehler pflanzt sich dabei folgendermaßen fort:

$$\Delta T_{1/2} = \frac{ln(2)\Delta\lambda_i}{\lambda_i^2} \tag{22}$$

mit i = 1,2. Letztendlich erhalten wir folgende Werte für die Halbwertszeiten:

$$T_{1/2}^1 = 25 \pm 5 \tag{23}$$
$$T_{1/2}^2 = 130 \pm 30 \tag{24}$$

Für eine Einschätzung der Genauigkeit, siehe Diskussion.

2.2 Indium

Im zweiten Teil des Experiments haben wir die gleichen Rechnungen durchgezogen, diesmal mit Indium. Da Indium nicht in zwei verschiedene Produkte zerfällt, reicht für die Fitfunktion eine e-Funktion:

$$f(t) = A_1 e^{-\lambda_1 t} + 12U_g \tag{25}$$

Dabei steht 12 für die Messzeit von 120s. Der Untergrundstrom für diese 2 Minuten wurde zu 48 ± 4 bestimmt.

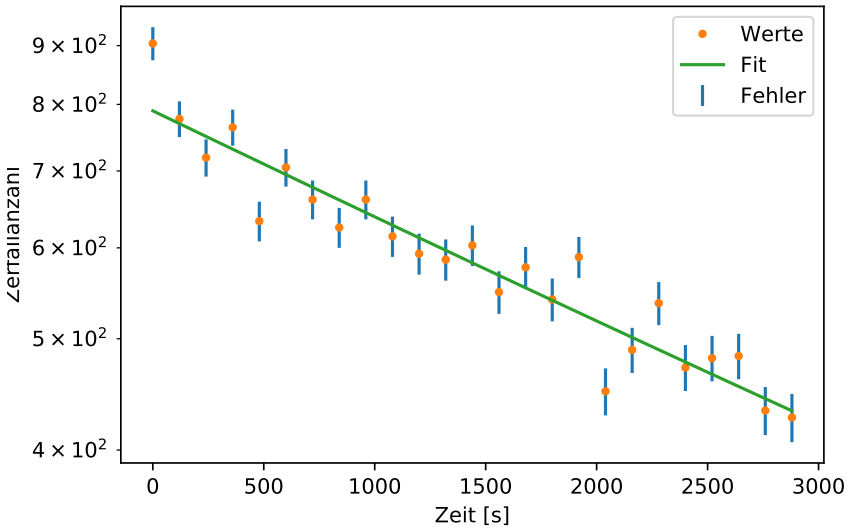


Figure 4: Indiummessung

Für den Fit ergaben sich die besten Parameter:

$$A_1 = 741 \pm 18 \tag{26}$$

$$\lambda_1 = (0,220 \pm 0,015)10^{-3}\frac{1}{s} \tag{27}$$

Dabei versuchten wir ebenfalls, die Proben so schnell wie möglich nach der Aktivierung in das Geigermüllerzählrohr einzusetzen. Genau wie bei Silber beachten wir den Fehler der Hintergrundstrahlung, indem wir die Bestfitparameter in zwei weiteren Plots voneinander abziehen.

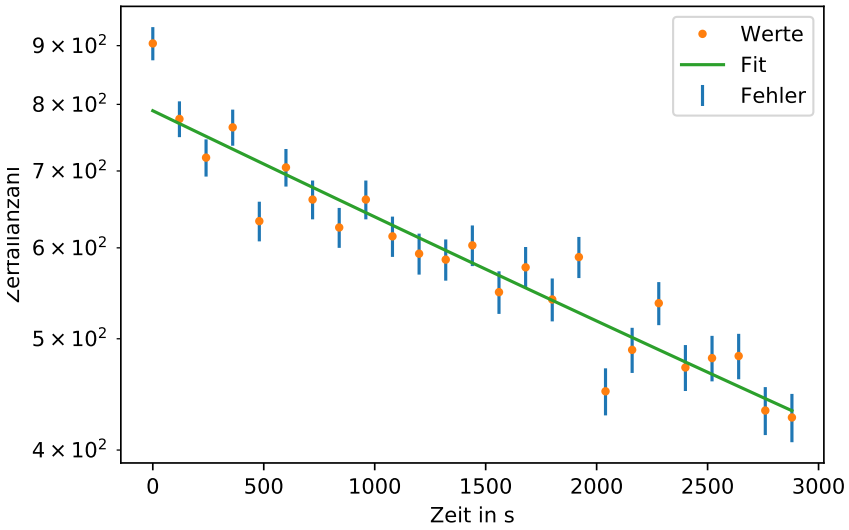


Figure 5: Indiummessung mit positivem Hintergrundstrahlungsfehler

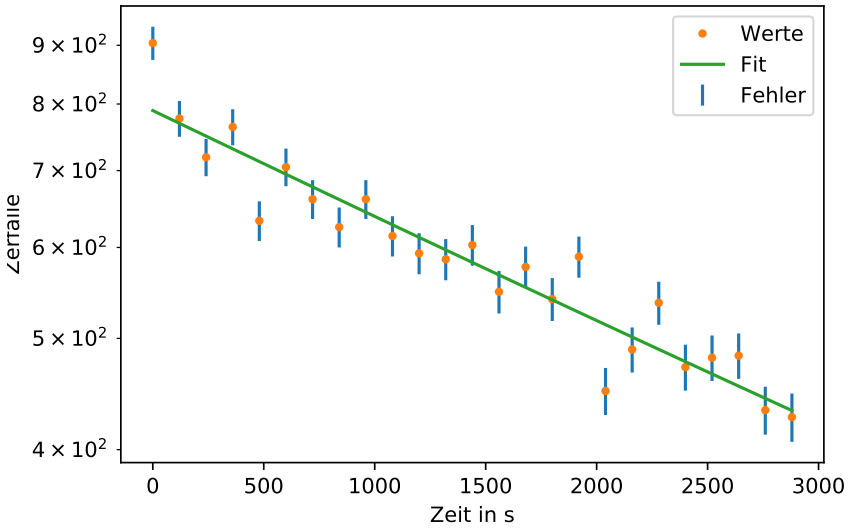


Figure 6: Indiummessung mit negativem Hintergrundstrahlungsfehler

$$|\lambda_1 - \lambda_1^-| = 1,4 \cdot 10^{-6} \frac{1}{s} \tag{28}$$

$$|\lambda_1 - \lambda_1^+| = 1,5 \cdot 10^{-6} \frac{1}{s} \tag{29}$$

quadratische Addition mit dem Fehler des Fits führt zu einem Gesamtfehler von:

$$\Delta\lambda_1 = 1,5 \cdot 10^{-5} \frac{1}{s} \tag{30}$$

somit ist die Halbwertszeit von Indium:

$$T_{1/2} = (3151 \pm 193)s \tag{31}$$

Der benutzte Fit (beispielhaft mit dem nach oben abgeschätzen Fehler) scheint die Daten nur annähernd zu beschreiben, da die χ^2_{red}-Summe mit 2,45 etwas entfernt von dem perfekten Wert 1 liegt und die χ^2-Summe 56,54 ist. Da die Werte sehr um den Fit streuen, ist die Fitwahrscheinlichkeit auch entsprechend $0,0\%$.

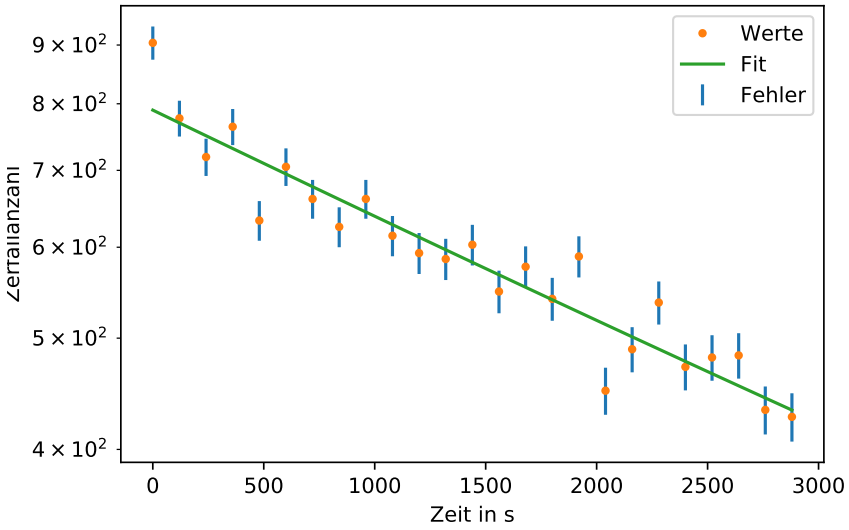


Figure 7: Indiummessung mit positivem Hintergrundstrahlungsfehler

3 Diskussion

In diesem Experiment haben wir die den radioaktiven Zerfall von instabilen Atomen ausgemessen. Wir haben mit einer Neutronenquelle instabile Isotope

kreiert und deren Zerfallskonstante bestimmt.
Die Halbwertszeiten der beiden Silberisotope sind:
für Ag^{110} ist die Halbwertszeit: $T^1_{1/2} = 24,755 \pm 0,01$
und für Ag^{108} ist die Halbwertszeit: $T^2_{1/2} = 130,785 \pm 0,012$
Im Vergleich zu den Literaturwerten
(Quelle: https://www.internetchemie.info/isotop.php?Kern=Ag-108)

$$T^{110}_{1/2} = 24,1s \tag{32}$$

$$T^{108}_{1/2} = 142,92s \tag{33}$$

unterscheiden sich die Werte stark.

Ein weiterer interessanter Parameter ist die Lebenszeit, die der Halbwertszeit durch ln 2 entspricht:

$$T^{110} = (35,71\pm)s \tag{34}$$

$$T^{108} = (188,682\pm)s \tag{35}$$

Der zweite Teil des Experiments war sehr ähnlich zum ersten Teil, bis auf die Messzeit von 120s und die Untersuchung von Indium statt Silber. Die Halbwertszeit von Indium wurde zu

$$T_{1/2} = (3151 \pm 193)s \tag{36}$$

bestimmt, woraus eine Lebenszeit von

$$(4545 \pm 287)s \tag{37}$$

resultiert. Im Vergleich mit dem Literaturwert von $3257,4s$
(Quelle: http://www.periodensystem-online.de/index.php?id=isotopeel=49mz=116nrg=0.1273show=nuklid)
weicht unser ermittelter wert um wenigerals 1σ vom Literaturwert ab. Wir beziehen uns dabei auf das Isotop ^{116}In.
Der kleinere Fehler kann durch die längere Messzeit erklärt werden. Würde man die Messungen häufiger wiederhohlen, wäre das Endergebnisse genauer.
Insgesamt konnten wir die Halbwertszeit von Indium und die Dimension der Halbwertszeit von Silber bestätigen. Vielleicht wäre eine Abschirmung von höherer Hintergrundstrahlung sinnvoll gewesen, das sich noch andere Experimente mit radioaktiver Strahlung im Raum befanden.

Versuch 252 Auswertung

November 24, 2021

[43]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.stats import chi2
```

1 Versuch 252 Auswertung

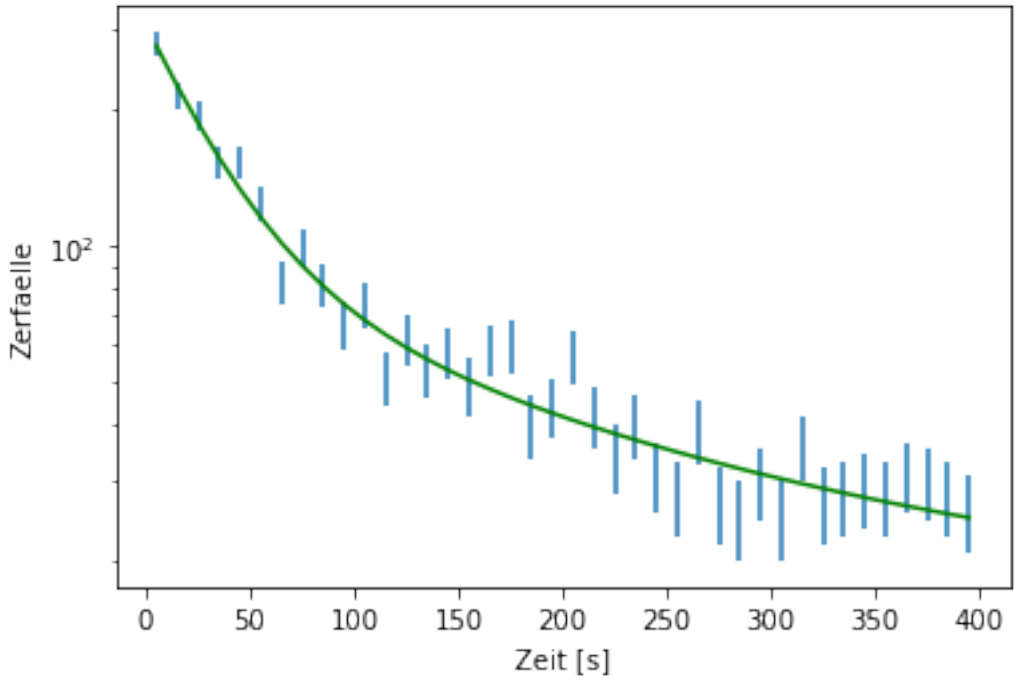
1.1 Aufgabe 1

[44]:
```
#untergrundstrahlung bestimmen
unterg = np.loadtxt('Untergrundmessung.dat', usecols=[1])
mittelw_unterg = np.mean(4*unterg)
fehler_unterg = np.std(4*unterg)/np.sqrt(len(unterg))
print('Mittelwert: ', mittelw_unterg, 'Fehler:', fehler_unterg)
```

```
Mittelwert:  16.083333333333332 Fehler: 1.1210708133380864
```

[45]:
```
n1 = np.loadtxt('AgMessung1.dat', usecols=[1])
n2 = np.loadtxt('AgMessung2.dat', usecols=[1])
n3 = np.loadtxt('AgMessung3.dat', usecols=[1])
n4 = np.loadtxt('AgMessung4.dat', usecols=[1])

N = n1[:-2]+n2[:-3]+n3+n4
dN = np.sqrt(N)
t = np.arange(5, 405, 10)

#einzelwerte plotten
plt.errorbar(t,N,dN,linestyle='none')
plt.xlabel('Zeit [s]')
plt.ylabel('Zerfaelle')
plt.yscale('log')

#fitten
y0=mittelw_unterg
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 = dN) ␣
↪#ändern

plt.plot(t,fit_func(t,*popt),color = 'green')
plt.savefig('Silbermessung_V252.pdf', format = 'pdf')
```



[ ]:

[46]:
```
#fitparameter
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]))

a = {'l1': popt[1], 'l2': popt[3]}
b = {'Fehler_l1': np.sqrt(pcov[1][1]), 'Fehler_l2': np.sqrt(pcov[3][3])}

#Güte des Fits
chi2_ = np.sum((fit_func(t, *popt)-N)**2/dN**2)
dof = len(N)-4
chi2_red = chi2_/dof
print('chi2 =', chi2_)
print('chi2_red =', chi2_red)
```

```
#Fitwahrscheinlichkeit
prob = round(1-chi2.cdf(chi2_, dof),2)*100
print('Fitwahrscheinlichkeit =', prob, '%')
```

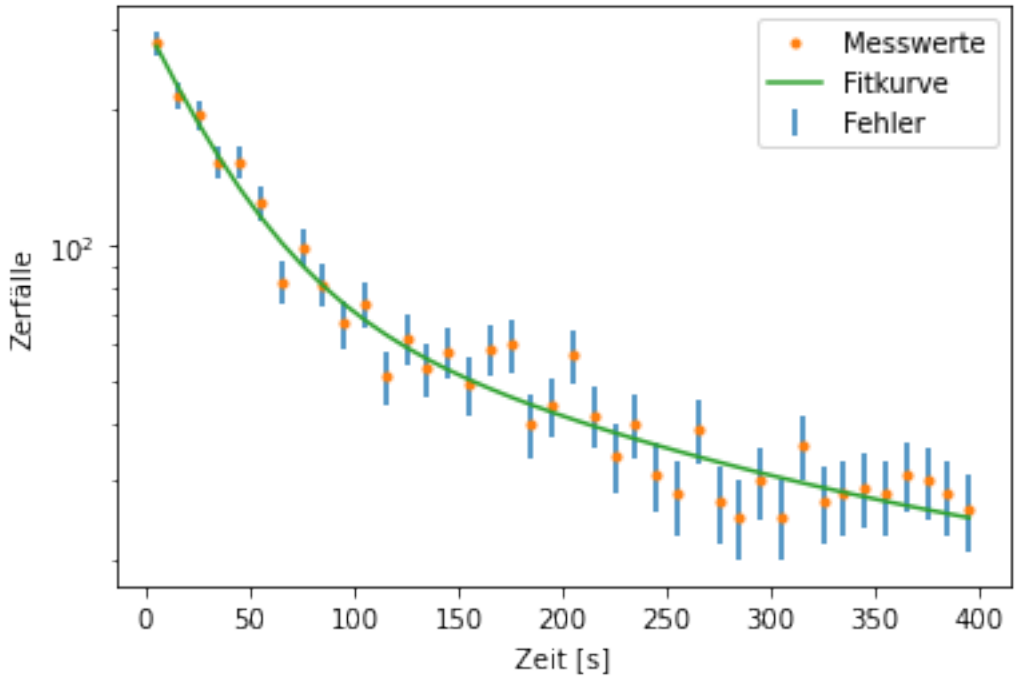
```
A1 = 220.0831637121549 , Standardfehler = 19.967668817375394
11 = 0.028668506727123306 , Standardfehler = 0.004916786582577869
A2 = 72.2617888141213 , Standardfehler = 19.2891001304612
l2 = 0.0053273127226763205 , Standardfehler = 0.001042156792106566
chi2 = 35.56760436781357
chi2_red = 0.9879890102170437
Fitwahrscheinlichkeit = 49.0 %
```

1.1.1 Wiederholung des Fits bei Beachtung des Fehlers

nach oben abgeschätzter Fehler

[47]:
```
unterg_korr_sub = mittelw_unterg - fehler_unterg
print('Wert für den nach unten korrigierten Untergrund:', unterg_korr_sub)

#Fit für die nach unten korrigierte Zerfallsfunktion
y0_sub = unterg_korr_sub

def fit_func_sub(x, A1, l1_sub, A2, l2_sub):
    return A1 * np.exp(-x*l1_sub) + A2*np.exp(-x*l2_sub) + y0_sub

popt, pcov = curve_fit(fit_func_sub, t, N, p0 = [500, 0.02, 50, 0.001], sigma =␣
↪dN)   #ändern

#plotten
plt.errorbar(t, N, dN, linestyle = 'None', label = 'Fehler')
plt.xlabel('Zeit [s]')
plt.ylabel('Zerfälle')
plt.yscale('log')
plt.plot(t, N, linestyle ='None', marker = '.', label = 'Messwerte')
plt.plot(t, fit_func_sub(t, *popt), label = 'Fitkurve')
plt.legend()
plt.savefig('Silbermessung_Abschätzung_unteren_V252.pdf', format = 'pdf')
```

```
Wert für den nach unten korrigierten Untergrund: 14.962262519995246
```



[48]:
```
#zugehörige Fitparameter
print('A1_sub =', popt[0], ', Standardfehler =', np.sqrt(pcov[0][0]))
print('l1_sub =', popt[1], ', Standardfehler =', np.sqrt(pcov[1][1]))
print('A2_sub =', popt[2], ', Standardfehler =', np.sqrt(pcov[2][2]))
print('l2_sub =', popt[3], ', Standardfehler =', np.sqrt(pcov[3][3]))

c = {'l1_sub': popt[1], 'l2_sub': popt[3]}
print(c)

#chi quarat summe und chi_reduziert werden bestimmt
chi2_ = np.sum((fit_func_sub(t, *popt)-N)**2/dN**2)
dof = len(N)-4
chi2_red = chi2_/dof
print('chi2= ', chi2_)
print('chi2_red= ', chi2_red)

#Bestimme die Fitwahrscheinlichkeit
prob = round(1-chi2.cdf(chi2_, dof),2)*100
print('Fitwahrscheinlichkeit =', prob, '%')
```

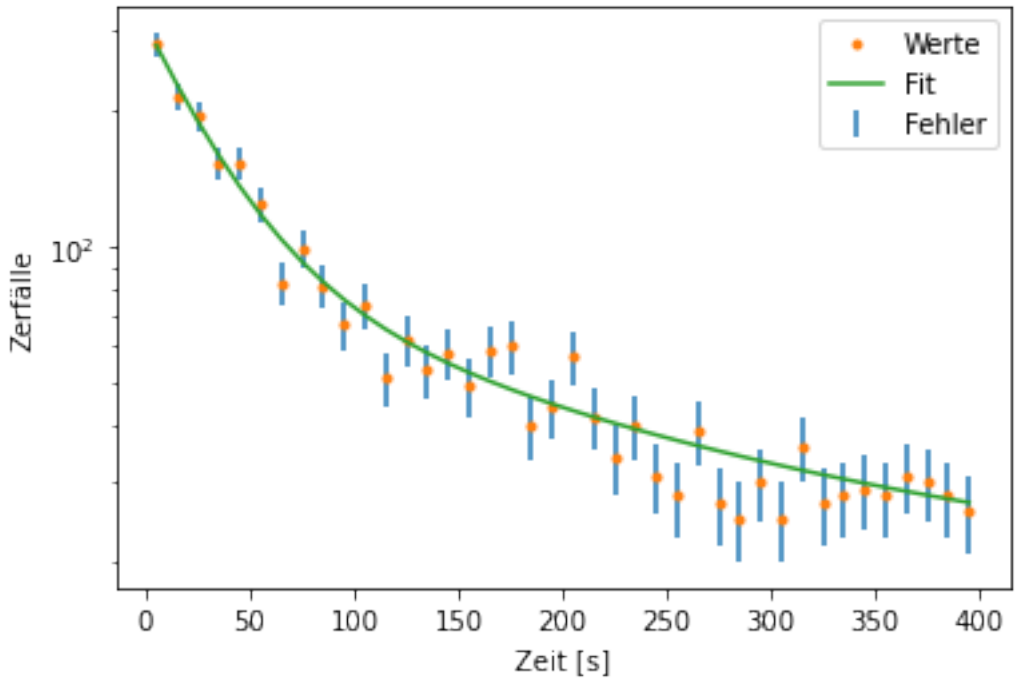
```
A1_sub = 222.56215062728563 , Standardfehler = 19.259444733997153
l1_sub = 0.028252156239278355 , Standardfehler = 0.004661844552937984
A2_sub = 70.41342890944405 , Standardfehler = 18.120624581715198
l2_sub = 0.004985603444046824 , Standardfehler = 0.00098914762267333
{'l1_sub': 0.028252156239278355, 'l2_sub': 0.004985603444046824}
```

```
chi2=  35.67712500878186
chi2_red=  0.9910312502439405
Fitwahrscheinlichkeit = 48.0 %
```

nach unten abgeschätzer Fehler

[49]:
```
unterg_korr_add = mittelw_unterg + fehler_unterg
print('Wert für den nach oben korrigierten Untergrund:', unterg_korr_add)

y0_add = unterg_korr_add

def fit_func_add(x, A1, l1_add, A2, l2_add):
    return A1 * np.exp(-x*l1_add) + A2*np.exp(-x*l2_add) + y0_add

plt.errorbar(t, N, dN, linestyle = 'None', label = 'Fehler')
plt.plot(t, N, marker = '.', linestyle = 'None', label ='Werte')
plt.xlabel('Zeit [s]')
plt.ylabel('Zerfälle')
plt.yscale('log')
plt.plot(t, fit_func_add(t, *popt), label ='Fit')
plt.legend()
plt.savefig('Silbermessung_Abschätzung_oberen_V252.pdf', format = 'pdf')
```

```
Wert für den nach oben korrigierten Untergrund: 17.20440414667142
```



[50]:
```
#zugehörige Fitparameter
print('A1_add =', popt[0], ', Standardfehler =', np.sqrt(pcov[0][0]))
print('l1_add =', popt[1], ', Standardfehler =', np.sqrt(pcov[1][1]))
print('A2_add =', popt[2], ', Standardfehler =', np.sqrt(pcov[2][2]))
print('l2_add =', popt[3], ', Standardfehler =', np.sqrt(pcov[3][3]))

d = {'l1_add': popt[1], 'l2_add': popt[3]}

#chi quadrat summe und chi_reduzidert bestimmen
chi2_ = np.sum((fit_func_add(t, *popt)-N)**2/dN**2)
dof = len(N)-4
chi2_red = chi2_/dof
print('chi2 =', chi2_)
print('chi2_red= ', chi2_red)

#Fitwahrscheinlichkeit
prob = round(1-chi2.cdf(chi2_, dof),2)*100
print('Fitwahrscheinlichkeit = ', prob, '%')
```

```
A1_add = 222.56215062728563 , Standardfehler = 19.259444733997153
l1_add = 0.028252156239278355 , Standardfehler = 0.004661844552937984
A2_add = 70.41342890944405 , Standardfehler = 18.120624581715198
l2_add = 0.004985603444046824 , Standardfehler = 0.00098914762267333
chi2 = 40.09238892460617
chi2_red=  1.113677470127949
Fitwahrscheinlichkeit =  28.999999999999996 %
```

Differenzen der Zerfallskonstanten

[51]:
```
print('|l1-l1_sub|=', abs(a['l1']-c['l1_sub']))
print('|l1-l1_add|= ', abs(a['l1']-d['l1_add']))

print('|l2-l12_sub|=', abs(a['l2']-c['l2_sub']))
print('|l2-l2_add|= ', abs(a['l2']-d['l2_add']))

print('l1_average= ', ((abs(a['l1']-c['l1_sub'])+abs(a['l1']-d['l1_add']))/2))
print('l2_average= ', ((abs(a['l2']-c['l2_sub'])+abs(a['l2']-d['l2_add'])))

#Fehler berechnen
print('Fehler_l1= ', np.
 ↪sqrt(((abs(a['l1']-c['l1_sub'])+abs(a['l1']-d['l1_add']))/2)**2 +␣
 ↪b['Fehler_l1']**2))

print('Fehler_l2= ', np.
 ↪sqrt(((abs(a['l2']-c['l2_sub'])+abs(a['l2']-c['l2_sub'])+abs(a['l2']-d['l2_add']))/
 ↪2)**2 + b['Fehler_l2']**2))
```

```
|l1-l1_sub|= 0.0004163504878449513
|l1-l1_add|=  0.0004163504878449513
```

```
|l2-l12_sub|= 0.00034170927862949615
|l2-l2_add|=  0.00034170927862949615
l1_average=  0.0004163504878449513
l2_average=  0.0006834185572589923
Fehler_l1=  0.004934383246905988
Fehler_l2=  0.0011613838940299638
```

[ ]:

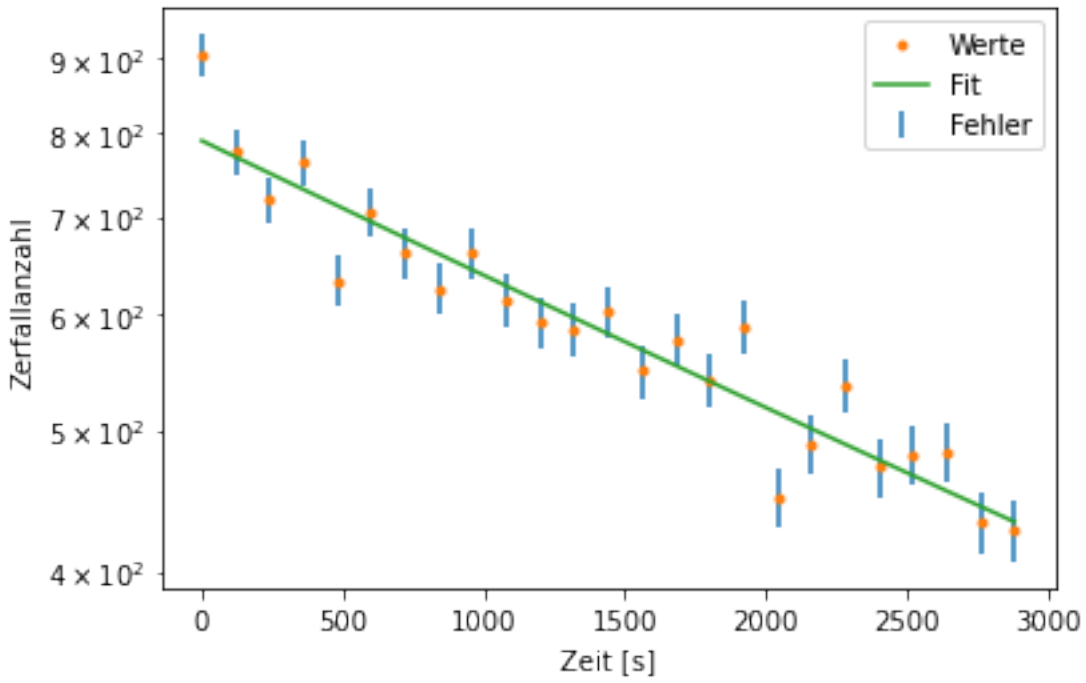
1.2 Aufgabe 2

[52]:
```
unterg = np.loadtxt('Untergrundmessung.dat', usecols = [1])
mittelw_unterg = np.mean(12 * unterg)
fehler_unterg = np.std(12*unterg)/np.sqrt(len(unterg))
print('Mittelwert:', mittelw_unterg, 'Fehler:', fehler_unterg)
```

```
Mittelwert: 48.25 Fehler: 3.363212440014259
```

[53]:
```
#Bestimme die Zerfallskonstante
n1 = np.loadtxt('InMessung.dat', usecols=[1])
N = n1
dN = np.sqrt(N)
print('N=', N)

#Arange Funktoin für die Zerfallszeiten jedes Intervalls
t = np.arange(0, 2900, 120)
print('t =', t)

#erstelle Plot
plt.errorbar(t, N, dN, linestyle = 'None', label = 'Fehler')
plt.plot(t, N, marker ='.', linestyle = 'None', label = 'Werte')
plt.xlabel('Zeit [s]')
plt.ylabel('Zerfallanzahl')
plt.yscale('log')

#Zerfallsfunktion fitten
y0 = mittelw_unterg
def fit_func(x, A1, l1):
    return(A1*np.exp(-x*l1))+y0
popt,pcov = curve_fit(fit_func, t, N[0:], p0=[700, 0.01], sigma = dN)
plt.plot(t, fit_func(t, *popt), label = 'Fit')
plt.legend()
plt.savefig('Indiummessung_V252.pdf', format = 'pdf')
```

```
N= [904. 777. 719. 764. 633. 705. 661. 625. 661. 614. 593. 586. 603. 549.
 577. 541. 589. 450. 489. 537. 472. 481. 483. 433. 427.]
t = [   0  120  240  360  480  600  720  840  960 1080 1200 1320 1440 1560
 1680 1800 1920 2040 2160 2280 2400 2520 2640 2760 2880]
```



[54]:
```
#Fitparameter
print('A1 =', popt[0], ', Standardfehler = ', np.sqrt(pcov[0][0]))
print('l1 =', popt[1], ', Standardfehler = ', np.sqrt(pcov[1][1]))

a = {'l1': popt[1]}
b = {'Fehler_l1': np.sqrt(pcov[1][1])}

#chi quadrat summer und chi reduziert bestimmen
chi2_ = np.sum((fit_func(t, *popt)-N)**2/dN**2)
dof = len(N)-2
chi2_red = chi2_/dof
print('chi2 = ', chi2_)
print('chi2_red = ', chi2_red)

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

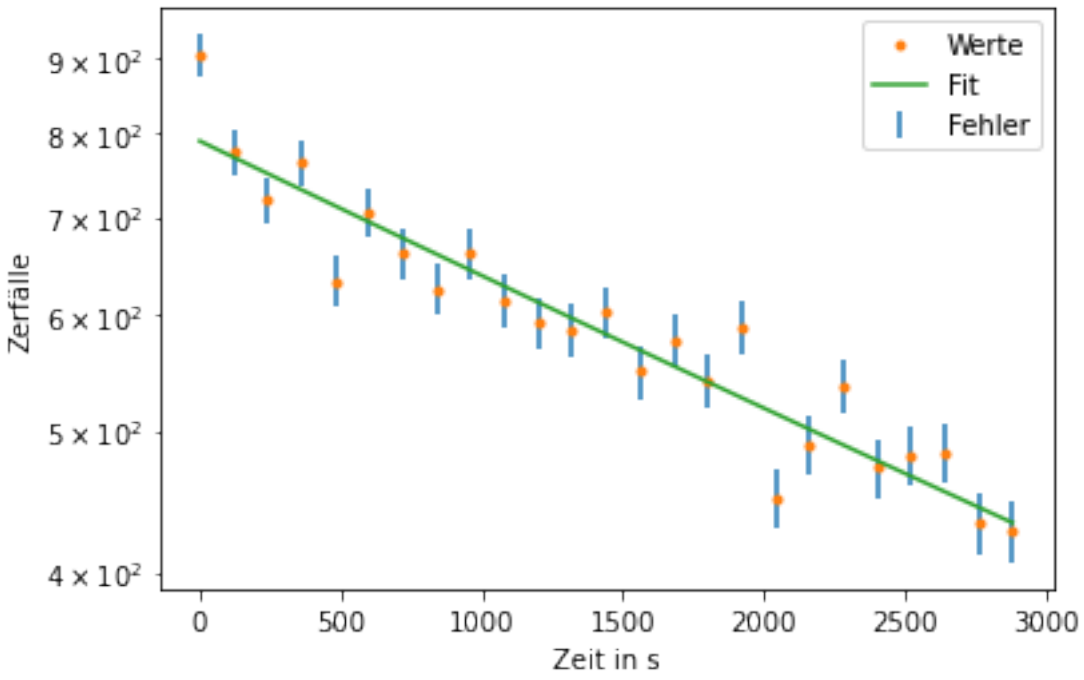
```
A1 = 741.469078422594 , Standardfehler =  18.608578617419727
l1 = 0.0002280203781967796 , Standardfehler =  1.6555902537545544e-05
chi2 =  56.58762458850176
chi2_red =  2.4603315038479026
Wahrscheinlichkeit =  0.0 %
```

1.2.1 Wiederholung des Fits bei Beachtung des Fehlers

nach oben abgeschätzter Fehler

[55]:
```
unterg_korr_sub = mittelw_unterg - fehler_unterg
print('Wert für den nach unten korrigierten Untergrund: ', unterg_korr_sub)

#Fit für den nach unten korrigierten Untergrund  korrigierter Untergrund?
y0_sub = unterg_korr_sub
def fit_func_sub(x, A1, l1_sub):
    return A1 * np.exp(-x * l1_sub) + y0_sub

popt, pcov = curve_fit(fit_func_sub, t, N[0:], p0 = [700, 0.01], sigma = dN)

#Plot
plt.errorbar(t, N, dN, linestyle = 'None', label = 'Fehler')
plt.plot(t, N, marker ='.', linestyle = 'None', label = 'Werte')
plt.xlabel('Zeit in s')
plt.ylabel('Zerfälle')
plt.yscale('log')
plt.plot(t, fit_func_sub(t, *popt), label = 'Fit')
plt.legend()
plt.savefig('Indiummessung_Abschätzung_unten_V252.pdf', format = 'pdf')
```

```
Wert für den nach unten korrigierten Untergrund:  44.88678755998574
```



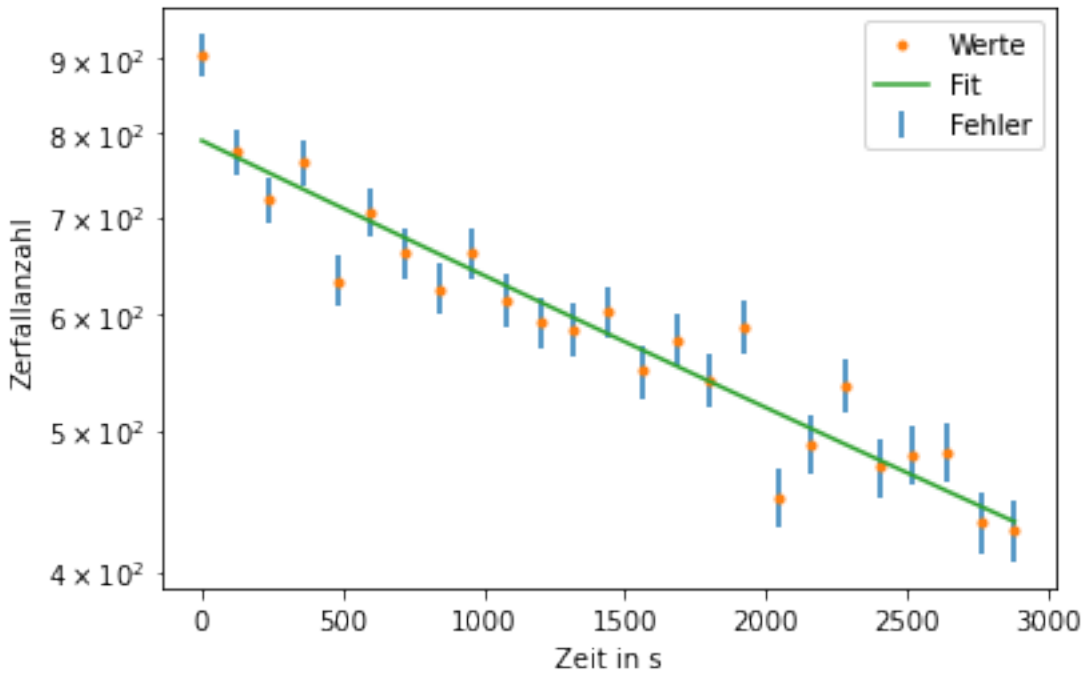
[56]:
```
#Fitparameter
print('A1_sub = ', popt[0], ', Standardfehler =', np.sqrt(pcov[0][0]))
print('l1_sub = ', popt[1], ', Standardfehler =', np.sqrt(pcov[1][1]))

c = {'l1_sub': popt[1]}
print(c)

#chi quadrat summe und chi_reduziert
chi2_ = np.sum((fit_func(t, *popt)-N)**2/dN**2)
dof = len(N)-2
chi2_red = chi2_/dof
print('chi2 = ', chi2_)
print('chi2_red = ', chi2_red)

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

```
A1_sub =  744.6811478521988 , Standardfehler = 18.59823691793363
l1_sub =  0.0002265768592430255 , Standardfehler = 1.645481427206612e-05
{'l1_sub': 0.0002265768592430255}
chi2 =  57.08032281386475
chi2_red =  2.4817531658202063
Wahrscheinlichkeit =  0.0 %
```

nach unten abgeschätzter Fehler

[57]:
```
#nach oben abgeschätzter Untergrund
unterg_korr_add = mittelw_unterg + fehler_unterg
print('Wert für den nach oben korrigierten Untergrund: ', unterg_korr_add)

y0 = unterg_korr_add
def fit_func_add(x, A1, l1):
    return A1*np.exp(-x*l1) + y0

popt, pcov = curve_fit(fit_func_add, t, N[0:], p0 = [700, 0.01], sigma = dN)

#erstelle Plot
plt.errorbar(t, N, dN, linestyle = 'None', label = 'Fehler')
plt.plot(t, N, marker = '.', linestyle = 'None', label = 'Werte')
plt.xlabel('Zeit in s')
plt.ylabel('Zerfallanzahl')
plt.yscale('log')
plt.plot(t, fit_func(t, *popt), label = 'Fit')
plt.legend()
plt.savefig('Indiummessung_Abschätzung_oberer_Untergrund.pdf', format = 'pdf')
```

```
Wert für den nach oben korrigierten Untergrund:  51.61321244001426
```



[58]:
```
#Gebe Fitparameter aus
print('A1_add = ', popt[0], ', Standardfehler = ', np.sqrt(pcov[0][0]))
print('l1_add = ', popt[1], ', Standardfehler = ', np.sqrt(pcov[1][1]))

d = {'l1_add': popt[1]}
print(d)

#bestimme chi quadrat summe und chi_reduziert
chi2_ = np.sum((fit_func(t, *popt)-N)**2/dN**2)
dof = len(N)-2
chi2_red = chi2_/dof
print('chi2 = ', chi2_)
print('chi2_red = ', chi2_red)

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

```
A1_add =  738.2588240019985 , Standardfehler =  18.61907423520785
l1_add =  0.00022948232259295954 , Standardfehler =  1.6658091821527702e-05
{'l1_add': 0.00022948232259295954}
chi2 =  56.547111792272275
chi2_red =  2.4585700779248816
Wahrscheinlichkeit =  0.0 %
```

Differenzen der Zerfallskonstanten

[59]:
```
#Differenz der Zerfallskonstanten
print('|l1- l1_sub| = ', abs(a['l1']-c['l1_sub']))
print('|l1- l1_add| = ', abs(a['l1']-d['l1_add']))
print('|l1_average| = ', (abs(a['l1']-c['l1_sub'])+abs(a['l1']-d['l1_add'])/2))

#Bestimme die Fehler
print('Der Fehler_l1 ist = ', np.
 ↪sqrt((abs(a['l1']-c['l1_sub'])+abs(a['l1']-d['l1_add'])/
 ↪2)**2+b['Fehler_l1']**2))
```

```
|l1- l1_sub| =  1.4435285766524518e-06
|l1- l1_add| =  1.4619347732815853e-06
|l1_average| =  2.1744959632932445e-06
Der Fehler_l1 ist =  1.6698093948923797e-05
```

[ ]:

[ ]:

[ ]: