back scattering

December 30, 2021

[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from scipy import optimize
import warnings

warnings.filterwarnings("ignore")
```

1 Datas

[13]:
```
# plateau datas
data_plateau = pd.read_excel("data.xlsx", sheet_name="plateau")
pl_voltage = data_plateau["p_voltage"]
pl_counts = data_plateau["p_counts"]

# al backscattrer
data_al = pd.read_excel("data.xlsx", sheet_name='al_back')
al_thick = data_al['al_thick']
al_0 = data_al['al_0']
al_counts = data_al['al_counts']

# cu backscattrer
data_cu = pd.read_excel("data.xlsx", sheet_name='cu_back')
cu_thick = data_cu['cu_thick']
cu_0 = data_cu['cu_0']
cu_counts = data_cu['cu_counts']

# ag backscattrer
data_ag = pd.read_excel("data.xlsx", sheet_name='ag_back')
ag_thick = data_ag['ag_thick']
ag_0 = data_ag['ag_0']
ag_counts = data_ag['ag_counts']
```

2 Functions

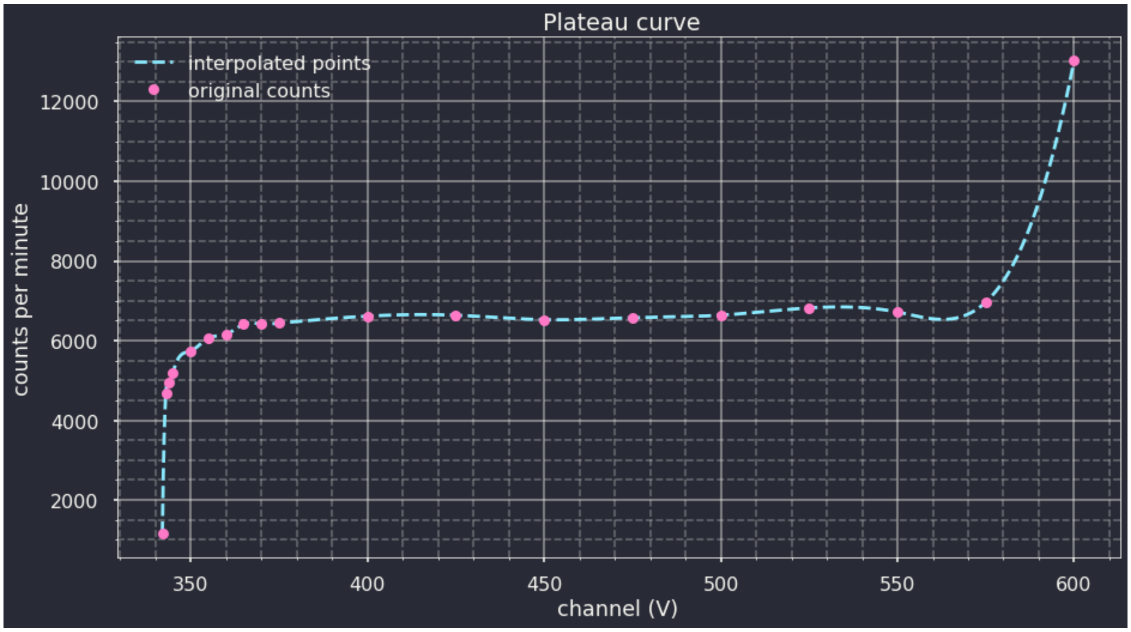
[14]:
```python
# curve fit
def cur_fit(x,y):
    func = lambda t, a, c, d: a*np.log(t + c) + d
    popt, pcov = optimize.curve_fit(func, x, y)
    xx = np.arange(x[0], x[len(x) - 1], 0.001)
    yy = func(xx, *popt)
    return xx, yy


# function for interpolation
def interpolate(x, y):
    f = interp1d(x, y, kind="cubic", fill_value="extrapolate")
    a = np.arange(x[0], x[len(x) - 1], 0.001)
    b = f(a)
    return a, b
```

3 Plateau

[15]:
```python
voltage_interpolated_pl, counts_interpolated_pl = interpolate(pl_voltage,␣
 ↪pl_counts)

# plt.style.use("seaborn-poster")
plt.style.use("dracula")
plt.figure(figsize=(15, 8))
plt.title(f" Plateau curve")
plt.xlabel("channel (V)")
plt.ylabel("counts per minute")
plt.plot(voltage_interpolated_pl, counts_interpolated_pl, "--",␣
 ↪label="interpolated points")
plt.plot(pl_voltage, pl_counts, "o", markersize=9, label="original counts")
plt.legend(loc="upper left")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")
plt.show()
```



I choosed the operating voltage at 400 V

4 Backscattering factor

[16]:
```
element_name = ["Aluminium", "Copper", "Silver"]
```

4.1 Aluminium

[17]:
```
f_al = al_counts/al_0
del_f_al = f_al*np.sqrt(1/al_0 + 1/al_counts)
print(f"{f_al}, {del_f_al}")
```

```
0    1.010056
1    1.025463
2    1.072005
3    1.060553
4    1.110214
5    1.063510
6    1.087963
7    1.014140
8    1.121895
9    1.095922
dtype: float64, 0    0.033678
1    0.034670
2    0.035770
3    0.035843
```

```
4    0.036768
5    0.035596
6    0.036257
7    0.033990
8    0.037084
9    0.036323
dtype: float64
```

[18]:

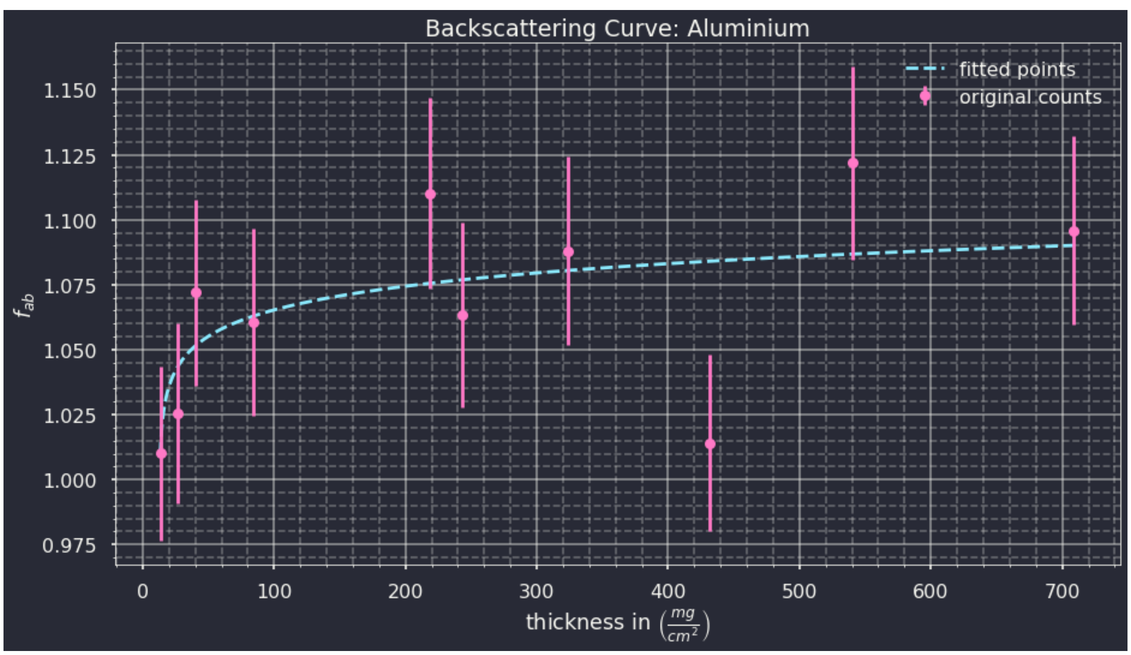
```python
# plotting the curves
# plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))

plt.title(f"Backscattering Curve: {element_name[0]}")
plt.xlabel(r"thickness in $\left(\frac{mg}{cm^2}\right)$")
plt.ylabel(r"$f_{ab}$")

pxal, pyal = cur_fit(al_thick, f_al)
plt.plot(pxal, pyal, "--", label="fitted points")
plt.errorbar(al_thick, f_al, yerr=del_f_al, fmt="o", markersize=9,␣
 ↪label="original counts")

plt.legend(loc="upper right")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.show()
```



4.2 Copper

[19]:
```
f_cu = cu_counts/cu_0
del_f_cu = f_cu*np.sqrt(1/cu_0 + 1/cu_counts)
# cu_thick_interpolate, f_cu_interpolate = polfit(cu_thick,f_cu, 4)
print(f"{f_cu}, {del_f_cu}")
```

```
0    1.212912
1    1.267908
2    1.233848
3    1.245614
4    1.235046
dtype: float64, 0    0.042935
1    0.045385
2    0.043995
3    0.044305
4    0.044075
dtype: float64
```

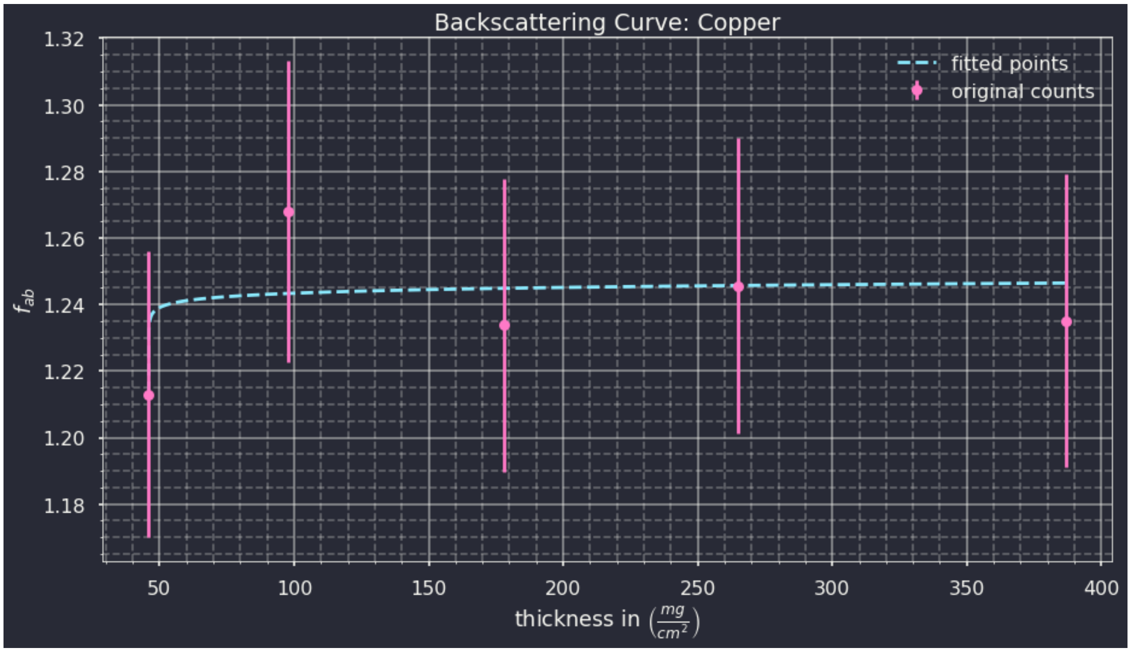
[20]:
```
# plotting the curves
# plt.style.use("seaborn-poster")
plt.figure(figsize=(15, 8))

plt.title(f"Backscattering Curve: {element_name[1]}")
plt.xlabel(r"thickness in $\left(\frac{mg}{cm^2}\right)$")
plt.ylabel(r"$f_{ab}$")

pxcu, pycu = cur_fit(cu_thick, f_cu)
plt.plot(pxcu, pycu, "--", label="fitted points")
plt.errorbar(cu_thick, f_cu, yerr=del_f_cu, fmt="o", markersize=9,␣
 ↪label="original counts")

plt.legend(loc="upper right")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.show()
```



4.3 Silver

[21]:
```python
f_ag = ag_counts/ag_0
del_f_ag = f_ag*np.sqrt(1/ag_0 + 1/ag_counts)
print(f"{f_ag}, {del_f_ag}")
```

```
0    1.267030
1    1.259109
2    1.274354
3    1.325000
4    1.328826
5    1.327477
6    1.335910
7    1.342254
dtype: float64, 0    0.044234
1    0.043810
2    0.043826
3    0.046253
4    0.046931
5    0.046597
6    0.046829
7    0.047053
dtype: float64
```

[24]:
```python
# plotting the curves
# plt.style.use("seaborn-poster")
```

```python
plt.figure(figsize=(15, 8))

plt.title(f"Backscattering Curve: {element_name[2]}")
plt.xlabel(r"thickness in $\left(\frac{mg}{cm^2}\right)$")
plt.ylabel(r"$f_{ab}$")

pxag, pyag = cur_fit(ag_thick,  f_ag)
plt.plot(pxag, pyag, "--", label="fitted points")
plt.errorbar(ag_thick, f_ag, yerr=del_f_ag, fmt="o", markersize=9,␣
 ↪label="original counts")

plt.legend(loc="upper right")
plt.grid(alpha=0.5, which="major")
plt.minorticks_on()
plt.grid(alpha=0.3, which="minor", ls="--")

plt.show()
```