auswertung256

April 9, 2025

[6]:
```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
from scipy.signal import argrelextrema
from scipy.optimize import curve_fit
from scipy.stats import chi2

plt.rcParams.update({'font.size': 12})

%run ../lib.ipynb
LibFormatter.OutputType = 'latex'

class Literaturwerte:
    E_R = ValErr(13.6, 0) # eV - Praktikumsanleitung
    sig12 = ValErr(1, 0) # https://de.wikipedia.org/wiki/Moseleysches_Gesetz
    sig13 = ValErr(1.8, 0) # https://de.wikipedia.org/wiki/Moseleysches_Gesetz

class Element:
    name = ""
    color = ""
    z = 0
    K_alpha = ValErr(0,0)
    K_beta = ValErr(0,0)

    def __init__(self, name, color, z, ka_val, ka_err, kb_val, kb_err):
        self.name = name
        self.color = color
        self.z = z
        self.K_alpha = ValErr(ka_val, ka_err)
        self.K_beta = ValErr(kb_val, kb_err)

    def __repr__(self) -> str:
        return f"{self.name} ({self.z}): K_a: {self.K_alpha.strfmtf2(2, 0)} /␣
↪K_b: {self.K_beta.strfmtf2(2, 0)}"
```

[7]:
```python
elements = [
    Element("Molybdän", "schwarz", 42, 17.46, 0.18, 19.57, 0.17),
    Element("Eisen", "rot", 26, 6.38, 0.17, 7.03, 0.42),
```

```
    Element("Nickel", "blau", 28, 7.46, 0.18, 8.26, 0.21),
    Element("Zink", "lila", 30, 8.64, 0.18, 9.59, 0.16),
    Element("Zirconium", "cyan", 40, 15.78, 0.18, 17.67, 0.19),
    Element("Titan", "ultramarin", 22, 4.44, 0.19, 4.44, 0.19),
    Element("Kupfer", "pink", 29, 8.04, 0.17, 8.91, 0.14),
    Element("Silber", "rostbraun", 47, 21.89, 0.21, 24.59, 0.18),
]

for i in range(0, len(elements)):
    print(elements[i])
```

```
Molybdän (42): K_a: 17.46 \pm 0.18 / K_b: 19.57 \pm 0.17
Eisen (26): K_a: 6.38 \pm 0.17 / K_b: 7.03 \pm 0.42
Nickel (28): K_a: 7.46 \pm 0.18 / K_b: 8.26 \pm 0.21
Zink (30): K_a: 8.64 \pm 0.18 / K_b: 9.59 \pm 0.16
Zirconium (40): K_a: 15.78 \pm 0.18 / K_b: 17.67 \pm 0.19
Titan (22): K_a: 4.44 \pm 0.19 / K_b: 4.44 \pm 0.19
Kupfer (29): K_a: 8.04 \pm 0.17 / K_b: 8.91 \pm 0.14
Silber (47): K_a: 21.89 \pm 0.21 / K_b: 24.59 \pm 0.18
```

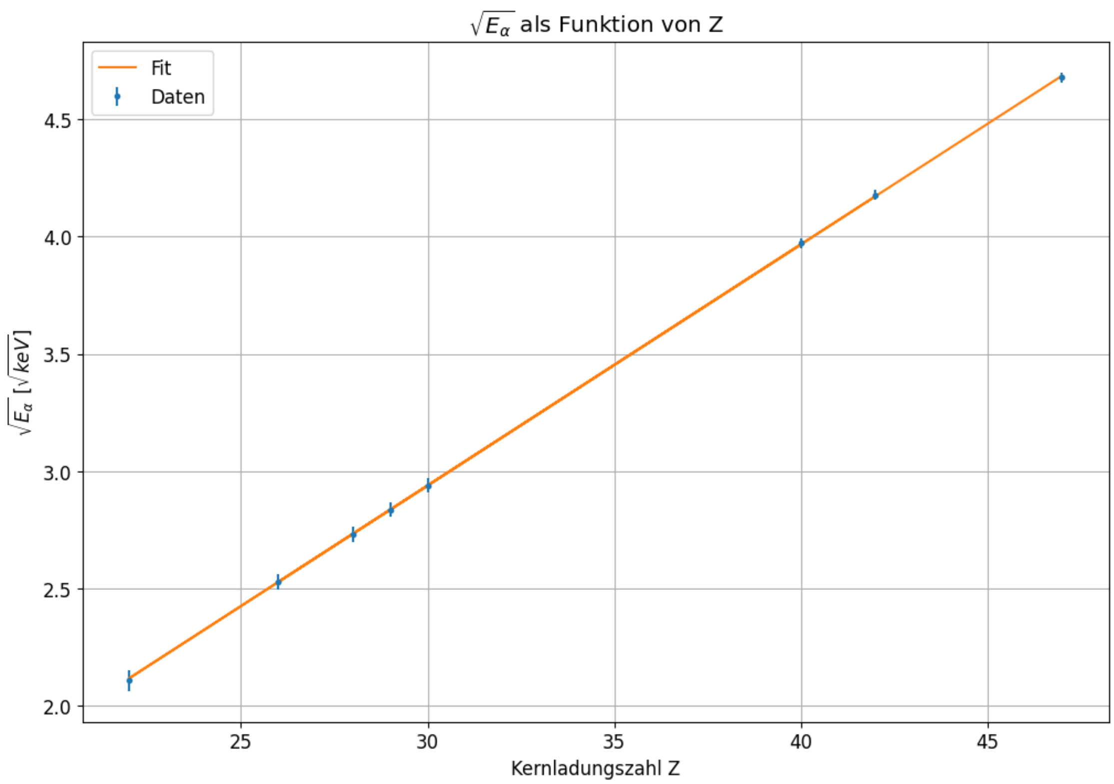
[8]:
```
def get_fit_func(n1, n2):
    def fit_func(x, sqrt_Er, sig12):
        return sqrt_Er * (x - sig12) * np.sqrt((1 / n1**2) - (1 / n2**2))
    return fit_func

Zs = np.array([x.z for x in elements])
```

Auswertung K_α-Linien

[9]:
```
K_alph_vals = np.array([x.K_alpha.val for x in elements])
K_alph_errs = np.array([x.K_alpha.err for x in elements])

sqrt_K_alph_vals = np.sqrt(K_alph_vals)
sqrt_K_alph_errs = (1 / (2 * sqrt_K_alph_vals)) * K_alph_errs

plt.figure(figsize=(12,8))
plt.errorbar(Zs, sqrt_K_alph_vals, sqrt_K_alph_errs, fmt=".", label="Daten")
plt.xlabel('Kernladungszahl Z')
plt.ylabel(r'$\sqrt{E_\alpha}$ [$\sqrt{keV}$]')
plt.title(r'$\sqrt{E_\alpha}$ als Funktion von Z')
plt.grid()
plt.legend()
plt.savefig("K_alpha_vs_Z.png", format="png", bbox_inches='tight')

fitfunc12 = get_fit_func(1, 2)

popt_K_alph, pcov_K_alph = curve_fit(fitfunc12, Zs, sqrt_K_alph_vals,␣
↪sigma=sqrt_K_alph_errs, absolute_sigma=True)
```

```
plt.plot(Zs, fitfunc12(Zs, *popt_K_alph), label="Fit")
plt.legend()
plt.savefig("K_alpha_vs_Z_with_fit.png", format="png", bbox_inches='tight')

sqrt_Er_K_alph = ValErr.fromFit(popt_K_alph, pcov_K_alph, 0)
sig12_K_alph = ValErr.fromFit(popt_K_alph, pcov_K_alph, 1)

Er_K_alph = sqrt_Er_K_alph.pow(2) * 10**3

print_all(
    sqrt_Er_K_alph.strfmtf2(5, 0, "sqrt(E_R)"),
    sig12_K_alph.strfmtf2(5, 0, " _12"),
    sig12_K_alph.sigmadiff_fmt(Literaturwerte.sig12),
    Er_K_alph.strfmtf2(5, 0, "E_R"),
    Er_K_alph.sigmadiff_fmt(Literaturwerte.E_R))
```

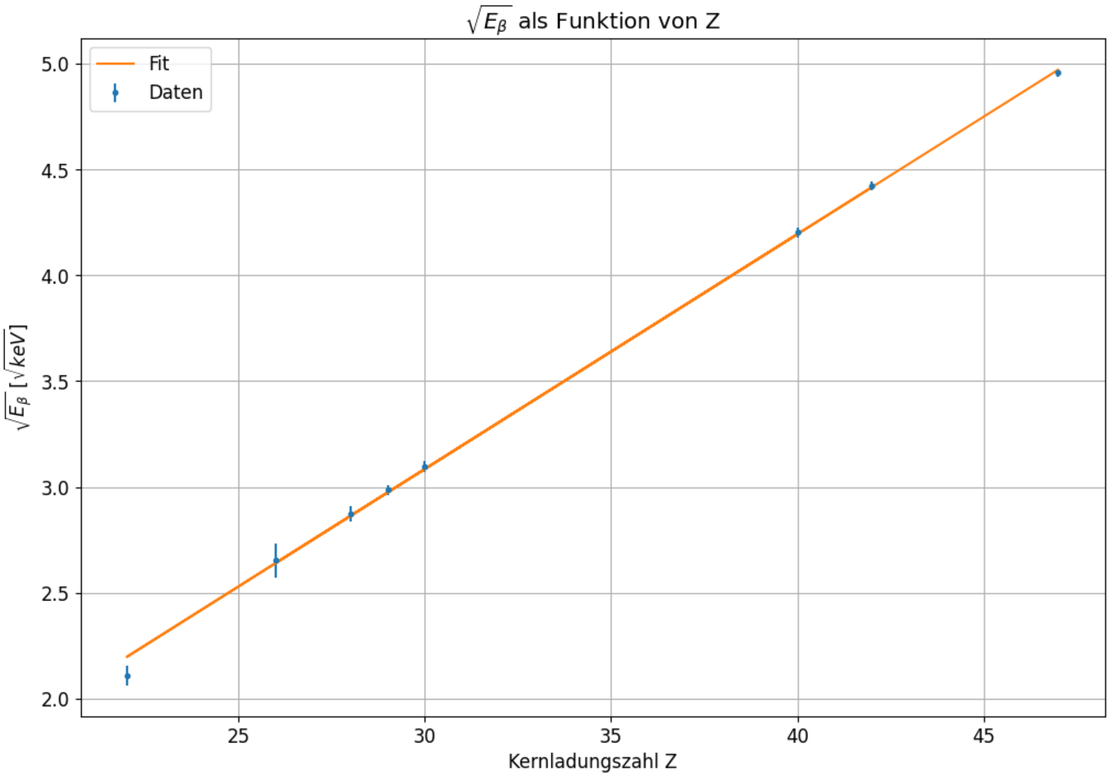
```
sqrt(E_R) = 0.11878 \pm 0.00140
 _12 = 1.43499 \pm 0.42318
1.03\sigma
E_R = 14.10930 \pm 0.33193
1.54\sigma
```



Auswertung K_β-Linien

[10]:
```
K_beta_vals = np.array([x.K_beta.val for x in elements])
K_beta_errs = np.array([x.K_beta.err for x in elements])

sqrt_K_beta_vals = np.sqrt(K_beta_vals)
sqrt_K_beta_errs = (1 / (2 * sqrt_K_beta_vals)) * K_beta_errs

plt.figure(figsize=(12,8))
plt.errorbar(Zs, sqrt_K_beta_vals, sqrt_K_beta_errs, fmt=".", label="Daten")
plt.xlabel('Kernladungszahl Z')
plt.ylabel(r'$\sqrt{E_\beta}$ [$\sqrt{keV}$]')
plt.title(r'$\sqrt{E_\beta}$ als Funktion von Z')
plt.grid()
plt.legend()
plt.savefig("K_beta_vs_Z.png", format="png", bbox_inches='tight')

fitfunc13 = get_fit_func(1, 3)

popt_K_beta, pcov_K_beta = curve_fit(fitfunc13, Zs, sqrt_K_beta_vals,␣
↪sigma=sqrt_K_beta_errs, absolute_sigma=True)

plt.plot(Zs, fitfunc13(Zs, *popt_K_beta), label="Fit")
plt.legend()
plt.savefig("K_beta_vs_Z_with_fit.png", format="png", bbox_inches='tight')

sqrt_Er_K_beta = ValErr.fromFit(popt_K_beta, pcov_K_beta, 0)
sig13_K_beta = ValErr.fromFit(popt_K_beta, pcov_K_beta, 1)

Er_K_beta = sqrt_Er_K_alph.pow(2) * 10**3

print_all(
    sqrt_Er_K_beta.strfmtf2(5, 0, "sqrt(E_R)"),
    sig13_K_beta.strfmtf2(5, 0, " _13"),
    sig13_K_beta.sigmadiff_fmt(Literaturwerte.sig13),
    Er_K_beta.strfmtf2(5, 0, "E_R"),
    Er_K_beta.sigmadiff_fmt(Literaturwerte.E_R))
```

```
sqrt(E_R) = 0.11778 \pm 0.00121
 _13 = 2.22144 \pm 0.37472
1.13\sigma
E_R = 14.10930 \pm 0.33193
1.54\sigma
```



[11]:
```
Er_mean = (Er_K_alph + Er_K_beta) / 2

print_all(
    Er_mean.strfmtf2(5, 0, "E_R"),
    Er_mean.sigmadiff_fmt(Literaturwerte.E_R))
```

```
E_R = 14.10930 \pm 0.23471
2.17\sigma
```

[ ]: