Versuch_251-RG_MP

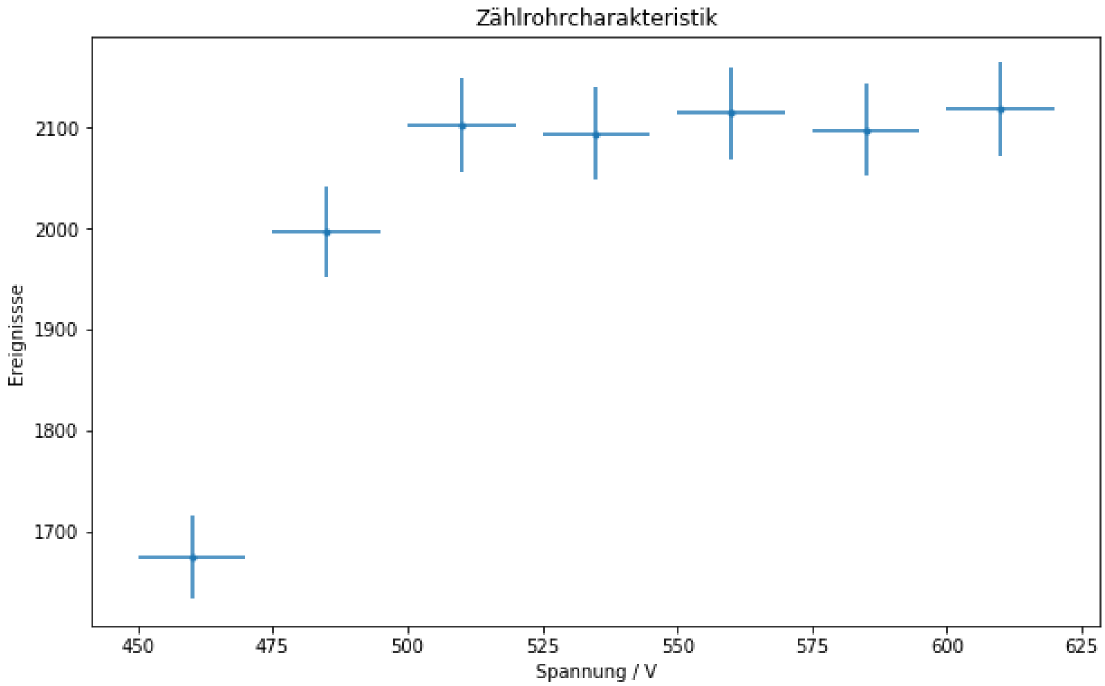
December 12, 2024

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

from scipy.optimize import curve_fit
from scipy.special import gamma
from scipy.stats import chi2
```

[2]:
```
# Aufgabe 2

#Zaehlrohrspannung:
U=np.arange(460, 615, 25)
print(U)
dU=10#Ablesefehler der Spannung

#gezaehlte Ereignisse
N=np.array([1674, 1996, 2102, 2094, 2114, 2097, 2118])
assert(len(U) == len(N))
Fehler_N=np.sqrt(N)
print(Fehler_N)
```

```
[460 485 510 535 560 585 610]
[40.91454509 44.67661581 45.84757355 45.76024475 45.97825573 45.79301257
 46.021734  ]
```

[3]:
```
plt.figure(figsize=(10,6))
plt.errorbar(U, N, yerr=Fehler_N,xerr=dU, fmt=".")
plt.xlabel('Spannung / V')
plt.ylabel('Ereignissse')
plt.title('Zählrohrcharakteristik')
plt.savefig("bestimmungU0.png", format="png")
```

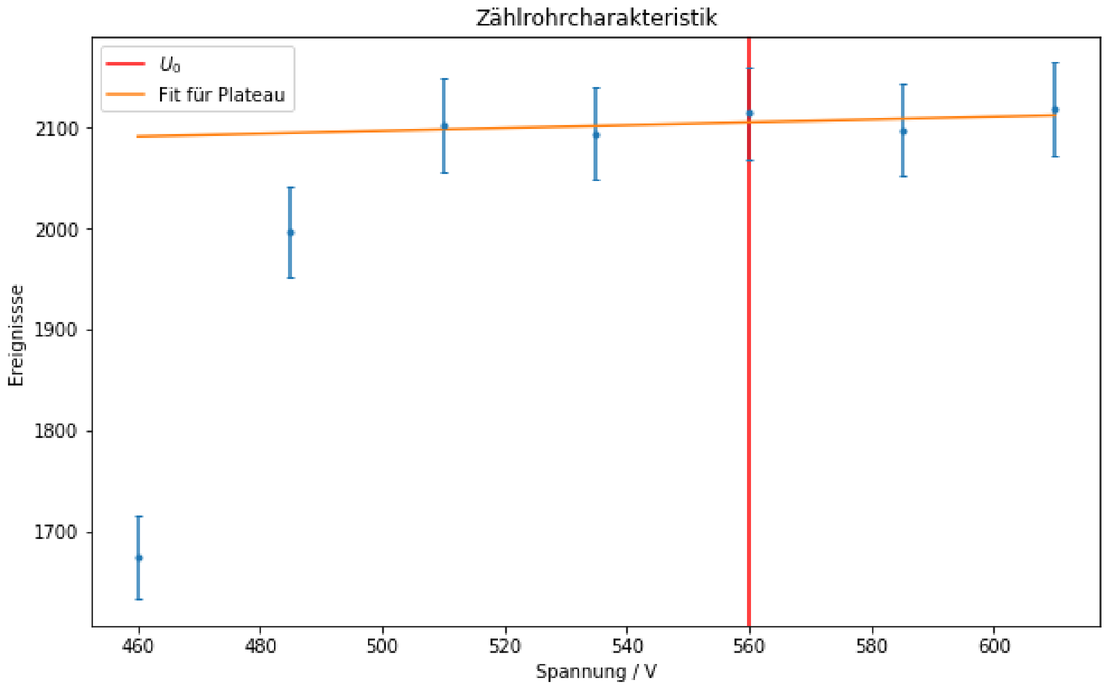


[4]:
```
def linear(x,a,b):
    return a*x+b

popt, pcov = curve_fit(linear, U[2:], N[2:])
```

[5]:
```
U0 = np.mean(U[[2,-1]])
U0
```

[5]: 560.0

[6]:
```
plt.figure(figsize=(10,6))
plt.axvline(x=U0, color="red", label="$U_0$")
plt.errorbar(U, N, Fehler_N, fmt=".",capsize=2)
plt.xlabel('Spannung / V')
plt.ylabel('Ereignissse')
plt.title('Zählrohrcharakteristik')
plt.plot(U[0:], linear(U[0:],*popt), label="Fit für Plateau")
plt.legend()
plt.savefig("bestimmungU0_fit_mean.png", format="png")
```



[7]:
```
# Aufgabe 3
# U_0 = 560
N_1 = np.array([10566, 10730]) # U0 , U0 +100, t=1min
N_3 = np.array([31199, 31756]) # U0 , U0 +100, t=3min
err_N_1 = np.sqrt(N_1)
err_N_3 = np.sqrt(N_3)
A_1 = N_1[1]-N_1[0]
err_A_1 = np.sqrt(N_1[1]+N_1[0])
A_3 = N_3[1]-N_3[0]
err_A_3 = np.sqrt(N_3[1]+N_3[0])

print(err_N_1, err_N_3)

print("1min: slope =", A_1 ,"+-", np.round(err_A_1))
print("3min: slope =", A_3 ,"+-", np.round(err_A_3))
print("sigma 1min: ", np.round(A_1/err_A_1,2))
print("sigma 3min: ", np.round(A_3/err_A_3,2))
```

```
[102.79105019 103.5857133 ] [176.63238661 178.20213242]
1min: slope = 164 +- 146.0
3min: slope = 557 +- 251.0
sigma 1min:  1.12
sigma 3min:  2.22
```

[8]:
```
print(f"Messung 1 Minute:\n3sigma-Bereich: [{-3 * err_A_1}, {3 * err_A_1}],␣
↪{A_1} entspricht {A_1 / err_A_1} sigma => Nicht signifikant")
print(f"Messung 3 Minuten:\n3sigma-Bereich: [{-3 * err_A_3}, {3 * err_A_3}],␣
↪{A_3} entspricht {A_3 / err_A_3} sigma => Nicht signifikant")
```

```
Messung 1 Minute:
3sigma-Bereich: [-437.7944723269128, 437.7944723269128], 164 entspricht
1.123815011608028 sigma => Nicht signifikant
Messung 3 Minuten:
3sigma-Bereich: [-752.7250494038311, 752.7250494038311], 557 entspricht
2.2199340932302647 sigma => Nicht signifikant
```

[9]:
```
ratio_1 = (N_1[1]-N_1[0])/N_1[0]
err_ratio_1 = ratio_1 * np.sqrt( (err_N_1[0]/N_1[0])**2 + (err_A_1/A_1)**2)
ratio_3 = (N_3[1]-N_3[0])/N_3[0]
err_ratio_3 = ratio_3 * np.sqrt( (err_N_3[0]/N_3[0])**2 + (err_A_3/A_3)**2)
print("1min Anstieg =", np.round(ratio_1*100,2), "+-", np.
↪round(err_ratio_1*100, 2),"%")
print("3min Anstieg =", np.round(ratio_3*100,2), "+-", np.
↪round(err_ratio_3*100, 2),"%")
```

```
1min Anstieg = 1.55 +- 1.38 %
3min Anstieg = 1.79 +- 0.8 %
```

[10]:
```
# Aufgabe (b)

# t_{1\%} &= (\frac{\sigma_{\Delta}}{\Delta} \frac{\sqrt{t_{mess}}}{1 \%})^2

t1p1 = (((err_A_1 / A_1)*(np.sqrt(1) / 0.01))**2)
t1p3 = (((err_A_3 / A_3)*(np.sqrt(3) / 0.01))**2)

print(t1p1, t1p1/(60*24), t1p3, t1p3/(60*24))
```

```
7917.906008328377 5.498545839116928 6087.529693891035 4.2274511763132185
```

[11]:
```
# Aufgabe = (c)

ratio_1_1sig = (N_1[1] + 1 * err_N_1[1] - (N_1[0] - 1 * err_N_1[0]))/(N_1[0] -␣
↪1 * err_N_1[0])

ratio_1_2sig = (N_1[1] + 2 * err_N_1[1] - (N_1[0] - 2 * err_N_1[0]))/(N_1[0] -␣
↪2 * err_N_1[0])

print(ratio_1_1sig * 100, ratio_1_2sig * 100)

ratio_3_1sig = (N_3[1] + 1 * err_N_3[1] - (N_3[0] - 1 * err_N_3[0]))/(N_3[0] -␣
↪1 * err_N_3[0])
```

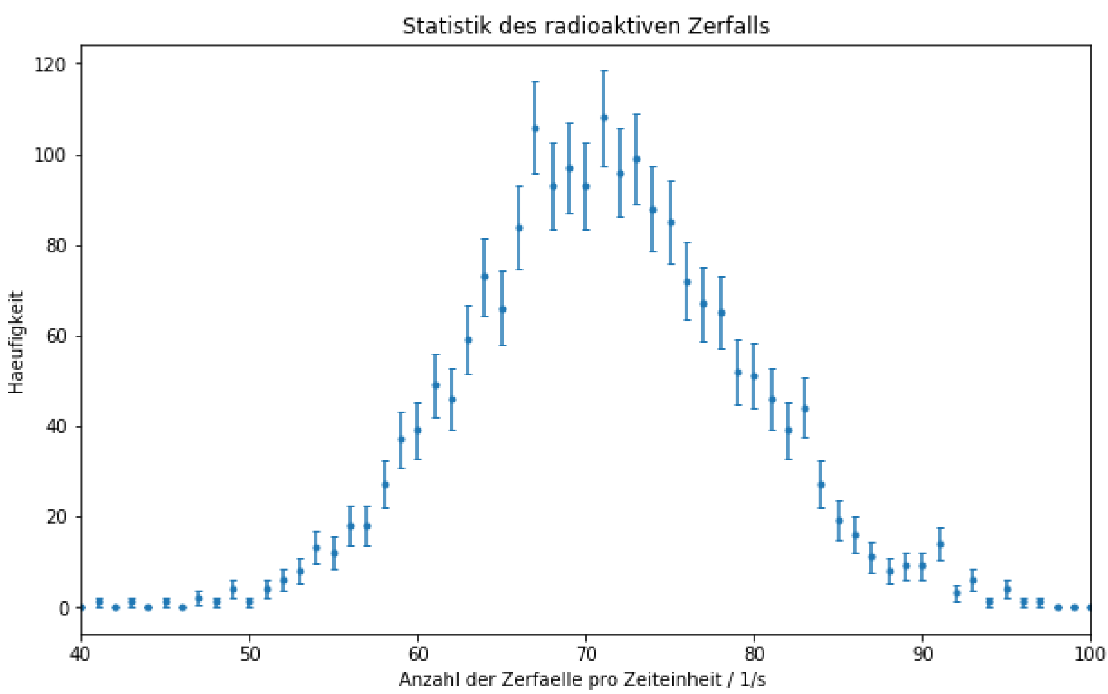
```
ratio_3_2sig = (N_3[1] + 2 * err_N_3[1] - (N_3[0] - 2 * err_N_3[0]))/(N_3[0] -
 ↪2 * err_N_3[0])

print(ratio_3_1sig * 100, ratio_3_2sig * 100)
```

```
3.539800889683748 5.566894430107933
2.939280877583519 4.106464082491329
```

Aufgabe 4

[12]:
```
# Aufgabe 4

anzahl, haeufigkeit=np.loadtxt("data_aufgabe2.txt", unpack=True,
                                delimiter = ",", skiprows = 4, usecols = (0,1) )
fehler=np.sqrt(haeufigkeit)
```

[13]:
```
plt.figure(figsize=(10,6))
plt.errorbar(anzahl, haeufigkeit, fehler, fmt=".",capsize=2)
plt.xlim((40, 100))
plt.xlabel('Anzahl der Zerfaelle pro Zeiteinheit / 1/s ')
plt.ylabel('Haeufigkeit')
plt.title('Statistik des radioaktiven Zerfalls')
plt.savefig("aufgabe4_data.png", format="png")
```



[14]:
```python
def gaussian(x, A, mu, sig): #A: Flaeche der Gaussfunktion
    return A/(np.sqrt(2 * np.pi)*sig)*np.exp(-(x-mu)**2 / (2 * sig**2))
```

[15]:
```python
zerf_gr10 = np.where(haeufigkeit > 10)
zerf_gr10_min = 21
zerf_gr10_max = 58

zerf_gr10
```

[15]:
```
(array([21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
        38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
        58]),)
```

[16]:
```python
popt_g, pcov_g=curve_fit(gaussian, anzahl[zerf_gr10_min:zerf_gr10_max],␣
↪haeufigkeit[zerf_gr10_min:zerf_gr10_max], p0 = [2000, 71.05, 8.23],␣
↪sigma=fehler[zerf_gr10_min:zerf_gr10_max], absolute_sigma = True)
# print(pcov_g)

print(f"A = {popt_g[0]:.2f} +- {np.sqrt(pcov_g[0][0]):.2f}")
print(f"mu = {popt_g[1]:.2f} +- {np.sqrt(pcov_g[1][1]):.2f}")
print(f"sig = {popt_g[2]:.2f} +- {np.sqrt(pcov_g[2][2]):.2f}")
```

```
A = 1965.84 +- 45.20
mu = 70.86 +- 0.20
sig = 8.00 +- 0.17
```

[17]:
```python
print(fehler[zerf_gr10_min:zerf_gr10_max])
```

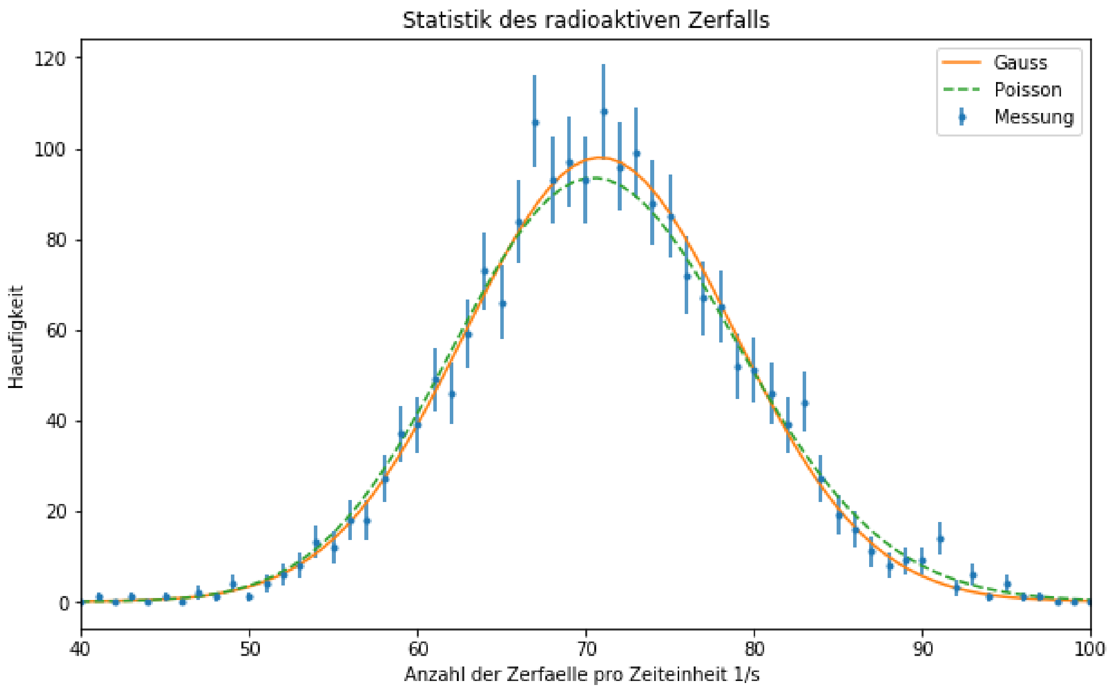
```
[ 3.60555128  3.46410162  4.24264069  4.24264069  5.19615242  6.08276253
  6.244998    7.          6.78232998  7.68114575  8.54400375  8.1240384
  9.16515139 10.29563014  9.64365076  9.8488578   9.64365076 10.39230485
  9.79795897  9.94987437  9.38083152  9.21954446  8.48528137  8.18535277
  8.06225775  7.21110255  7.14142843  6.78232998  6.244998    6.63324958
  5.19615242  4.35889894  4.          3.31662479  2.82842712  3.
  3.        ]
```

[18]:
```python
def poisson(x, A_p, mu_p):
    return A_p*np.exp(-mu_p)*mu_p**x/gamma(x+1)

popt_p, pcov_p = curve_fit(poisson, anzahl[zerf_gr10_min:zerf_gr10_max],␣
↪haeufigkeit[zerf_gr10_min:zerf_gr10_max], p0=[2000, 71.05], sigma =␣
↪fehler[zerf_gr10_min:zerf_gr10_max], absolute_sigma=True)
#print(pcov_g)

print(f"A = {popt_p[0]:.2f} +- {np.sqrt(pcov_p[0][0]):.2f}")
print(f"mu = {popt_p[1]:.2f} +- {np.sqrt(pcov_p[1][1]):.2f}")
```

```
A = 1972.29 +- 45.05
```

```
mu = 70.96 +- 0.20
```

[19]:
```
plt.figure(figsize=(10,6))
plt.errorbar(anzahl,haeufigkeit,fehler, fmt=".",label='Messung')
plt.xlabel('Anzahl der Zerfaelle pro Zeiteinheit 1/s ')
plt.ylabel('Haeufigkeit')
plt.title('Statistik des radioaktiven Zerfalls')
x=np.linspace(min(anzahl), max(anzahl), int(max(anzahl)))

plt.plot(x, gaussian(x,*popt_g), label='Gauss')
plt.plot(x, poisson(x,*popt_p), label='Poisson',linestyle='dashed')
plt.legend()
plt.xlim((40, 100))
# plt.yscale('log')
plt.savefig("aufgabe4_gauss_poisson_fit.png", format="png")
```



[20]:
```
print("Gaussfit:")
print("A=", popt_g[0], ", Standardfehler=", np.sqrt(pcov_g[0][0]))
print("mu=", popt_g[1], ", Standardfehler=", np.sqrt(pcov_g[1][1]))
print("sig=", popt_g[2], ", Standardfehler=", np.sqrt(pcov_g[2][2]))
print("Poissonfit:")
print("A=", popt_p[0], ", Standardfehler=", np.sqrt(pcov_p[0][0]))
print("mu=", popt_p[1], ", Standardfehler=", np.sqrt(pcov_p[1][1]))
```

```
Gaussfit:
```

```
A= 1965.8373349113892 , Standardfehler= 45.20203874142463
mu= 70.85823505791575 , Standardfehler= 0.19655842820679315
sig= 8.004791336335561 , Standardfehler= 0.17160470029907776
Poissonfit:
A= 1972.288272915696 , Standardfehler= 45.05407969780351
mu= 70.96489066870816 , Standardfehler= 0.20287774389457452
```

[21]:
```
#Gauss:
x1 = ((gaussian(anzahl[zerf_gr10_min:
↪zerf_gr10_max],*popt_g)-haeufigkeit[zerf_gr10_min:zerf_gr10_max])**2)
x2 = fehler[zerf_gr10_min:zerf_gr10_max]**2
chi2_div = x1/x2
chi2_g= np.sum(chi2_div)
dof_g=len(anzahl[zerf_gr10_min:zerf_gr10_max])-3 #dof:degrees of freedom,␣
↪Freiheitsgrad
chi2_red_g=chi2_g/dof_g
print("chi2_g=", chi2_g)
print("chi2_red_g=",chi2_red_g)
```

```
chi2_g= 18.49944005335566
chi2_red_g= 0.5441011780398723
```

[22]:
```
#Poisson:
x11 = (poisson(anzahl[zerf_gr10_min:
↪zerf_gr10_max],*popt_p)-haeufigkeit[zerf_gr10_min:zerf_gr10_max])**2
x22 = fehler[zerf_gr10_min:zerf_gr10_max]**2
chi2_div = x11/x22
chi2_p = np.sum(chi2_div)
dof_p = len(anzahl[zerf_gr10_min:zerf_gr10_max])-2 #poisson hat nur 2 Parameter
chi2_red_p=chi2_p/dof_p
print("chi2_p=", chi2_p)
print("chi2_red_p=",chi2_red_p)
```

```
chi2_p= 24.617953649112792
chi2_red_p= 0.7033701042603655
```

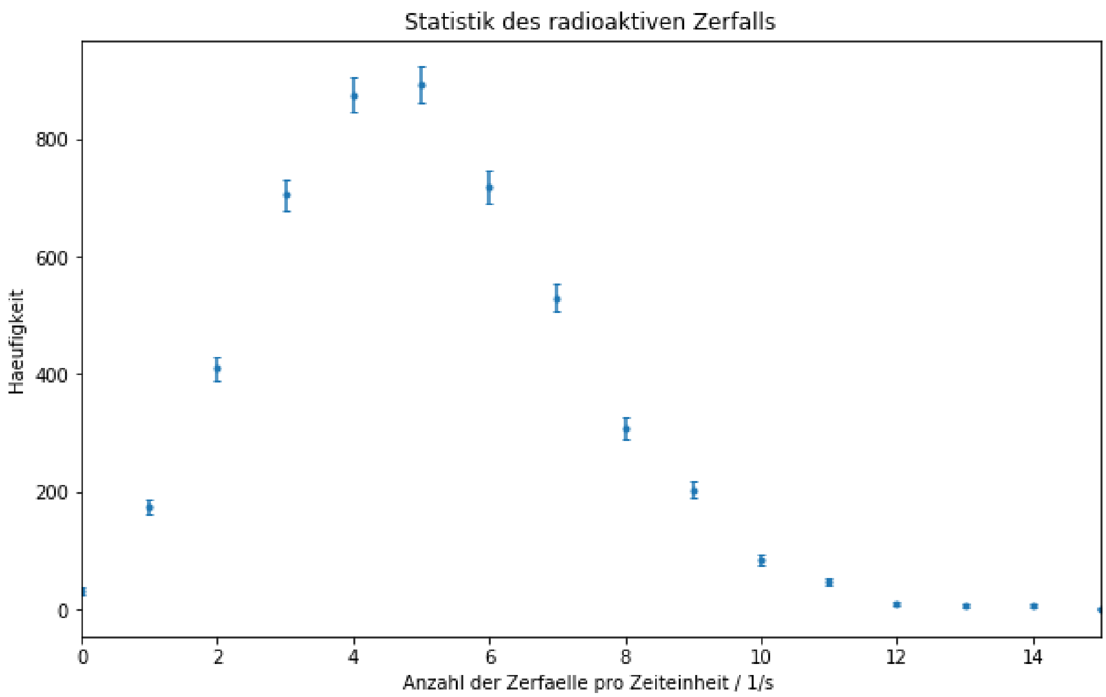
[23]:
```
prob_g = round(1-chi2.cdf(chi2_g, dof_g), 2)*100
prob_p = round(1-chi2.cdf(chi2_p, dof_p), 2)*100
print("Wahrscheinlichkeit Gauss=", prob_g, "%")
print("Wahrscheinlichkeit Poisson=", prob_p, "%")
```

```
Wahrscheinlichkeit Gauss= 99.0 %
Wahrscheinlichkeit Poisson= 90.0 %
```

[24]:
```
# Aufgabe 5

anzahl, haeufigkeit=np.loadtxt("data_aufgabe3.txt", unpack=True,
                               delimiter = ",", skiprows = 4, usecols = (0,1) )
```

```
fehler=np.sqrt(haeufigkeit)
```

[25]:
```
plt.figure(figsize=(10,6))
plt.errorbar(anzahl, haeufigkeit, fehler, fmt=".",capsize=2)
plt.xlim((0, 15))

plt.xlabel('Anzahl der Zerfaelle pro Zeiteinheit / 1/s ')
plt.ylabel('Haeufigkeit')
plt.title('Statistik des radioaktiven Zerfalls')
plt.savefig("aufgabe5_data.png", format="png")
```



[26]:
```
zerf_5_gr10 = np.where(haeufigkeit > 10)
zerf_5_gr10_min = 0
zerf_5_gr10_max = 11
```

[27]:
```
popt_g, pcov_g=curve_fit(gaussian, anzahl[zerf_5_gr10_min:zerf_5_gr10_max],␣
 ↪haeufigkeit[zerf_5_gr10_min:zerf_5_gr10_max], p0 = [5000, 5.01, 2.25],␣
 ↪sigma=fehler[zerf_5_gr10_min:zerf_5_gr10_max], absolute_sigma = True)
print(pcov_g)
```

```
[[4.94368575e+03 1.57823182e-02 1.04571080e-01]
 [1.57823182e-02 1.04433066e-03 1.34655224e-04]
 [1.04571080e-01 1.34655224e-04 6.34671949e-04]]
```

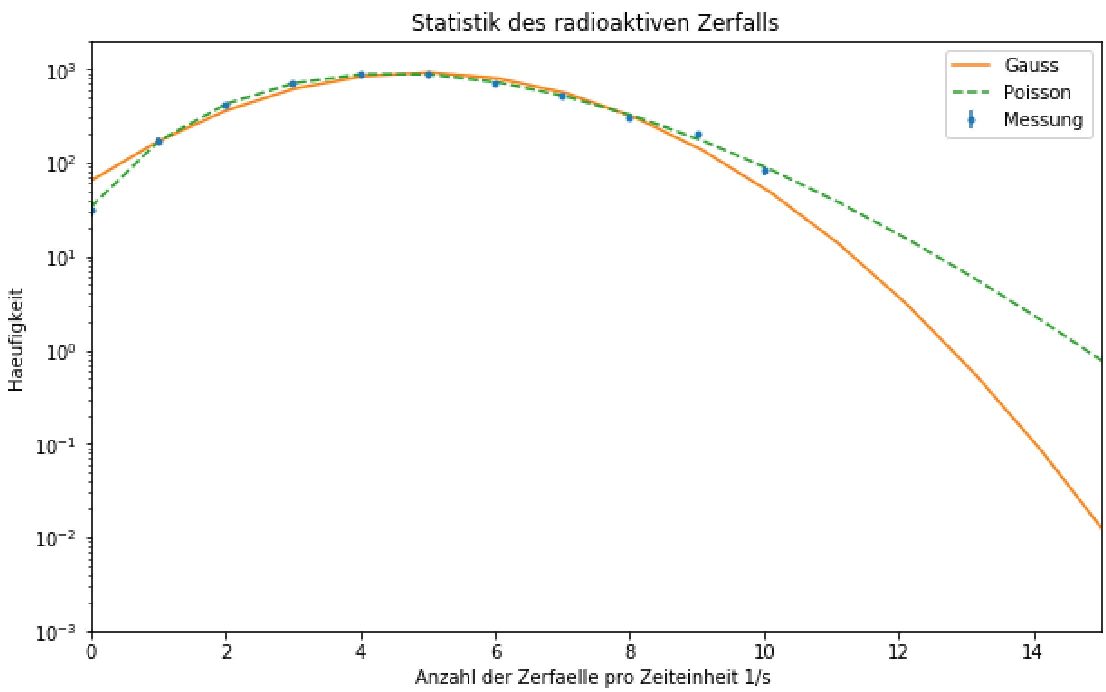
[28]:
```
print(fehler[zerf_5_gr10_min:zerf_5_gr10_max])
```

```
[ 5.65685425 13.19090596 20.22374842 26.55183609 29.58039892 29.86636905
 26.79552201 23.         17.54992877 14.24780685  9.16515139]
```

[29]:
```
def poisson(x, A_p, mu_p):
    return A_p*np.exp(-mu_p)*mu_p**x/gamma(x+1)

popt_p, pcov_p = curve_fit(poisson, anzahl[zerf_5_gr10_min:zerf_5_gr10_max],␣
 ↪haeufigkeit[zerf_5_gr10_min:zerf_5_gr10_max], p0=[5000, 5.01], sigma =␣
 ↪fehler[zerf_5_gr10_min:zerf_5_gr10_max], absolute_sigma=True)
print(pcov_g)
```

```
[[4.94368575e+03 1.57823182e-02 1.04571080e-01]
 [1.57823182e-02 1.04433066e-03 1.34655224e-04]
 [1.04571080e-01 1.34655224e-04 6.34671949e-04]]
```

[30]:
```
fig = plt.figure(figsize=(10,6))
plt.errorbar(anzahl[zerf_5_gr10_min:
 ↪zerf_5_gr10_max],haeufigkeit[zerf_5_gr10_min:
 ↪zerf_5_gr10_max],fehler[zerf_5_gr10_min:zerf_5_gr10_max], fmt=".
 ↪",label='Messung')
plt.xlabel('Anzahl der Zerfaelle pro Zeiteinheit 1/s ')
plt.ylabel('Haeufigkeit')
plt.title('Statistik des radioaktiven Zerfalls')
x=np.linspace(min(anzahl), max(anzahl), int(max(anzahl)))

plt.plot(x, gaussian(x,*popt_g), label='Gauss')
plt.plot(x, poisson(x,*popt_p), label='Poisson',linestyle='dashed')
plt.legend()

plt.yscale('log')

plt.semilogy()
plt.xlim((0,15))
plt.ylim((1e-3,2e3))
plt.savefig("aufgabe5_gauss_poisson_fit.png", format="png")
```



[31]:
```
print("Gaussfit:")
print("A=", popt_g[0], ", Standardfehler=", np.sqrt(pcov_g[0][0]))
print("mu=", popt_g[1], ", Standardfehler=", np.sqrt(pcov_g[1][1]))
print("sig=", popt_g[2], ", Standardfehler=", np.sqrt(pcov_g[2][2]))
print("Poissonfit:")
print("A=", popt_p[0], ", Standardfehler=", np.sqrt(pcov_p[0][0]))
print("mu=", popt_p[1], ", Standardfehler=", np.sqrt(pcov_p[1][1]))
```

```
Gaussfit:
A= 4880.366788352602 , Standardfehler= 70.31134863922186
mu= 4.919135221373067 , Standardfehler= 0.032316105333346779
sig= 2.134058560699586 , Standardfehler= 0.025192696339506696
Poissonfit:
A= 4992.297029938469 , Standardfehler= 71.21313047216086
mu= 5.000742744202284 , Standardfehler= 0.033774351963710 74
```

[32]:
```
#Gauss:
x1 = ((gaussian(anzahl[zerf_5_gr10_min:
↪zerf_5_gr10_max],*popt_g)-haeufigkeit[zerf_5_gr10_min:zerf_5_gr10_max])**2)
x2 = fehler[zerf_5_gr10_min:zerf_5_gr10_max]**2
chi2_div = x1/x2
chi2_g= np.sum(chi2_div)
dof_g=len(anzahl[zerf_5_gr10_min:zerf_5_gr10_max])-3 #dof:degrees of freedom,␣
↪Freiheitsgrad
chi2_red_g=chi2_g/dof_g
```

```
print("chi2_g=", chi2_g)
print("chi2_red_g=",chi2_red_g)
```

```
chi2_g= 94.25083765206813
chi2_red_g= 11.781354706508516
```

[33]:
```
#Poisson:
x11 = (poisson(anzahl[zerf_5_gr10_min:
↪zerf_5_gr10_max],*popt_p)-haeufigkeit[zerf_5_gr10_min:zerf_5_gr10_max])**2
x22 = fehler[zerf_5_gr10_min:zerf_5_gr10_max]**2
chi2_div = x11/x22
chi2_p = np.sum(chi2_div)
dof_p = len(anzahl[zerf_5_gr10_min:zerf_5_gr10_max])-2 #poisson hat nur 2␣
↪Parameter
chi2_red_p=chi2_p/dof_p
print("chi2_p=", chi2_p)
print("chi2_red_p=",chi2_red_p)
```

```
chi2_p= 5.141080127786679
chi2_red_p= 0.5712311253096309
```

[34]:
```
prob_g = round(1-chi2.cdf(chi2_g, dof_g), 2)*100
prob_p = round(1-chi2.cdf(chi2_p, dof_p), 2)*100
print("Wahrscheinlichkeit Gauss=", prob_g, "%")
print("Wahrscheinlichkeit Poisson=", prob_p, "%")
```

```
Wahrscheinlichkeit Gauss= 0.0 %
Wahrscheinlichkeit Poisson= 82.0 %
```

[ ]:
```
```