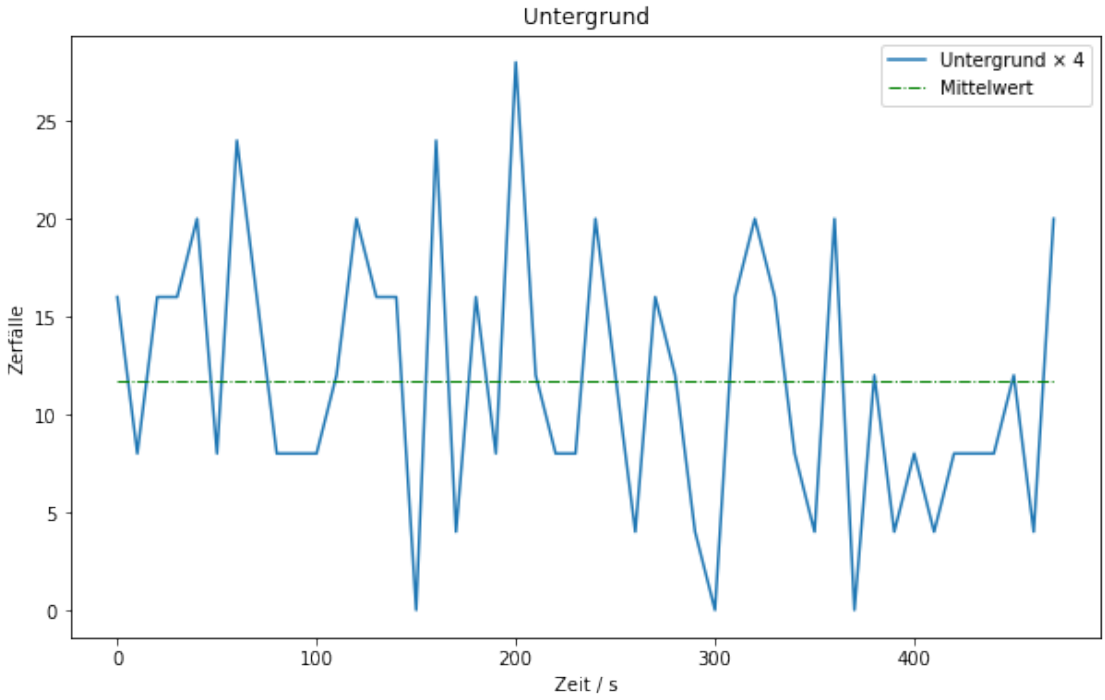
Versuch_252_Mabert

December 12, 2024

[4]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from scipy.stats import chi2
```

[7]:
```
unterg_data=np.loadtxt('untergrund.txt', usecols=(0,1), delimiter=',',␣
 ↪skiprows=4)
unterg_t = unterg_data[:,0]
unterg = unterg_data[:,1]

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

plt.figure(figsize=(10,6))
plt.plot(unterg_t, 4*unterg, label='Untergrund × 4')
plt.plot((unterg_t[0], unterg_t[-1]), (mittelw_unterg, mittelw_unterg),␣
 ↪linewidth=1, linestyle='-.', color='g', label='Mittelwert')
plt.xlabel('Zeit / s')
plt.ylabel('Zerfälle')
plt.title('Untergrund')
plt.legend()
plt.savefig('untergrund.png',format='png')
#plt.yscale('log')
```

```
Mittelwert: 11.666666666666666 Fehler: 0.9706336227586749
```

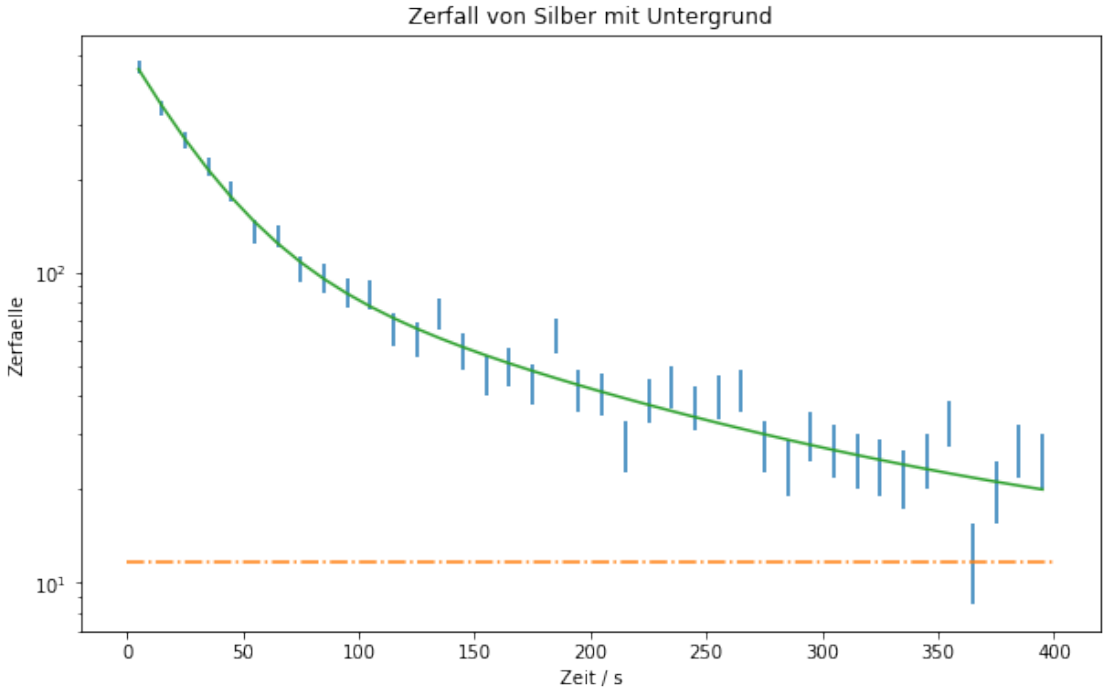


[8]:
```
#Silber
n1 =np.loadtxt('silber1.txt', usecols=[1], delimiter=',', skiprows=4)
n2 =np.loadtxt('silber2.txt', usecols=[1], delimiter=',', skiprows=4)
n3 =np.loadtxt('silber3.txt', usecols=[1], delimiter=',', skiprows=4)
n4 =np.loadtxt('silber4.txt', usecols=[1], delimiter=',', skiprows=4)
N=n1+n2+n3+n4
Fehler_N=np.sqrt(N)
t=np.arange(5,405,10)
```

[9]:
```
# Fit with background = mean

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)

plt.figure(figsize=(10,6))
plt.errorbar(t,N, Fehler_N, linestyle='None')
plt.plot([0,400], [y0, y0], linestyle='-.')
plt.xlabel('Zeit / s')
plt.ylabel('Zerfaelle')
plt.title('Zerfall von Silber mit Untergrund')
```

```
plt.yscale('log')
plt.plot(t,fit_func(t,*popt))
plt.savefig('silber.png',format='png')
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]))

l1 = popt[1]
l1_err = np.sqrt(pcov[1][1])
l2 = popt[3]
l2_err = np.sqrt(pcov[3][3])
```

```
A1= 394.04572684134564 , Standardfehler= 25.75251485363819
l1= 0.0358759181701219 , Standardfehler= 0.004170154725547418
A2= 113.935801849386 , Standardfehler= 19.436365369928318
l2= 0.006632896230054428 , Standardfehler= 0.0007463782296500707
```



[10]:
```
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)
```

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

```
chi2= 37.55902108627056
chi2_red= 1.0433061412852933
Wahrscheinlichkeit= 40.0 %
```

[11]:
```
# Fit with background = mean - 1sigma

y0 = mittelw_unterg - fehler_unterg #Untergrund

def fit_func(x, A1,l1,A2,l2):
    return A1*np.exp(-x*l1) + A2*np.exp(-x*l2) + y0

popt_subsig, pcov_subsig = curve_fit(fit_func,t,N, p0=[500,0.02,50,0.001,],␣
↪sigma=Fehler_N)

l1_sub = popt_subsig[1]
l1_sub_err = np.sqrt(pcov_subsig[1][1])
l2_sub = popt_subsig[3]
l2_sub_err = np.sqrt(pcov_subsig[3][3])
```

[12]:
```
# Fit with background = mean - 1sigma

y0 = mittelw_unterg + fehler_unterg #Untergrund

def fit_func(x, A1,l1,A2,l2):
    return A1*np.exp(-x*l1) + A2*np.exp(-x*l2) + y0

popt_addsig, pcov_addsig = curve_fit(fit_func,t,N, p0=[500,0.02,50,0.001,],␣
↪sigma=Fehler_N)

l1_add = popt_addsig[1]
l1_add_err = np.sqrt(pcov_addsig[1][1])
l2_add = popt_addsig[3]
l2_add_err = np.sqrt(pcov_addsig[3][3])
```

[13]:
```
l1_sub_diff = np.abs(l1 - l1_sub)
l1_add_diff = np.abs(l1 - l1_add)
l2_sub_diff = np.abs(l2 - l2_sub)
l2_add_diff = np.abs(l2 - l2_add)

print(f"|l1 - l1^-| = {l1_sub_diff}")
print(f"|l2 - l2^-| = {l2_sub_diff}")
print(f"|l1 - l1^+| = {l1_add_diff}")
print(f"|l2 - l2^+| = {l2_add_diff}")
```

```
l1_err_mean = np.mean([l1_sub_diff, l1_add_diff])
l2_err_mean = np.mean([l2_sub_diff, l2_add_diff])

print(f"l1 bkg err mean: {l1_err_mean}")
print(f"l2 bkg err mean: {l2_err_mean}")

l1_err_total = np.sqrt(pcov[1][1] + l1_err_mean**2)
l2_err_total = np.sqrt(pcov[3][3] + l2_err_mean**2)

print("A1=",popt[0], ", Standardfehler=", np.sqrt(pcov[0][0]))
print("l1=",popt[1], ", Standardfehler=", l1_err_total)
print("A2=",popt[2], ", Standardfehler=", np.sqrt(pcov[2][2]))
print("l2=",popt[3], ", Standardfehler=", l2_err_total)
```

```
|l1 - l1^-| = 0.00042124146933473355
|l2 - l2^-| = 0.0002915609365967088
|l1 - l1^+| = 0.00046744794798004446
|l2 - l2^+| = 0.0003150092715650272
l1 bkg err mean: 0.000444344708657389
l2 bkg err mean: 0.000303285104080868
A1= 394.04572684134564 , Standardfehler= 25.75251485363819
l1= 0.0358759181701219 , Standardfehler= 0.004193761158568438
A2= 113.935801849386 , Standardfehler= 19.436365369928318
l2= 0.006632896230054428 , Standardfehler= 0.0008056440380545968
```

[16]:
```
# Halbwertszeit berechnen

halbwertszeit_1 = np.log(2) / l1
halbwertszeit_2 = np.log(2) / l2

halbwertszeit_1_err = (np.log(2) / (lebensdauer_1**2)) * lebensdauer_1_err
halbwertszeit_2_err = (np.log(2) / (lebensdauer_2**2)) * lebensdauer_2_err

print(halbwertszeit_1, halbwertszeit_1_err, halbwertszeit_2,␣
↪halbwertszeit_2_err)
```

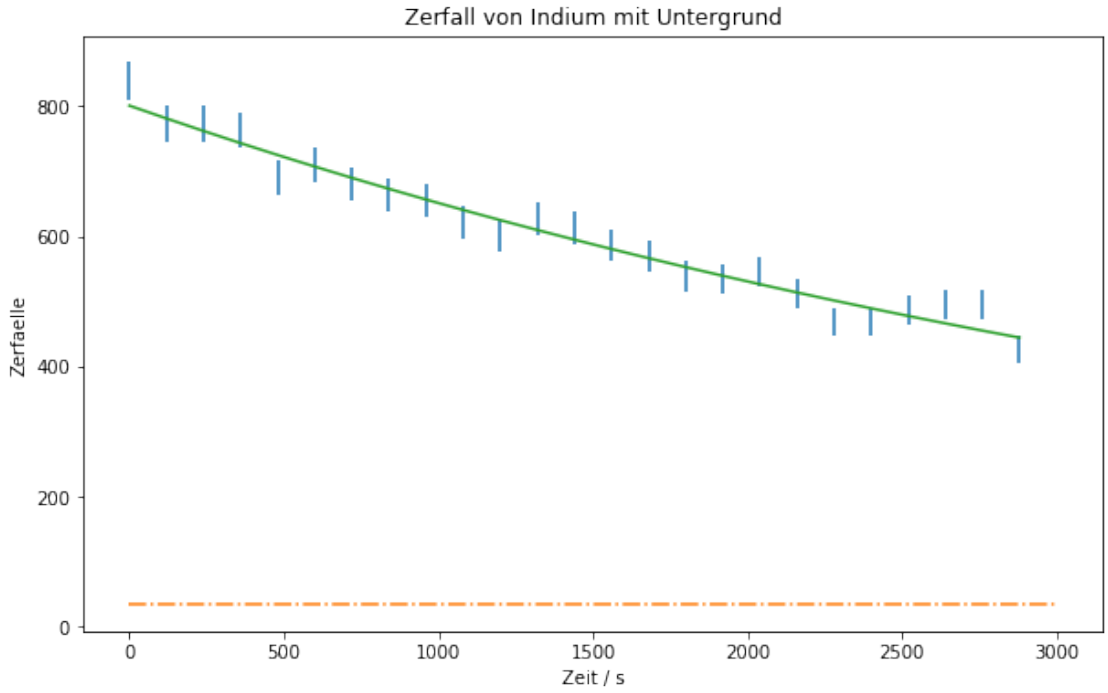
```
19.320681279098537 1.5654837562077193 104.50143595179651 8.798076093221706
```

0.0.1 Indiumzerfall

[43]:
```
#Indium
indium_data = np.loadtxt('indium.txt', usecols=[1], delimiter=',', skiprows=4)

mittelw_120_unterg=np.mean(unterg) * 12
fehler_120_unterg=np.std(unterg)/np.sqrt(len(unterg)) * 12
print('Mittelwert:', mittelw_120_unterg, 'Fehler:',fehler_120_unterg)
```

```
t_120 = t=np.arange(0, 3000, 120)

indium_error = np.sqrt(indium_data)

plt.figure(figsize=(10,6))
plt.errorbar(t_120, indium_data, indium_error, linestyle='None')
plt.plot([0,3000], [y0, y0], linestyle='-.')
plt.xlabel('Zeit / s')
plt.ylabel('Zerfaelle')
plt.title('Zerfall von Indium mit Untergrund')

#plt.yscale('log')

y0 = mittelw_120_unterg

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

popt_ind, pcov_ind=curve_fit(fit_func,t_120,indium_data, p0=[50,0.001],␣
 ↪sigma=indium_error)

plt.plot(t_120,fit_func(t_120,*popt_ind))

plt.savefig('indium.png',format='png')

l1_ind = popt_ind[1]

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

```
Mittelwert: 35.0 Fehler: 2.9119008682760246
A= 765.6323496164731 , Standardfehler= 10.412435073562266
l= 0.00021744099371320672 , Standardfehler= 8.899017215255906e-06
```



[18]:
```
chi2_=np.sum((fit_func(t_120,*popt_ind)-indium_data)**2/indium_error**2)
dof=len(indium_data)-4 #dof:degrees of freedom, Freiheitsgrad
chi2_red=chi2_/dof
print("chi2=", chi2_)
print("chi2_red=",chi2_red)

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

```
chi2= 17.722120844041417
chi2_red= 0.8439105163829246
Wahrscheinlichkeit= 67.0 %
```

[19]:
```
# Fit with background = mean - 1sigma

y0 = mittelw_120_unterg - fehler_120_unterg #Untergrund

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

popt_ind_subsig, pcov_ind_subsig = curve_fit(fit_func, t_120, indium_data,␣
↪p0=[50,0.001], sigma=np.sqrt(indium_data))

l1_ind_sub = popt_ind_subsig[1]
```

```
l1_ind_sub_err = np.sqrt(pcov_ind_subsig[1][1])
```

[20]:
```
# Fit with background = mean + 1sigma

y0 = mittelw_120_unterg + fehler_120_unterg #Untergrund

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

popt_ind_addsig, pcov_ind_addsig = curve_fit(fit_func, t_120, indium_data,␣
↪p0=[50,0.001], sigma=np.sqrt(indium_data))

l1_ind_add = popt_ind_addsig[1]
l1_ind_add_err = np.sqrt(pcov_ind_addsig[1][1])
```

[44]:
```
l1_ind_sub_diff = np.abs(l1_ind - l1_ind_sub)
l1_ind_add_diff = np.abs(l1_ind - l1_ind_add)

print(f"|l1 - l1^-| = {l1_ind_sub_diff}")
print(f"|l1 - l1^+| = {l1_ind_add_diff}")

l1_ind_err_mean = np.mean([l1_ind_sub_diff, l1_ind_add_diff])

print(f"l1 bkg err mean: {l1_ind_err_mean}")

l1_ind_err_total = np.sqrt(pcov_ind[1][1] + l1_ind_err_mean**2)

print("A1=",popt_ind[0], ", Standardfehler=", np.sqrt(pcov_ind[0][0]))
print("l1=",popt_ind[1], ", Standardfehler=", l1_ind_err_total)
```

```
|l1 - l1^-| = 1.1299513612909298e-06
|l1 - l1^+| = 1.1417382263016165e-06
l1 bkg err mean: 1.1358447937962732e-06
A1= 765.6323496164731 , Standardfehler= 10.412435073562266
l1= 0.00021744099371320672 , Standardfehler= 8.97121233685922e-06
```

[22]:
```
halbwertszeit_ind = np.log(2) / l1_ind

halbwertszeit_ind_err = (np.log(2) / (l1_ind**2)) * l1_ind_err_total

print(halbwertszeit_ind / 60, halbwertszeit_ind_err/60)
```

```
53.12914005796672 2.1920098349228767
```

[ ]:
```

```