Versuch 252 **30. November 2021**

Aktivierung mit thermischen Neutronen

Physikalisches Anfängerpraktikum II

Juan Provencio Betreuer/in: Henrik zu Jeddeloh

Inhaltsverzeichnis

1 Ziel des Versuches **2**

2 Grundlagen **2**
2.1 Aktivierung eines Isotops 2

3 Versuchsaufbau **3**
3.1 Materialen und Geräte 3
3.2 Aufbau 4

4 Messung und Auswertung **5**
4.1 Messprotokoll 5
4.2 Auswertung 6

5 Zusammenfassung und Diskussion **11**
5.1 Zusammenfassung 11
5.2 Diskussion 11

6 Quellen **12**

1 Ziel des Versuches

Durch thermische Aktivierung durch Kernreaktionen werden wir in diesem Versuch zwei stabile Isotope ^{115}In und $^{107/109}$Ag zu radioaktive Quellen aktiviert. Damit werden wir die Halbwertszeit der radioaktiven Isotopen bestimmen.

2 Grundlagen

2.1 Aktivierung eines Isotops

Aus einer Neutronenquelle kann ein Präparat (das Berylliumspäne und einen α-Strahler enthält) schnelle Neutronen erzeugen. Diese Neutronen stoßen gegen ein Paraffinblock was sie umgibt elastisch. Durch die Stöße gegen die Wasserstoffkernen werden sie abgebremst, bis sie nahezu thermische Energie besitzen. Wenn sie langsam genug geworden sind, können andere Atome diese abfangen und dann wird das Isotop zu einem Isotop mit einer um eins erhöhten Massenzahl. Wir bestrahlen ^{115}In zu ^{116}In, ein β-Strahler. Dabei wird pro Sekunde eine Anzahl an radioaktiven Kernen erzeugt, aber auch gleichzeitig werden einige Kerne zerfallen. Die Aktivität kann damit gemäß

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

berechnet werden. A_∞ ist die Gleichgewichtsaktivität, bei der pro Sekunde gleichviel Kerne neu gebildet werden wir pro Sekunde zerfallen. Nach der Aktivierung bleibt nur noch der exponentielle Zerfall:

$$A(t) = A_0\mathrm{e}^{-\lambda t} \tag{2}$$

und man hat dafür eine Halbwertszeit von

$$T_{1/2} = \frac{\ln 2}{\lambda} \tag{3}$$

und eine Lebensdauer von

$$\tau = \frac{1}{\lambda} \tag{4}$$

Silber besteht natürlicherweise aus 51% ^{107}Ag und 49% ^{109}Ag, das heißt, bei der Aktivierung entstehen zwei Isotope: $^{108/110}$Ag. Ihre Halbwertszeite unterscheiden sich allerdings sehr stark, um etwa einen Faktor 6. Das heißt,

man kann sie möglichst getrennt voneinander unterscheiden, indem man die Aktivierung unterschiedlich lang durchführt. Für eine kurze Aktivierung entstehen hauptsächlich ^{110}Ag Isotope, und für eine lange Aktivierung überwiegt ^{108}Ag.

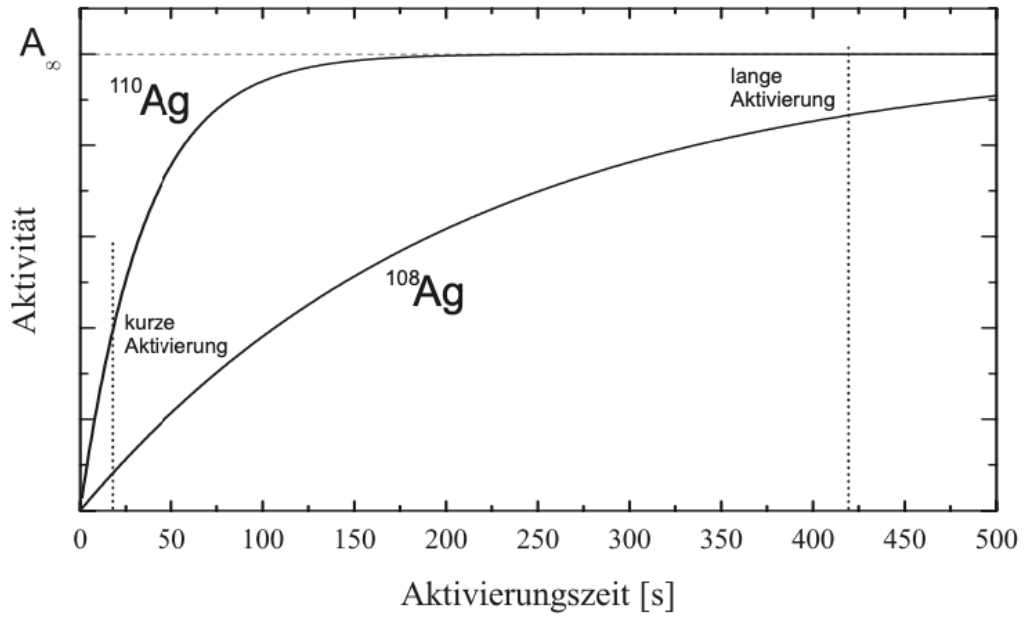


Abbildung 1: Aktivität von $^{108/110}$Ag in Abhängigkeit der Aktivierungszeit

3 Versuchsaufbau

3.1 Materialen und Geräte

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

3.2 Aufbau

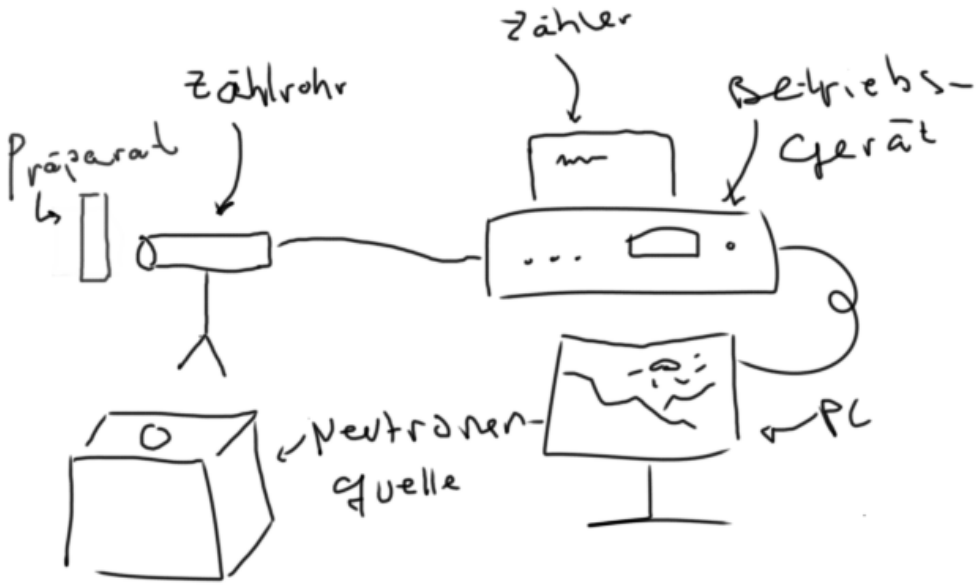


Abbildung 2: Aufbau

4 Messung und Auswertung

4.1 Messprotokoll

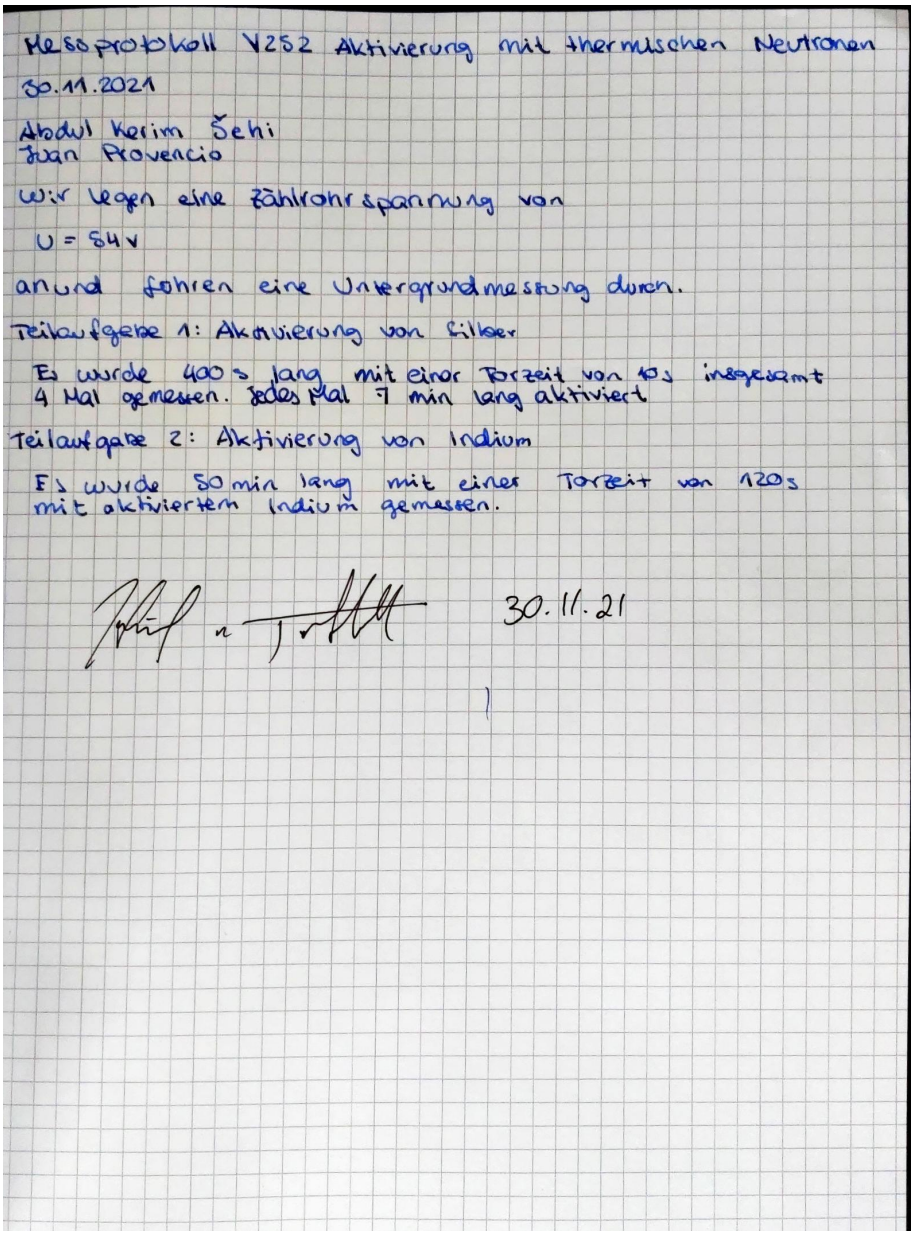

Messprotokoll V252 Aktivierung mit thermischen Neutronen
30.11.2021
Abdul Kerim Şehi
Juan Provencio
Wir legen eine Zählrohrspannung von
U = 54 V
an und führen eine Untergrundmessung durch.
Teilaufgabe 1: Aktivierung von Silber
Es wurde 400 s lang mit einer Totzeit von 10 s insgesamt 4 Mal gemessen. Jedes Mal 7 min lang aktiviert
Teilaufgabe 2: Aktivierung von Indium
Es wurde 50 min lang mit einer Totzeit von 120 s mit aktiviertem Indium gemessen.
30.11.21

Abbildung 3: Messprotokoll

Die Messdaten sind heruntergeladen und mit Python ausgearbeitet worden.

4.2 Auswertung

4.2.1 Zerfall der Silberisotope

Untergrundmessung

Als erstes wird eine Messung ohne radioaktive Quellen im Raum durchgeführt, um die Untergrundstrahlung zu bestimmen und diese dann in den Messreihen mitberücksichtigen können. Dafür wird mit einer Torzeit von 10 Sekunden 8 Minuten lang gemessen. Der Fehler daraus ergibt sich statistisch als der Fehler des Mittelwertes. Zusätzlich wird diese Messung mit 4 Multipliziert um zu berücksichtigen, dass man für Silber 4 Messreihen durchgeführt hat, die miteinander addiert werden. Der Mittelwert lautet:

$$N_{\mathrm{UG}} = 1,34(11)\,\mathrm{s}^{-1} \tag{5}$$

Bestimmung der Zeitkonstante

Die vier Messreihen werden miteinander addiert und gegen die Zeit auf einem Diagramm aufgetragen. Darüber haben wir mit Python eine Exponentialfunktion angepasst, diese ist von der Form

$$f(x) = A_1 \mathrm{e}^{-\lambda_1 x} + A_2 \mathrm{e}^{-\lambda_2 x} + y_0. \tag{6}$$

Die zwei von einander unabhängige Exponentialfunktionen beziehen sich jeweils auf einen der Isotope. In unserem Fall entspricht der Index 1 dem Isotop ^{110}Ag und der Index 2 entspricht ^{108}Ag. Der Term y_0 stellt den Untergrund dar.

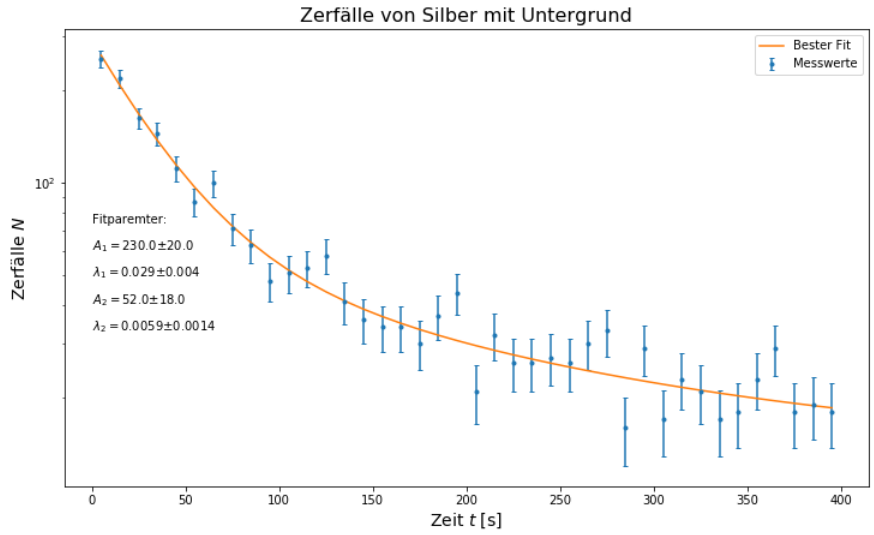


Diagramm 1: Zerfall von Silber und Untergrund in Abhängigkeit der Zeit

Um den Fehler der Fitparameter abzuschätzen, fitten wir die Daten unter Berücksichtigung des Fehlers der Untergrundmessung, dabei wird jeweils die Untergrundmessung $\pm$ den Fehler als Parameter y_0 benutzt.

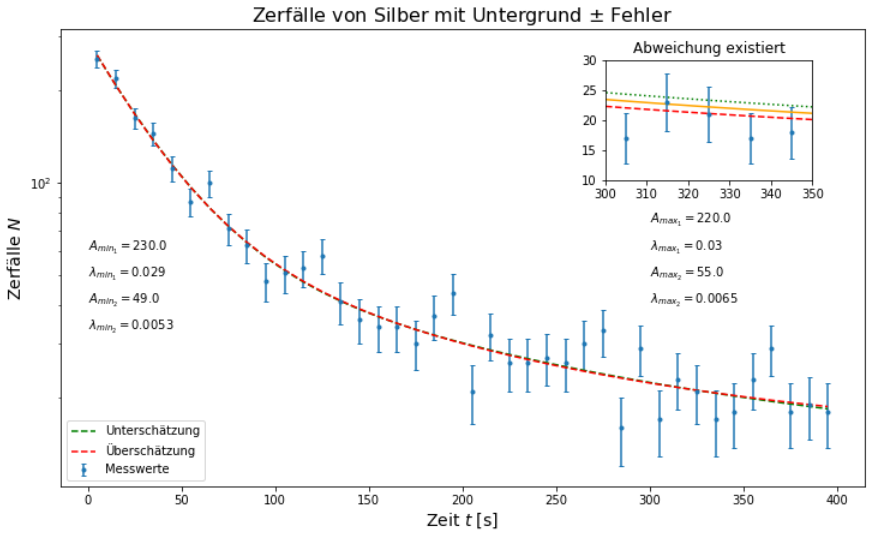


Diagramm 2: Zerfall von Silber unter Berücksichtigung des Fehlers des Untergrunds

Der Fehler der Zerfallskonstanten wird aus der quadratischen Addition des statistischen Fehlers, welcher der Wurzel aus dem dazugehörigen "pcov" Wert entspricht und der Summe der Differenzen aus $|\lambda - \lambda_{\min}|$ und $|\lambda - \lambda_{\max}|$. Mithilfe der Zerfallskonstanten lassen sich die Halbwertszeiten nach (3) und die Lebensdauer (4) der Isotopen feststellen.

Wir tragen tabellarisch nochmal die relevanten Werten auf:

Größe	N_{UG}	$N_{\mathrm{Ug}} + \Delta N_{\mathrm{Ug}}$	$N_{\mathrm{Ug}} - \Delta N_{\mathrm{Ug}}$	Gesamt
$\lambda_1[\mathrm{s}^{-1}]$	0,029(4)	0,030	0,029	0,0291(5)
$\lambda_2[\mathrm{s}^{-1}]$	0,0059(14)	0,0065	0,0053	0,0059(6)
$T_{1/2,1}[\mathrm{s}]$				23,8(4)
$T_{1/2,2}[\mathrm{s}]$				118(12)
$\tau_1[\mathrm{s}]$				34,3(6)
$\tau_2[\mathrm{s}]$				171(18)

Tabelle 1: Zusammenfassung der Ergebnisse für Silber

Um die Güte des Fits zu bestimmen wurde mit der χ^2 Methode die Fitwahrscheinleichkeit bestimmt. Daraus ergaben sich folgende Werte

	Wert
χ^2	35,07
χ^2_{red}	0,97
Wahrscheinlichkeit	51 %

Tabelle 2: Güte des Fits

4.2.2 Indium

Wir fahren analog zur Bestimmung der Zerfallskonstante von Silber fort, dieses Mal unter Berücksichtigung, dass nur ein Isotop relevant ist, und dass dieses Mal mit einer Torzeit von 120 s gemessen wurde, weshalb wir die Untergrundmessung um den Faktor 12 adjustieren:

$$N_{\text{UG}} = 0,335(27)\,\text{s}^{-1} \tag{7}$$

Bestimmung der Zeitkonstante

Die Werte werden auf einem Diagramm gegen die Zeit eingetragen und darüber wird eine einfache Exponentialfunktion mit dem Korrekturterm der Untergrundmessung angepasst. Hier wird allerdings der erste Messwert absichtlich weggelassen, denn ^{116}In in β^+ und β^- zerfällt, ersteres aber nur für eine ganz kurze Zeit und danach vernachlässigt sein kann. Auf diesem Grund liegt der erste Messwert deutlich über den Verlauf des gefundenen Fits.

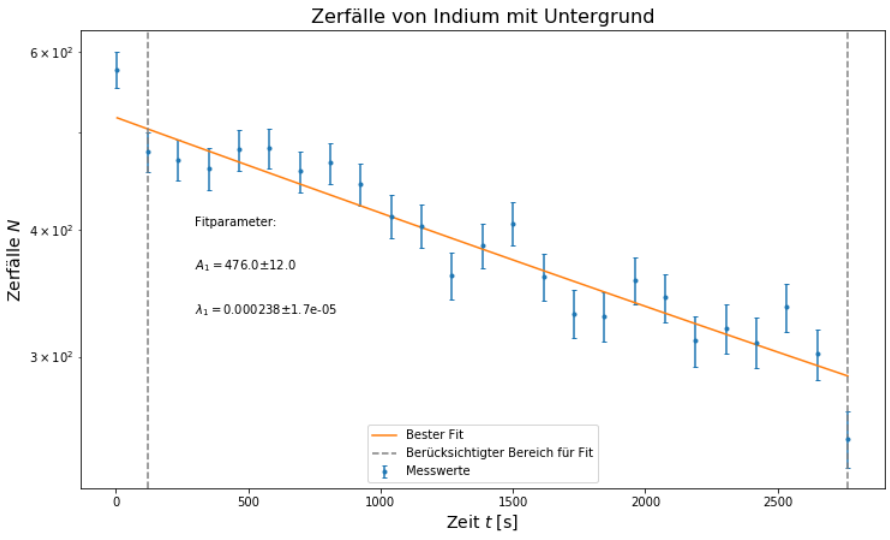


Diagramm 3: Zerfall von Indium mit Untergrund

Analog zur Untersuchung von Silber schätzen wir den Fehler der Zerfallskonstante indem wir einmal den Untergrund überschätzen und unterschätzen.

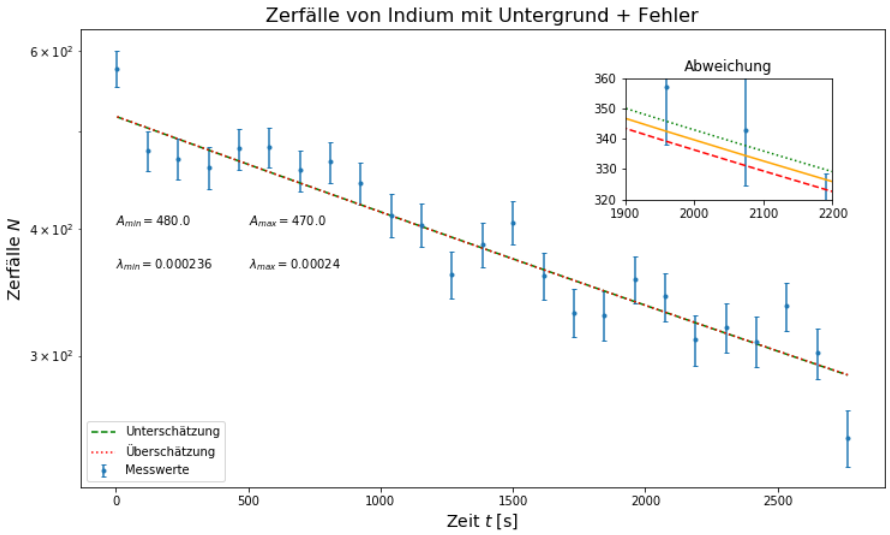


Diagramm 4: Zerfall von Indium mit Untergrund ± Fehler

Die relevanten Werten sind in folgender Tabelle aufgetragen.

Größe	N_{UG}	$N_{\mathrm{Ug}} + \Delta N_{\mathrm{Ug}}$	$N_{\mathrm{Ug}} - \Delta N_{\mathrm{Ug}}$	Gesamt
$\lambda[\mathrm{s}^{-1}]$	0,000238(17)	0,00024	0,000236	0,000238(2)
$T_{1/2}[\mathrm{s}]$				2915(28)
$\tau[\mathrm{s}]$				4205(40)

Tabelle 3: Zusammenfassung der Ergebnisse für Indium

Um die Güte des Fits zu bestimmen wird in der Diskussion auf die Ergebnisse der χ^2 Analyse zurückgegriffen:

	Wert
χ^2	34,8
χ^2_{red}	1,5
Wahrscheinlichkeit	5 %

Tabelle 4: Güte des Fits

5 Zusammenfassung und Diskussion

5.1 Zusammenfassung

In diesem Versuch wurde die Methode der Aktivierung durch thermische Neutronen zur Herstellung radioaktiver Isotopen verwendet, um aus Silber $^{107/109}$Ag und Indium ^{115}In ihre instabilen Isotopen $^{108/110}$Ag und ^{116}In zu erzeugen. Aus diesen radioaktiven Isotopen konnten wir anhand der Bestimmung der Zerfallskonstante die Halbwertszeit und die Lebensdauer bestimmen.

5.2 Diskussion

Wir wollen nun die in der Auswertung gefundenen Werten mit den Literaturwerten aus dem Praktikumsskript vergleichen. Dies machen wir tabellarisch und berechnen die σ-Abweichung gemäß

$$\frac{|G - G_{\text{Lit}}|}{\sqrt{(\Delta G)^2 + (\Delta G_{\text{Lit}})^2}}. \tag{8}$$

Größe	Messung	Literturwert	Relativer Fehler	$\frac{\text{Lit}}{\text{Messung}}$	Abweichung
$T_{1/2}^{^{108}\text{Ag}}$ [s]	118(12)	144.6	0,1	1,2	2,2
$T_{1/2}^{^{110}\text{Ag}}$ [s]	23,8(4)	24,6	0,017	1,03	0,2
$T_{1/2}^{^{116}\text{In}}$ [s]	2915(28)	3240	0,01	1,11	12

Tabelle 5: Zusammenfassung und Vergleich der Ergebnissen

Als erstes lässt sich bemerken, dass die Messungen für Silber unterschiedlich gut gelungen sind. Bei der σ-Abweichung unterscheiden sich diese um einen Faktor größer als 10. Es ist insbesondere bemerkenswert, dass der Isotop mit der kleineren Halbwertszeit (^{110}Ag) ein deutlich präziseres Ergebnis rauswirft, selbst mit einem relativen Fehler welcher um einen Faktor 6 kleiner ist als bei ^{108}Ag. Die Abweichung von $2,2$ von ^{108}Ag liegt immernoch innerhalb des akzeptablen Bereichs, und die von $0,2$ von ^{110}Ag ist sogar ziemlich gut. Allerdings ist der Literaturwert bei keinem dieser Werte innerhalb des abgeschätzten Fehlerbereiches. Daraus lässt sich vermuten, dass der Fehler zu klein abgeschätzt wurde. Dies kann man auch an den Plots beobachten,

denn die beiden Fehlerkurven sich kaum von einander unterscheiden, erst bei einer näheren Aufnahme kann man den Unterschied erkennen. Anhand der χ^2 Analyse lässt sich die Güte des Fits besprechen. Mit einer Fitwahrscheinlichkeit von 51% lässt sich vom Modell noch einiges zu erwarten, aber der lineare Zusammenhang ist immernoch gut.

Im Fall von ^{116}In lässt sich eine deutlich größere Abweichung erkennen, welche weit über den akzeptablen Bereich hinausgeht. Der Literaturwert ist 11% größer als der gemessene Wert und der relative Fehler beträgt nur 1%. Es lässt sich vermuten, dass einerseits der Fehler sehr gering abgeschätzt wurde, aber hier spielt zusätzlich ein anderer Faktor eine wichtige Rolle. Nämlich haben wir die Entscheidung getroffen, den ersten Messwert nicht im Fit zu berücksichtigen, eine kleine Anpassung des Analysebereiches um diesen hinzuzuzählen ergibt eine größere prozentuelle Abweichung vom Literaturwert, einen deutlich größeren Fehler und damit auch eine σ-Abweichung von nur noch 3,6. Wir haben uns trotzdem entschieden, den Wert aus der Anpassung rauszulassen, weil die argumentative Erklärung überzeugender ist als die rein numerische. Die χ^2 Analyse liefert eine Fitwahrscheinlichkeit von 5%, woran man erkennen kann, dass der Fit an Qualität mangelt.

Allgemein kann man für beide Elementen die statistische Natur des radioaktiven Zerfalls als Fehlerquelle. Allerdings ist diese Fehlerquelle nicht genug um die großen Abweichungen der Indium-Messung zu erklären. Wahrscheinlicht sind diese auf einen systematischen Fehler bei der Durchführung des Experiments zurückzuführen, oder einen Fehler in der Berechnung auf Python. Dieser Fehler ist allerdings nicht gefunden worden

6 Quellen

Wagner, J., Universität Heidelberg (2021). Physikalisches Praktikum PAP 2.2 für Studierende der Physik B.Sc..

Anhang

VI Auswertung mit Python

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from numpy import exp, sqrt, log
from scipy.optimize import curve_fit
from scipy.stats import chi2
```

1 Zerfall der Silberisotope

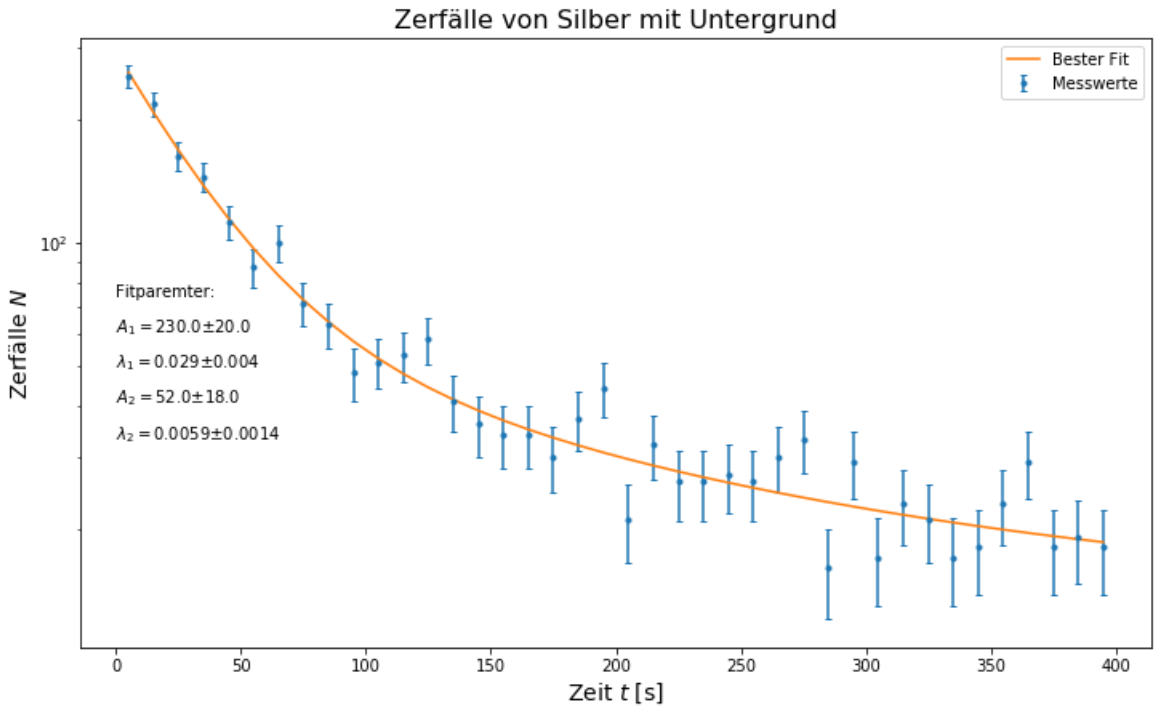
In [2]:

```
# Untergrundmessung
unterg = np.loadtxt('data/252/untergrundAJ.dat', usecols = [1])

mean_unterg = 4 * np.mean(unterg)
sig_mean_unterg = np.std(4 * unterg) / np.sqrt(len(unterg))

print('Mittelwert: ' ,np.round(mean_unterg,1)) # counts pro 10 Sekunden
print('Fehler des Mittelwerts: ', np.round(sig_mean_unterg,1))
```

```
Mittelwert:  13.4
Fehler des Mittelwerts:  1.1
```

In [3]:

```
# Bestimmung der Zerfallskonstante
n1 = np.loadtxt('data/252/SilberAJ1.dat', usecols = [1])
n2 = np.loadtxt('data/252/SilberAJ2.dat', usecols = [1])
n3 = np.loadtxt('data/252/SilberAJ3.dat', usecols = [1])
n4 = np.loadtxt('data/252/SilberAJ4.dat', usecols = [1])
N = n1 + n2 + n3 + n4
sig_N = np.sqrt(N)
t = np.arange(5, 405, 10)

# Fit
y0 = mean_unterg
def expo(x, A1, l1, A2, l2):
    return A1 * exp(-l1 * x) + A2 * exp(-l2 * x) + y0

popt, pcov = curve_fit(expo, t, N, sigma = sig_N, p0 = [250, 0.2, 50, 0.001])

# Fitparameter
A_1 = popt[0]
sig_A_1 = pcov[0,0]
l_1 = popt[1]
sig_l_1 = pcov[1,1]
A_2 = popt[2]
sig_A_2 = pcov[2,2]
l_2 = popt[3]
sig_l_2 = pcov[3,3]

# Plot
plt.figure(figsize = (12,7))
# Messungen
plt.errorbar(t, N, yerr = sig_N, fmt = '.', capsize = 2, label = 'Messwerte')
plt.xlabel('Zeit $t$ [s]', size = 14)
plt.ylabel('Zerfälle $N$', size = 14)
plt.title('Zerfälle von Silber mit Untergrund', size = 16)
plt.yscale('log')

# Fit
plt.plot(t, expo(t, *popt), label = 'Bester Fit')
plt.text(0, exp(4.3), 'Fitparemter:')
plt.text(0, exp(4.1), '$A_1 = $' + str(np.round(A_1,-1)) + '$\pm$'
         + str(np.round(sqrt(sig_A_1),-1)))
plt.text(0, exp(3.9), '$\\lambda_1 = $' + str(np.round(l_1, 3)) + '$\pm$'
         + str(np.round(sqrt(sig_l_1),3)))
plt.text(0, exp(3.7), '$A_2 = $' + str(np.round(A_2,0)) + '$\pm$'
         + str(np.round(sqrt(sig_A_2),0)))
plt.text(0, exp(3.5), '$\\lambda_2 = $' + str(np.round(l_2,4)) + '$\pm$'
         + str(np.round(sqrt(sig_l_2),4)))

plt.legend(loc = 'best')
plt.savefig('images/252/V252Diagramm1.png')
plt.show()
```



In [4]:

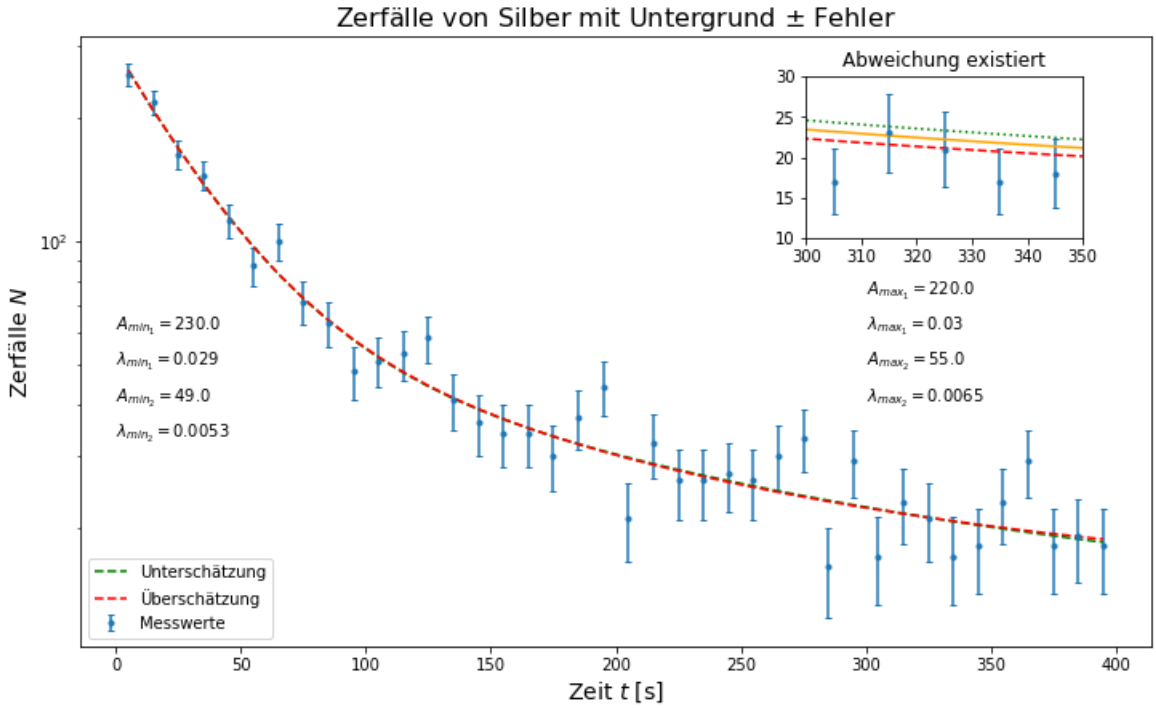
```
# Güte des Fits
chi2_= np.sum((expo(t, *popt) - N) ** 2 / sig_N ** 2)
dof = len(N) - 4 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi2_/dof

print("chi2 =", chi2_)
print("chi2_red =",chi2_red)

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

```
chi2 = 35.06843196724907
chi2_red = 0.974123110201363
Wahrscheinlichkeit = 51.0 %
```

In [5]:

```
# Fit unter Berücksichtigung des Untergrund-Fehlers

# Plot
plt.figure(figsize = (12,7))
# Messungen
plt.errorbar(t, N, yerr = sig_N, fmt = '.', capsize = 2, label = 'Messwerte')
plt.xlabel('Zeit $t$ [s]', size = 14)
plt.ylabel('Zerfälle $N$', size = 14)
plt.title('Zerfälle von Silber mit Untergrund $\pm$ Fehler', size = 16)
plt.yscale('log')

# Fit
y0 = mean_unterg - sig_mean_unterg # Unterschätzt
popt_min, pcov_min = curve_fit(expo, t, N, sigma = sig_N,
                               p0 = [250, 0.2, 50, 0.01])

# Fitparemeter
A_min_1 = popt_min[0]
sig_A_min_1 = pcov_min[0,0]
l_min_1 = popt_min[1]
sig_l_min_1 = pcov_min[1,1]
A_min_2 = popt_min[2]
sig_A_min_2 = pcov_min[2,2]
l_min_2 = popt_min[3]
sig_l_min_2 = pcov_min[3,3]

plt.plot(t, expo(t, *popt_min), label = 'Unterschätzung', ls = '--',
         color = 'green')
plt.text(0, exp(4.1), '$A_{min_1} = $' + str(np.round(A_min_1,-1)) )
plt.text(0, exp(3.9), '$\\lambda_{min_1} = $' + str(np.round(l_min_1, 3)) )
plt.text(0, exp(3.7), '$A_{min_2} = $' + str(np.round(A_min_2,0)) )
plt.text(0, exp(3.5), '$\\lambda_{min_2} = $' + str(np.round(l_min_2,4)) )


y0 = mean_unterg + sig_mean_unterg # Überschätzt
popt_max, pcov_max = curve_fit(expo, t, N, sigma = sig_N,
                               p0 = [250, 0.2, 50, 0.01])

# Fitparameter
A_max_1 = popt_max[0]
sig_A_max_1 = pcov_max[0,0]
l_max_1 = popt_max[1]
sig_l_max_1 = pcov_max[1,1]
A_max_2 = popt_max[2]
sig_A_max_2 = pcov_max[2,2]
l_max_2 = popt_max[3]
sig_l_max_2 = pcov_max[3,3]

plt.plot(t, expo(t, *popt_max), label = 'Überschätzung', ls = '--',
         color = 'red')
plt.text(300, exp(4.3), '$A_{max_1} = $' + str(np.round(A_max_1,-1)) )
plt.text(300, exp(4.1), '$\\lambda_{max_1} = $' + str(np.round(l_max_1, 3)) )
plt.text(300, exp(3.9), '$A_{max_2} = $' + str(np.round(A_max_2,0)) )
plt.text(300, exp(3.7), '$\\lambda_{max_2} = $' + str(np.round(l_max_2,4)) )
plt.legend(loc = 'lower left')
```

```
# Kleines Fenster oben rechts
a = plt.axes([.65, .63, .2, .2], facecolor = 'white')
plt.plot(t, expo(t, *popt_min), color = 'green', ls = 'dotted')
plt.plot(t, expo(t, *popt_max), color = 'red', ls = '--')
plt.plot(t, expo(t, *popt), color = 'orange')
plt.errorbar(t, N, yerr = sig_N, fmt = '.', capsize = 2)
plt.xlim(300, 350)
plt.ylim(10, 30)
plt.title('Abweichung existiert')

plt.savefig('images/252/V252Diagramm2.png')
plt.show()
```



In [6]:

```
# Differenz der Zerfallskonstanten:
sig_min_l_1 = np.abs(l_1 - l_min_1)
sig_max_l_1 = np.abs(l_1 - l_max_1)
sig_min_l_2 = np.abs(l_2 - l_min_2)
sig_max_l_2 = np.abs(l_2 - l_max_2)

f_l_1 = sqrt( ((sig_min_l_1 + sig_max_l_1) / 2) ** 2 + sig_l_1 ** 2 )
f_l_2 = sqrt( ((sig_min_l_2 + sig_max_l_2) / 2) ** 2 + sig_l_2 ** 2 )

print('Zerfallskonstante l_1 = ' + str(np.round(l_1,4))
      + ' +/- ' + str(np.round(f_l_1,4))) # s^-1
print('Zerfallskonstante l_2 = ' + str(np.round(l_2,4))
      + ' +/- ' + str(np.round(f_l_2,4))) # s^-1
print('Halbwertszeit T_12_1 = ' + str(np.round( log(2) / l_1, 1))
     + ' +/- ' + str(np.round(log(2) * f_l_1 / (l_1 ** 2),1)))
print('Halbwertszeit T_12_2 = ' + str(np.round( log(2) / l_2, 0))
     + ' +/- ' + str(np.round(log(2) * f_l_2 / (l_2 ** 2),0)))
print('Lebensdauer tau_1 = ' + str(np.round( 1 / l_1, 1))
     + ' +/- ' + str(np.round(1 * f_l_1 / (l_1 ** 2),1)))
print('Lebensdauer tau_2 = ' + str(np.round( 1 / l_2, 0))
     + ' +/- ' + str(np.round(1 * f_l_2 / (l_2 ** 2),0)))
```

```
Zerfallskonstante l_1 = 0.0291 +/- 0.0005
Zerfallskonstante l_2 = 0.0059 +/- 0.0006
Halbwertszeit T_12_1 = 23.8 +/- 0.4
Halbwertszeit T_12_2 = 118.0 +/- 12.0
Lebensdauer tau_1 = 34.3 +/- 0.6
Lebensdauer tau_2 = 171.0 +/- 18.0
```

2 Indiumzerfall

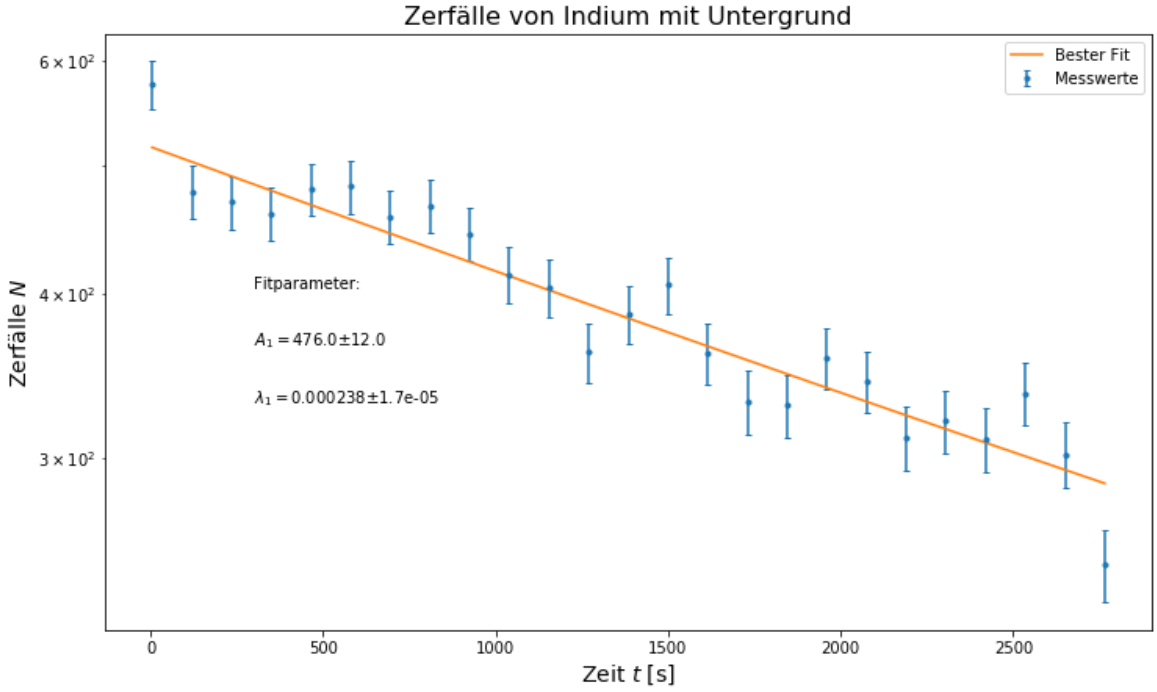
In [7]:

```
# Untergrundmessung
unterg_In = np.loadtxt('data/252/untergrundAJ.dat', usecols = [1])

mean_unterg_In = 12 * np.mean(unterg_In)
sig_mean_unterg_In = np.std(12 * unterg_In) / np.sqrt(len(unterg_In))

print('Mittelwert: ' ,np.round(mean_unterg_In,2))
print('Fehler des Mittelwerts: ', np.round(sig_mean_unterg_In,2))
```

```
Mittelwert:  40.25
Fehler des Mittelwerts:  3.29
```

In [8]:

```
# Bestimmung der Zerfallskonstante
N_In = np.loadtxt('data/252/IndiumAJ.dat', usecols = [1])
sig_N_In = np.sqrt(N_In)
t_In = np.arange(5, 2800, 115)



# Fit
y0 = mean_unterg_In
def expo(x, A1, l1):
    return A1 * exp(-l1 * x) + y0

popt, pcov = curve_fit(expo, t_In[1:], N_In[1:], sigma = sig_N_In[1:],
                       p0 = [600, 0.002])

# Fitparameter
A_In = popt[0]
sig_A_In = pcov[0,0]
l_In = popt[1]
sig_l_In = pcov[1,1]

# Plot
plt.figure(figsize = (12,7))
# Messungen
plt.errorbar(t_In, N_In, yerr = sig_N_In, fmt = '.', capsize = 2,
             label = 'Messwerte')
plt.xlabel('Zeit $t$ [s]', size = 14)
plt.ylabel('Zerfälle $N$', size = 14)
plt.title('Zerfälle von Indium mit Untergrund', size = 16)
plt.yscale('log')

# Fit
plt.plot(t_In, expo(t_In, *popt), label = 'Bester Fit')
plt.text(300, exp(6), 'Fitparameter:')
plt.text(300, exp(5.9), '$A_1 = $' + str(np.round(A_In,0)) + '$\pm$'
         + str(np.round(sqrt(sig_A_In),0)))
plt.text(300, exp(5.8), '$\\lambda_1 = $' + str(np.round(l_In, 6)) + '$\pm$'
         + str(np.round(sqrt(sig_l_In),6)))

plt.legend(loc = 'best')
plt.savefig('images/252/V252Diagramm3.png')
plt.show()
```


Zerfälle von Indium mit Untergrund
Bester Fit
Messwerte
Fitparameter:
$A_1 = 476.0 \pm 12.0$
$\lambda_1 = 0.000238 \pm 1.7e\text{-}05$
Zerfälle N
Zeit t [s]
6×10^2
4×10^2
3×10^2
0
500
1000
1500
2000
2500

In [9]:

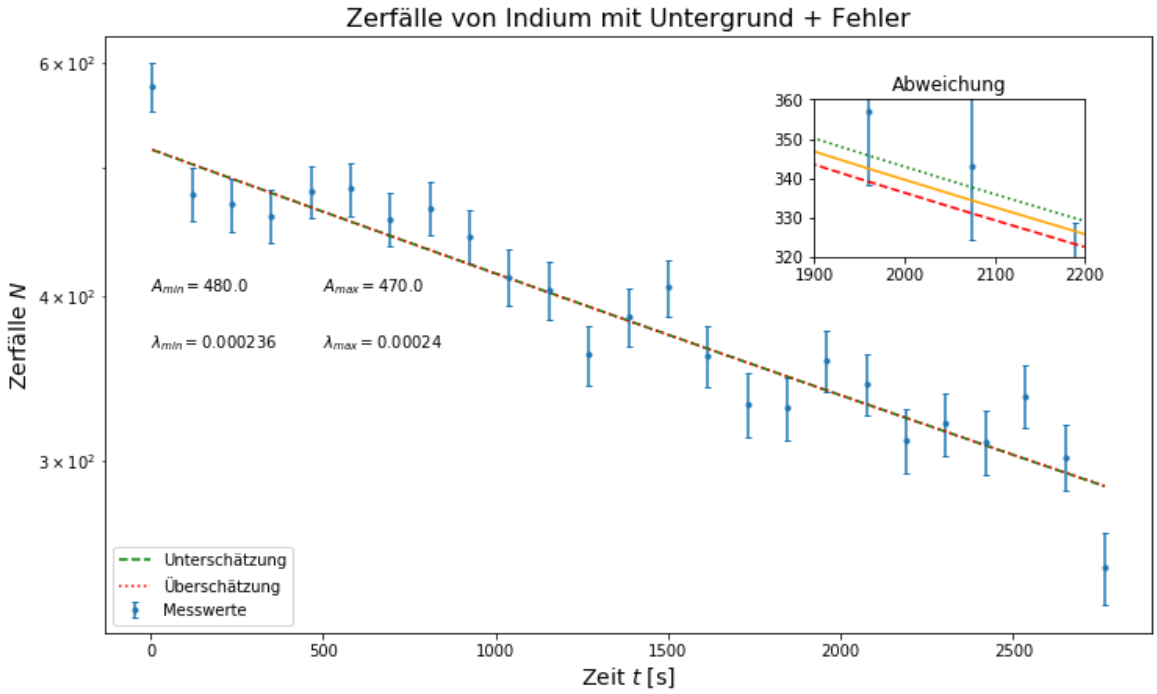
```
# Güte des Fits
chi2_= np.sum((expo(t_In, *popt) - N_In) ** 2 / sig_N_In ** 2)
dof = len(N_In) - 2 #dof:degrees of freedom, Freiheitsgrad
chi2_red = chi2_/dof

print("chi2 =", chi2_)
print("chi2_red =",chi2_red)

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

```
chi2 = 34.7570978870017
chi2_red = 1.5111781690000738
Wahrscheinlichkeit = 5.0 %
```

In [10]:

```
# Fit unter Berücksichtigung des Untergrund-Fehlers

# Fit unterschätzt
y0 = mean_unterg_In - sig_mean_unterg_In
popt_min, pcov_min = curve_fit(expo, t_In[1:], N_In[1:], sigma = sig_N_In[1:],
                               p0 = [600, 0.002])

# Fitparemeter
A_min_In = popt_min[0]
sig_A_min_In = pcov_min[0,0]
l_min_In = popt_min[1]
sig_l_min_In = pcov_min[1,1]

# Plot
plt.figure(figsize = (12,7))
# Messungen
plt.errorbar(t_In, N_In, yerr = sig_N_In, fmt = '.', capsize = 2,
             label = 'Messwerte')
plt.xlabel('Zeit $t$ [s]', size = 14)
plt.ylabel('Zerfälle $N$', size = 14)
plt.title('Zerfälle von Indium mit Untergrund + Fehler', size = 16)
plt.yscale('log')

# Plot min
plt.plot(t_In, expo(t_In, *popt_min), label = 'Unterschätzung', ls = '--',
         color = 'green')
plt.text(0, exp(6), '$A_{min} = $' + str(np.round(A_min_In,-1)) )
plt.text(0, exp(5.9), '$\\lambda_{min} = $' + str(np.round(l_min_In, 6)) )


# Fit überschätzt
y0 = mean_unterg_In + sig_mean_unterg_In
popt_max, pcov_max = curve_fit(expo, t_In[1:], N_In[1:], sigma = sig_N_In[1:],
                               p0 = [600, 0.002])

# Fitparameter
A_max_In = popt_max[0]
sig_A_max_In = pcov_max[0,0]
l_max_In = popt_max[1]
sig_l_max_In = pcov_max[1,1]

# Plot max
plt.plot(t_In, expo(t_In, *popt_max), label = 'Überschätzung', ls = 'dotted',
         color = 'red')
plt.text(500, exp(6), '$A_{max} = $' + str(np.round(A_max_In,-1)) )
plt.text(500, exp(5.9), '$\\lambda_{max} = $' + str(np.round(l_max_In, 6)) )
plt.legend(loc = 'lower left')

# Kleines Rechteck oben rechts
a = plt.axes([.65, .6, .2, .2], facecolor = 'white')
plt.plot(t_In, expo(t_In, *popt_min), color = 'green', ls = 'dotted')
plt.plot(t_In, expo(t_In, *popt_max), color = 'red', ls = '--')
plt.plot(t_In, expo(t_In, *popt), color = 'orange')
plt.errorbar(t_In, N_In, yerr = sig_N_In, fmt = '.', capsize = 2)
plt.xlim(1900, 2200)
```

```
plt.ylim(320, 360)
plt.title('Abweichung')


plt.savefig('images/252/V252Diagramm4.png')
plt.show()
```



In [11]:

```
# Differenz der Zerfallskonstanten:
sig_min_l_In = np.abs(l_In - l_min_In)
sig_max_l_In = np.abs(l_In - l_max_In)

f_l_In = sqrt( ((sig_min_l_In + sig_max_l_In) / 2) ** 2 + sig_l_In ** 2 )

print('Zerfallskonstante l_In = ' + str(np.round(l_In,6))
      + ' +/- ' + str(np.round(f_l_In,6))) # s^-1
print('Halbwertszeit T_12 = ' + str(np.round( log(2) / l_In, 0))
     + ' +/- ' + str(np.round(log(2) * f_l_In / (l_In ** 2),0)))
print('Lebensdauer tau = ' + str(np.round( 1 / l_In, 0))
     + ' +/- ' + str(np.round(1 * f_l_In / (l_In ** 2),0)))
```

```
Zerfallskonstante l_In = 0.000238 +/- 2e-06
Halbwertszeit T_12 = 2915.0 +/- 28.0
Lebensdauer tau = 4205.0 +/- 40.0
```

In [ ]: