

# Auswertung

February 12, 2024

```
[1]: import os
import numpy as np
import pandas as pd
import seaborn as sns

from io import BytesIO
from matplotlib import pyplot as plt
from scipy.stats import linregress
```

```
[2]: sns.set_theme(context='paper', style="whitegrid", color_codes=True)

plt.rcParams["axes.titlesize"] = 13 # default: 9
plt.rcParams["axes.labelsize"] = 13 # default: 9
plt.rcParams["legend.fontsize"] = 11 # default: 8.8
plt.rcParams["legend.title_fontsize"] = 11 # default: 8.8
plt.rcParams["xtick.labelsize"] = 11 # default: 8.8
plt.rcParams["ytick.labelsize"] = 11 # default: 8.8
```

## 0.1 Methods

```
[3]: x_column = r'$z_0$ [\AA]$\n$'
y_column = r'$\ln(I(z_0))$ [\ln(\mathrm{A})]$\n$'
```

```
[4]: def read_vert_file(path) -> pd.DataFrame:
    """
    Read a *.VERT file from the given path.

    1. Read the file and create a list of lines, which are seperated by \r\n
    2. Extract the Title field
    3. Extract the data: Starting 2 lines after a line named "DATA"
    4. Convert data into DataFrame
    """
    # read
    with open(path, 'rb') as io:
        txt = io.read()
    txt = txt.split(b'\r\n')

    # extract title
```

```

for line in txt:
    if line.startswith(b'Titel'):
        title = line.split(b'=')[1]
        break

# extract data
for index, line in enumerate(txt):
    if b'\nDATA' in line:
        data = txt[index+2:] # data start 2 lines after the found line
        break

# convert to df
with BytesIO(b'\n'.join(data)) as io:
    df = pd.read_csv(io, sep='\t', index_col=0, header=None)

df.attrs['title'] = os.path.basename(path)
df = df.drop(columns=4)
df = df.rename(columns={
    1: 'U [V]',
    2: x_column,
    3: '$I(z_0)$'
})
df[x_column] *= 0.0024
df[y_column] = np.log(df['$I(z_0)$'])

return df

```

```

[5]: def plot(data, title=None, filename=None):
    data = data.copy()
    img = sns.relplot(
        data=data,
        x=x_column,
        y=y_column
    )
    if title is not None:
        plt.title(title)
    else:
        plt.title(data.attrs['title'])
    if filename is not None:
        img.figure.savefig(filename, bbox_inches='tight')

```

```

[6]: def read_files(folder):
    data = {}
    for dirpath, dirnames, filenames in os.walk(folder):
        for f in filenames:
            filename = f"{dirpath}/{f}"
            print('read file: ', filename)

```

```

        df = read_vert_file(filename)
        data[df.attrs['title']] = df
    return data

```

```

[7]: def regression(df):
    result = linregress(
        x=df[x_column],
        y=df[y_column]
    )
    print(f"$m = {round(result.slope, 2)} \pm {round(result.stderr, 2)}$")

    x_values = np.linspace(df[x_column].min(), df[x_column].max())
    fig = plt.errorbar(
        x_values,
        result.slope * x_values + result.intercept,
        yerr=np.abs(result.stderr * x_values + result.intercept_stderr)
    )

    return result.slope, result.stderr

```

# 1 A

## 1.0.1 Messwerte

```

[8]: data = read_files('a')

```

```

read file: a/A231128.133315.VERT
read file: a/A231128.133424.VERT
read file: a/A231128.134137.VERT
read file: a/A231128.134156.VERT
read file: a/A231128.134230.VERT
read file: a/A231128.134345.VERT
read file: a/A231128.134401.VERT
read file: a/A231128.134418.VERT
read file: a/A231128.134446.VERT

```

```

[9]: data['A231128.133315.VERT']

```

```

[9]:      U [V]  $z_0\ [\AA]$  $I(z_0)$  $\ln(I(z_0))\ [\ln(\mathrm{A})]$
0
0      500.0      0.0  49948.9      10.818756
1      500.0      0.0  47002.5      10.757956
2      500.0      0.0  48437.0      10.788019
3      500.0      0.0  51164.5      10.842801
4      500.0      0.0  49354.3      10.806780
...
1995  500.0      0.0  40239.8      10.602612
1996  500.0      0.0  40602.4      10.611582

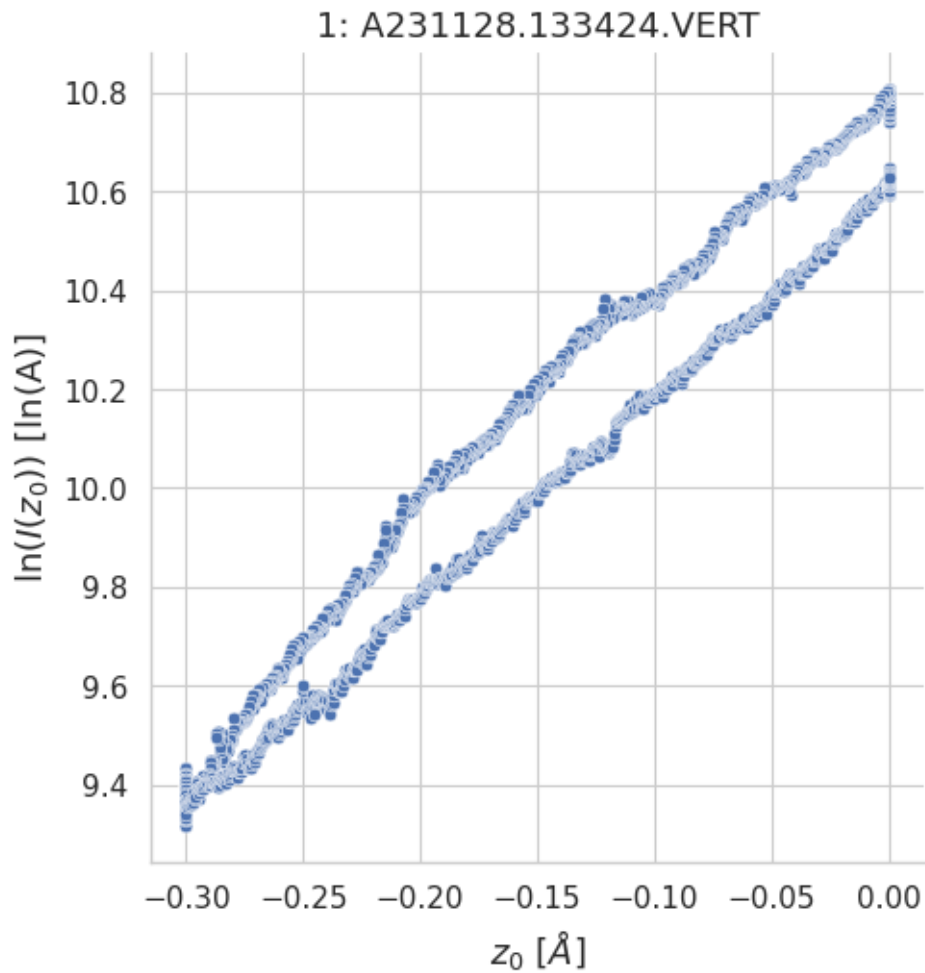
```

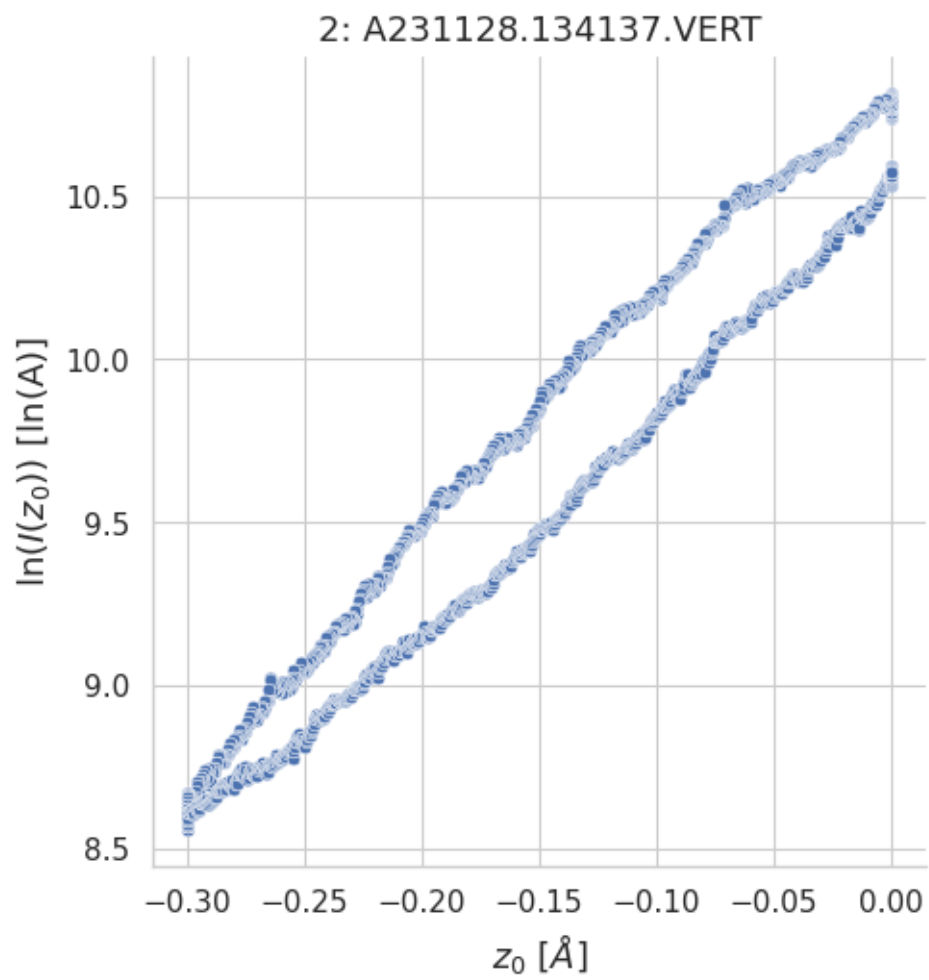
1997	500.0	0.0	40734.4	10.614828
1998	500.0	0.0	40515.4	10.609437
1999	500.0	0.0	40497.2	10.608988

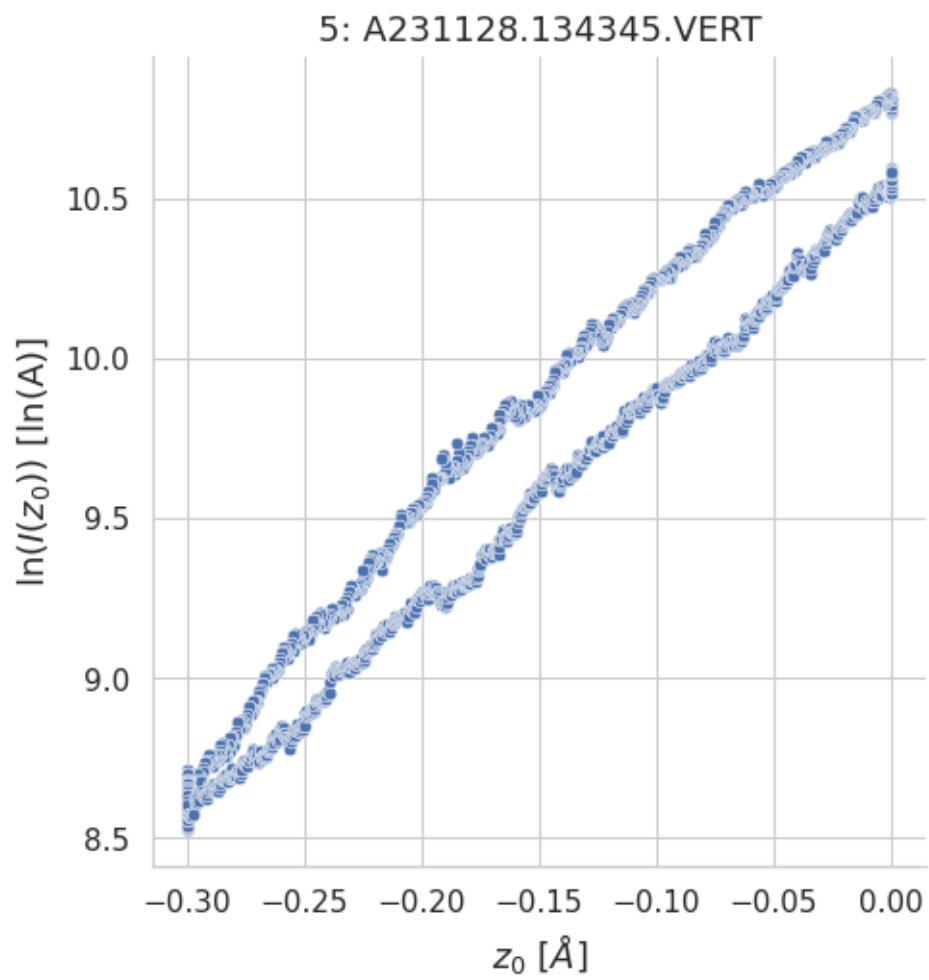
[2000 rows x 4 columns]

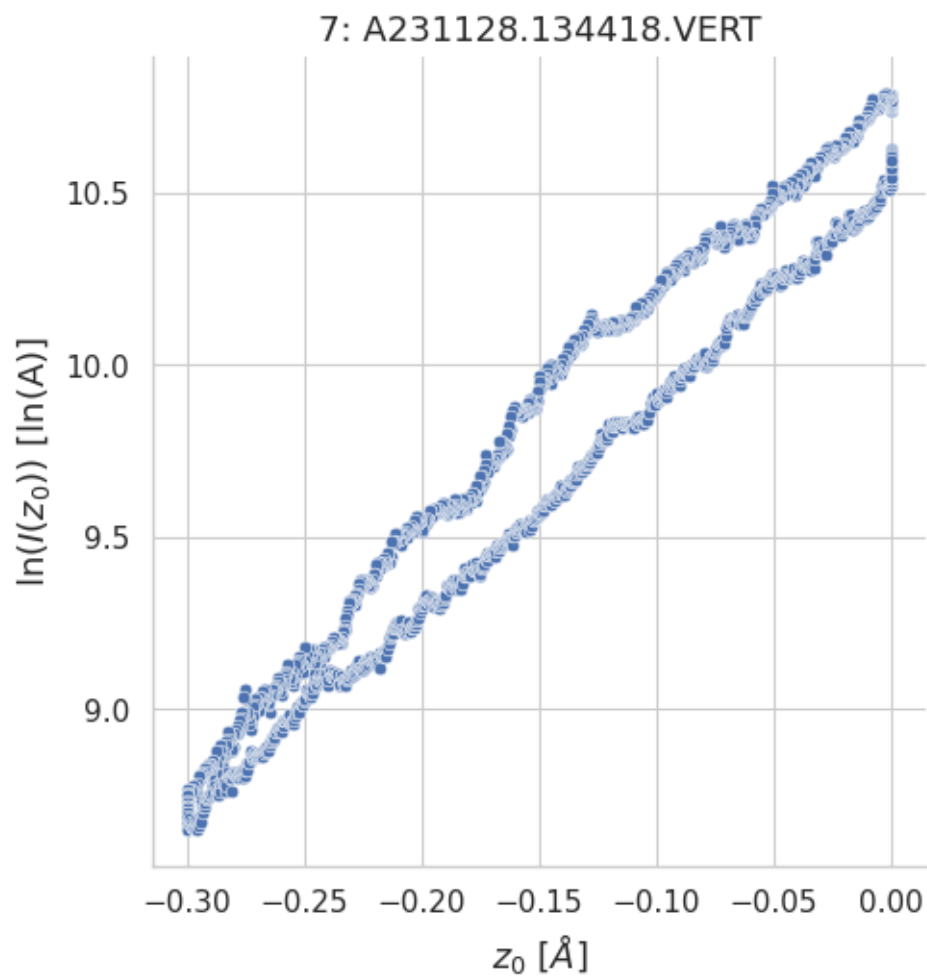
```
[10]: # # Plotte alle Messungen
# for i, key in enumerate(data.keys()):
#     plot(data[key], title=f"{i}: {key}")
```

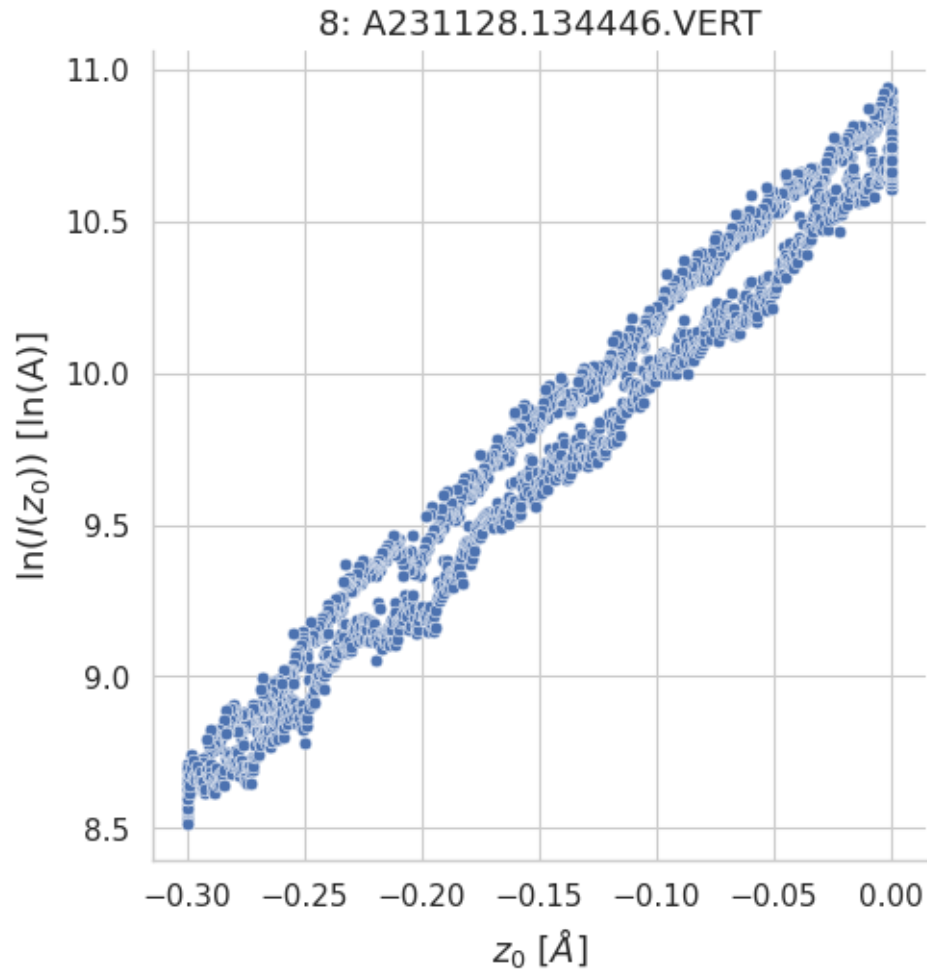
```
[11]: # Plotte die 5 besten Messungen
selected_measurements_a = []
for i, key in enumerate(data.keys()):
    if i in (2, 5, 1, 7, 8):
        selected_measurements_a.append(data[key])
        plot(data[key], title=f"{i}: {key}")
```











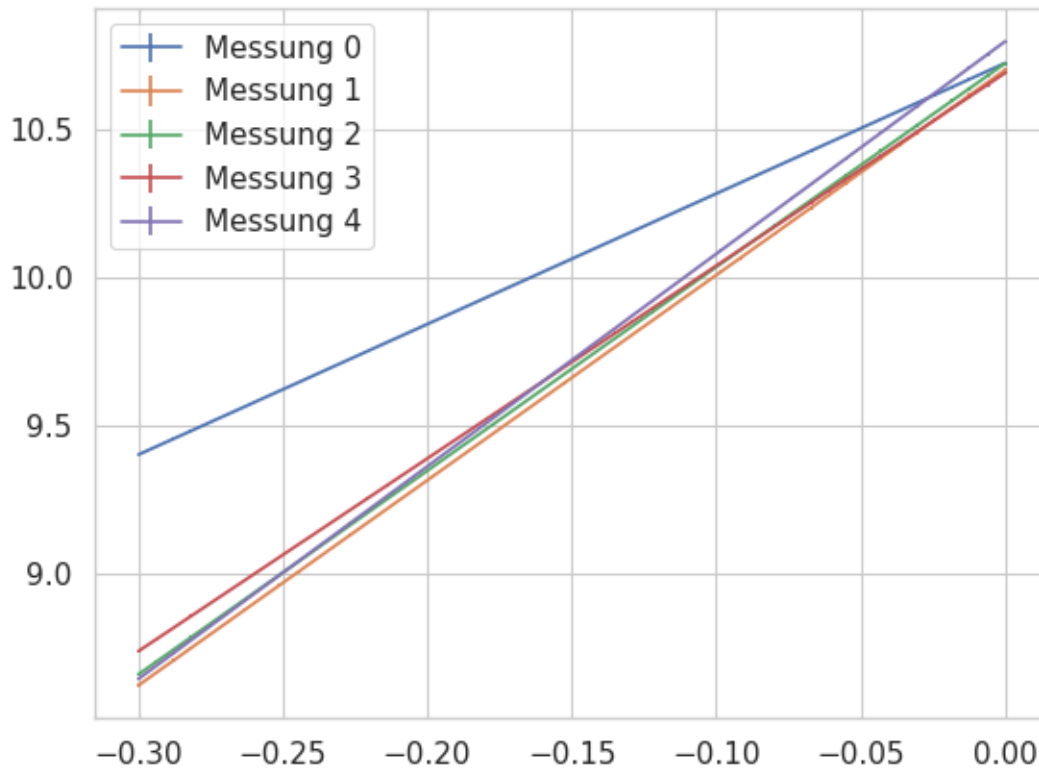
### 1.0.2 Regression

```
[12]: for df in selected_measurements_a:
        regression(df)
        plt.legend([ f"Messung {i}" for i in range(5)])
```

```
$m = 4.4 \pm 0.02$
$m = 6.91 \pm 0.03$
$m = 6.86 \pm 0.03$
$m = 6.49 \pm 0.03$
$m = 7.16 \pm 0.02$
```

```
[12]: <matplotlib.legend.Legend at 0x7f149888f0e0>
```





## 2 B

### 2.0.1 Messwerte

```
[13]: data = read_files('b')
```

```
read file: b/A231128.134613.VERT
read file: b/A231128.134739.VERT
read file: b/A231128.134817.VERT
read file: b/A231128.134842.VERT
read file: b/A231128.134923.VERT
read file: b/A231128.134937.VERT
read file: b/A231128.134951.VERT
read file: b/A231128.135005.VERT
read file: b/A231128.135041.VERT
```

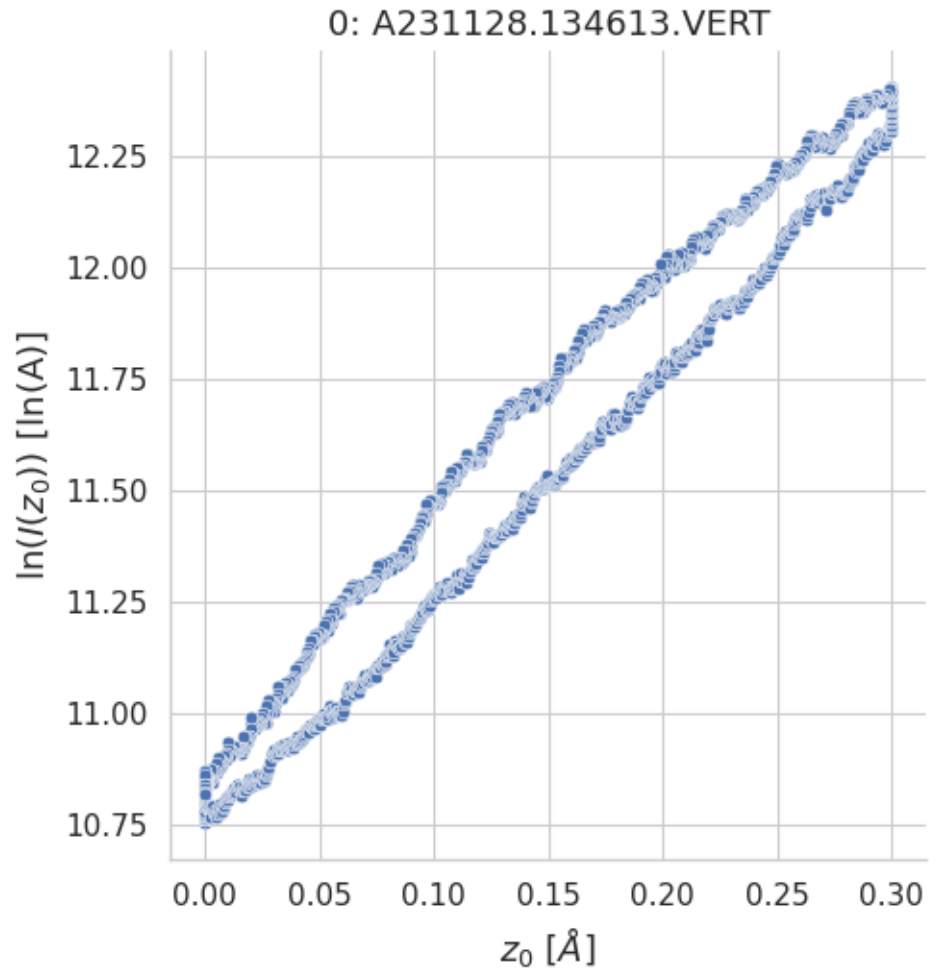
```
[14]: # # Plotte alle Messungen
# for i, key in enumerate(data.keys()):
#     plot(data[key], title=f"{i}: {key}")
```

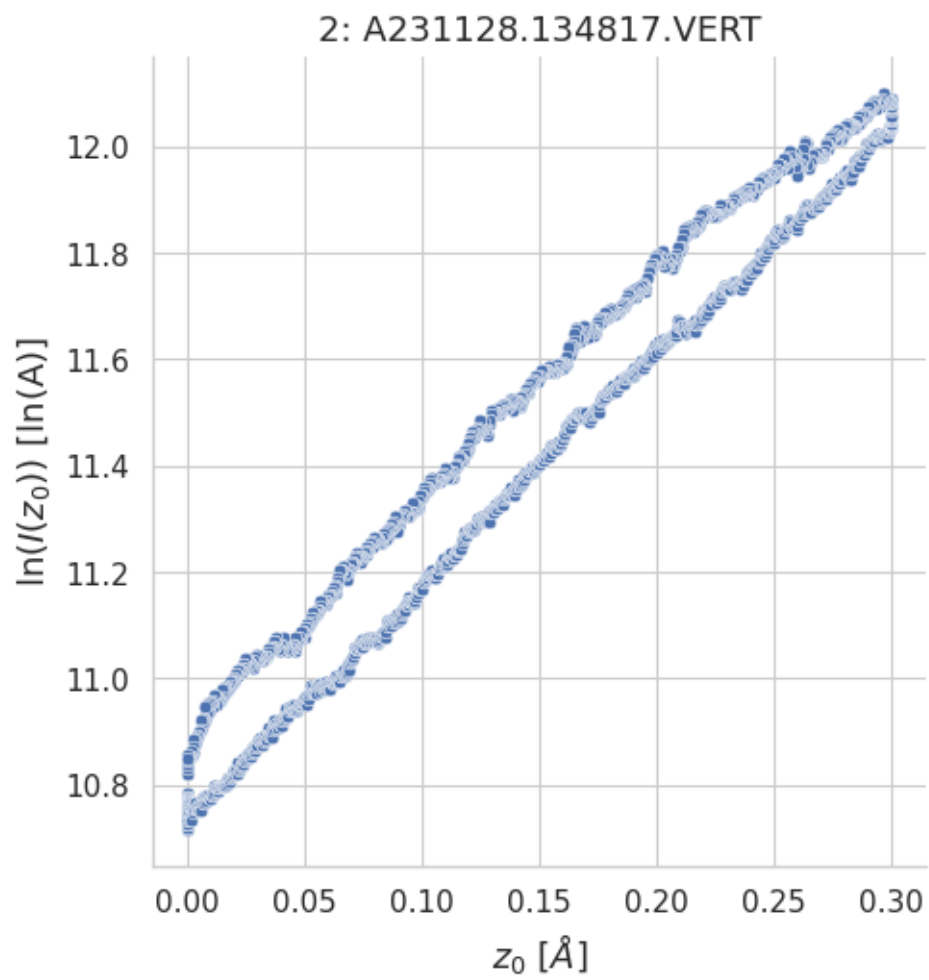
```
[15]: # Plotte die 5 besten Messungen
selected_measurements_b = []
```

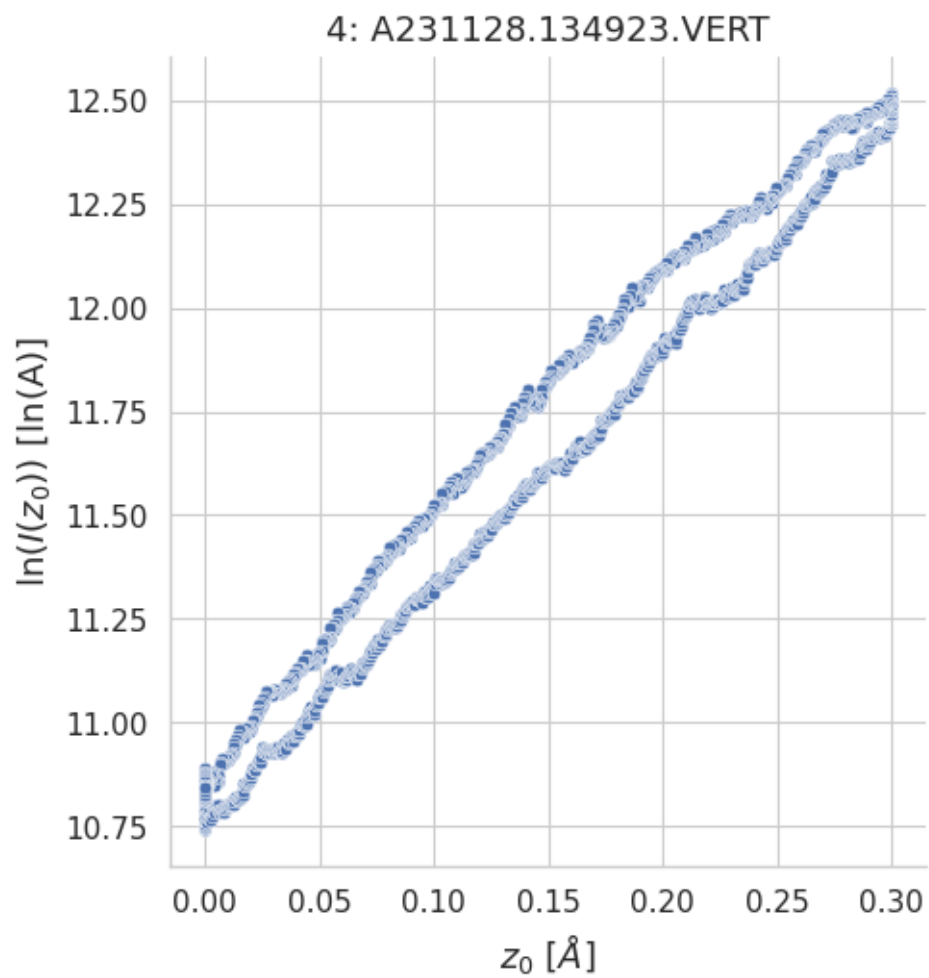
```

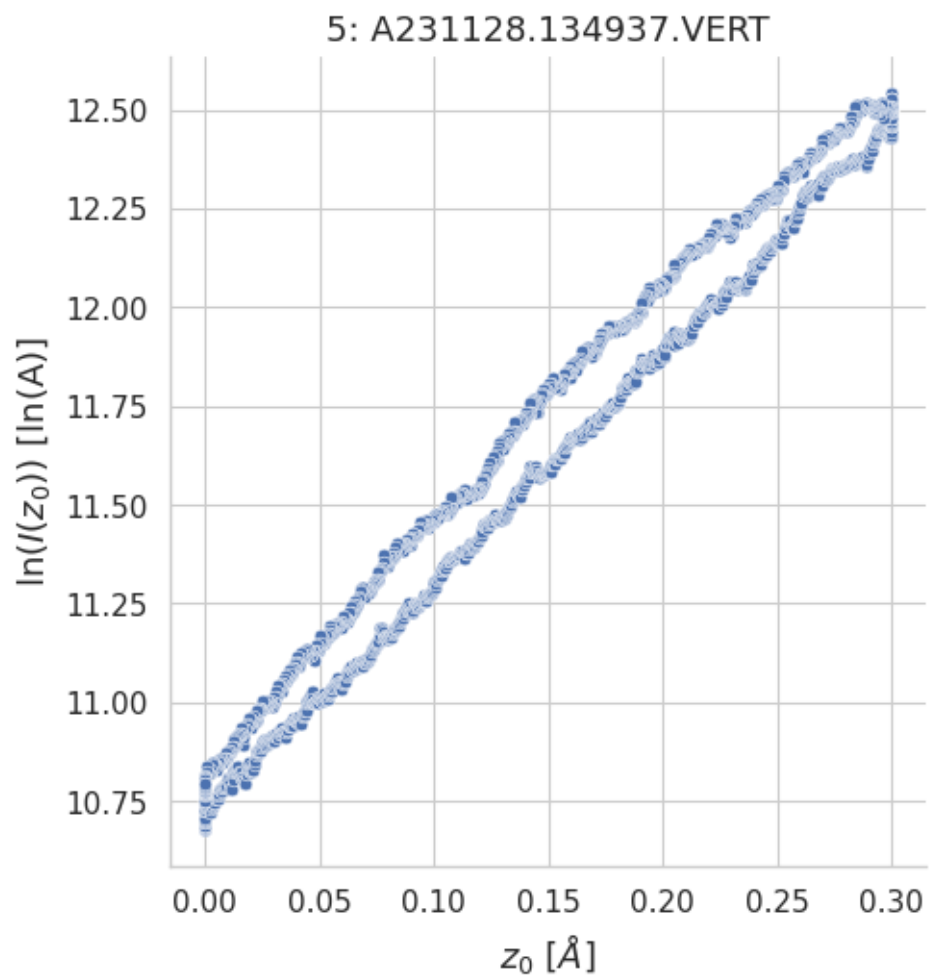
for i, key in enumerate(data.keys()):
    if i in (2, 4, 5, 6, 0):
        selected_measurements_b.append(data[key])
        plot(data[key], title=f"{i}: {key}")

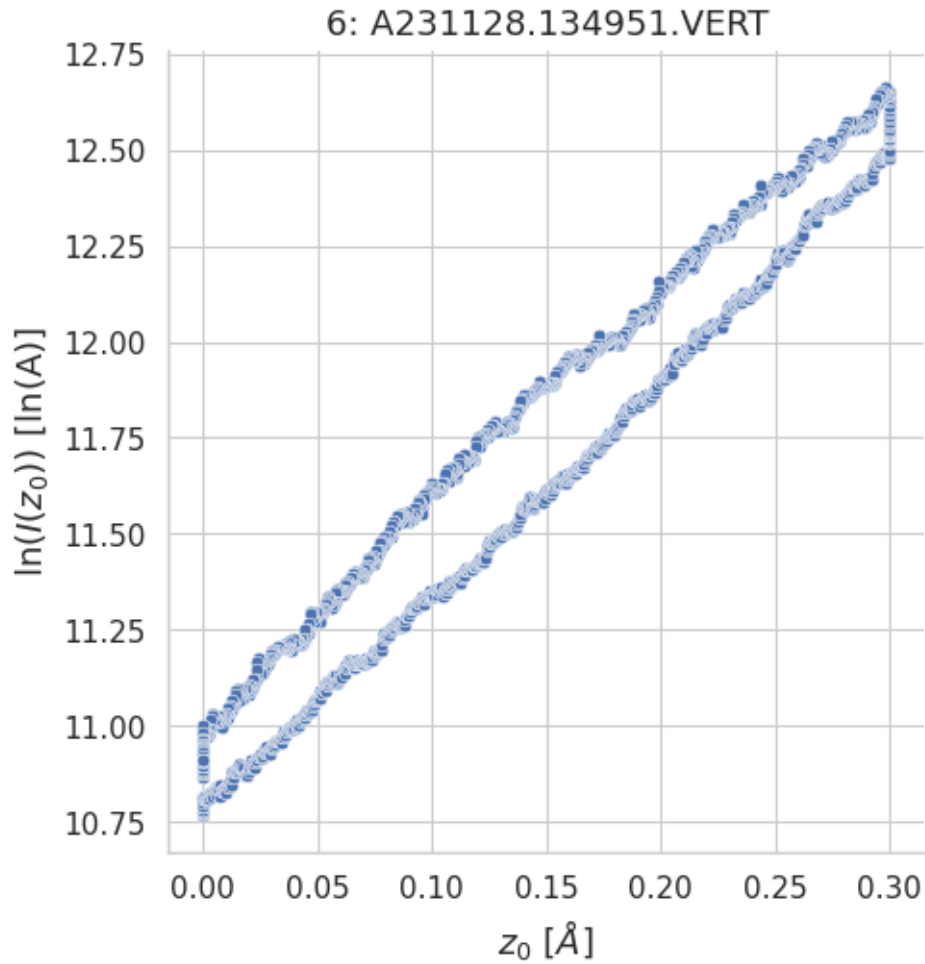
```











## 2.0.2 Regression

```
[16]: # plote alle Regressionen
      # for df in selected_measurements_a:
      #     plot_regression(df)
```

```
[17]: for df in selected_measurements_b:
      regression(df)
      plt.legend([ f"Messung {i}" for i in range(5)])
```

$m = 5.18 \pm 0.02$

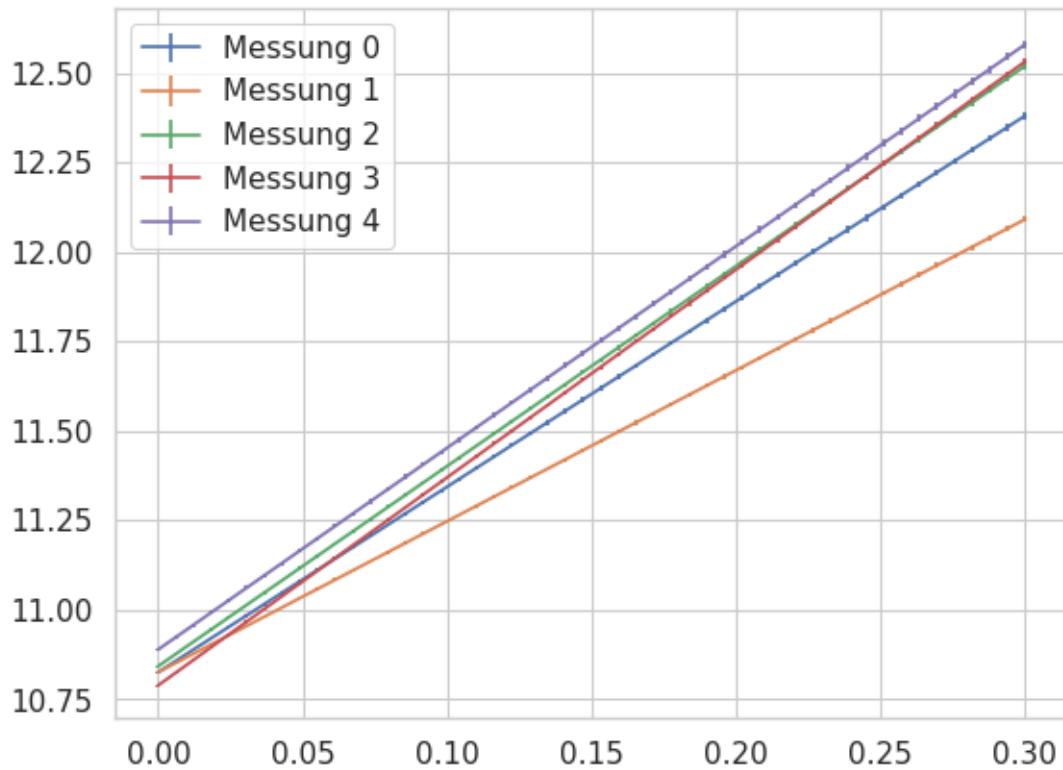
$m = 4.22 \pm 0.02$

$m = 5.59 \pm 0.02$

$m = 5.81 \pm 0.02$

$m = 5.64 \pm 0.02$

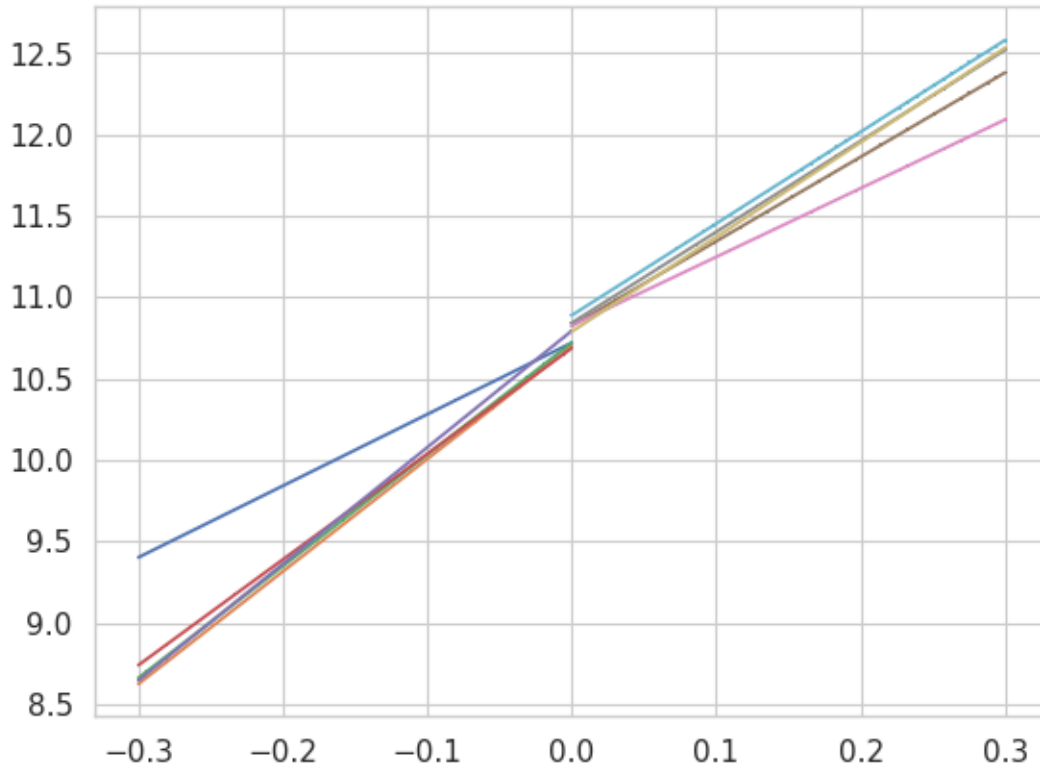
```
[17]: <matplotlib.legend.Legend at 0x7f14982b8890>
```



### 3 A und B

```
[18]: ms = []
      errs = []
      for df in selected_measurements_a + selected_measurements_b:
          m, err = regression(df)
          ms.append(m)
          errs.append(err)
```

```
$m = 4.4 \pm 0.02$
$m = 6.91 \pm 0.03$
$m = 6.86 \pm 0.03$
$m = 6.49 \pm 0.03$
$m = 7.16 \pm 0.02$
$m = 5.18 \pm 0.02$
$m = 4.22 \pm 0.02$
$m = 5.59 \pm 0.02$
$m = 5.81 \pm 0.02$
$m = 5.64 \pm 0.02$
```



```
[19]: for i in range(10):
    phi = (ms[i]/0.51)**2
    err_phi = (errs[i]/0.51)**2
    print(f'${round(ms[i], 2)} \pm {round(errs[i], 2)}$ & ${round(phi, 3)}$ \pm
    \pm {round(err_phi, 3)}$ \pm \pm')
```

```
$4.4 \pm 0.02$ & $74.283 \pm 0.002$ \pm
$6.91 \pm 0.03$ & $183.471 \pm 0.004$ \pm
$6.86 \pm 0.03$ & $180.925 \pm 0.004$ \pm
$6.49 \pm 0.03$ & $161.996 \pm 0.003$ \pm
$7.16 \pm 0.02$ & $196.918 \pm 0.002$ \pm
$5.18 \pm 0.02$ & $103.284 \pm 0.002$ \pm
$4.22 \pm 0.02$ & $68.402 \pm 0.001$ \pm
$5.59 \pm 0.02$ & $120.152 \pm 0.001$ \pm
$5.81 \pm 0.02$ & $129.797 \pm 0.001$ \pm
$5.64 \pm 0.02$ & $122.115 \pm 0.002$ \pm
```

```
[20]: 196.918/4, 68.402/4
```

```
[20]: (49.2295, 17.1005)
```