Auswertung

November 25, 2023

0.0.1 Vorbereitungen

yum install texlive-collection-latexextra texlive-collection-mathscience python-pip pandoc

pip install –user notebook pandas seaborn scipy

[1]:
```
import math
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from scipy.stats import linregress
```

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

[3]:
```
H_column = r'$H$ in $10^3 \frac{A}{m}$'
H_column_detailed = r'$H$ in $\frac{A}{m}$'
I_column = r'$I_{max}$ in A'
M_column = r'M in $10^6\ \frac{A}{m}$'
M_column_detailed = r'M in $10^3\ \frac{A}{m}$'
```

[4]:
```
def plot(data, hue_column=I_column, filename=None):
    img = sns.relplot(
        data=data,
        x=H_column,
        y=M_column,
        hue=hue_column,
        height=5,
        legend='full',

    )
    if filename is not None:
        img.figure.savefig(filename, bbox_inches='tight')
```

[5]:
```
def subplot(data, x_column=H_column, y_column=M_column, axis=None):
    return sns.scatterplot(
        data=data,
        x=x_column,
        y=y_column,
        hue=I_column,
        marker='x',
        ax=axis
    )
```

0.1 3.3.1

[6]:
```
heizbar_a = pd.read_csv("3.3.1.a.csv", sep='\t')
heizbar_b = pd.read_csv("3.3.1.b.csv", sep='\t')
heizbar_c = pd.read_csv("3.3.1.c.csv", sep='\t')
heizbar_d = pd.read_csv("3.3.1.d.csv", sep='\t')
```

[10]:
```
def H(U):
    U_max = heizbar_a.H.max()
    n_p=17
    r=1.5/100 # m
    return n_p/(2 * math.pi * r) * (3.0/U_max) * U / 1e3
```

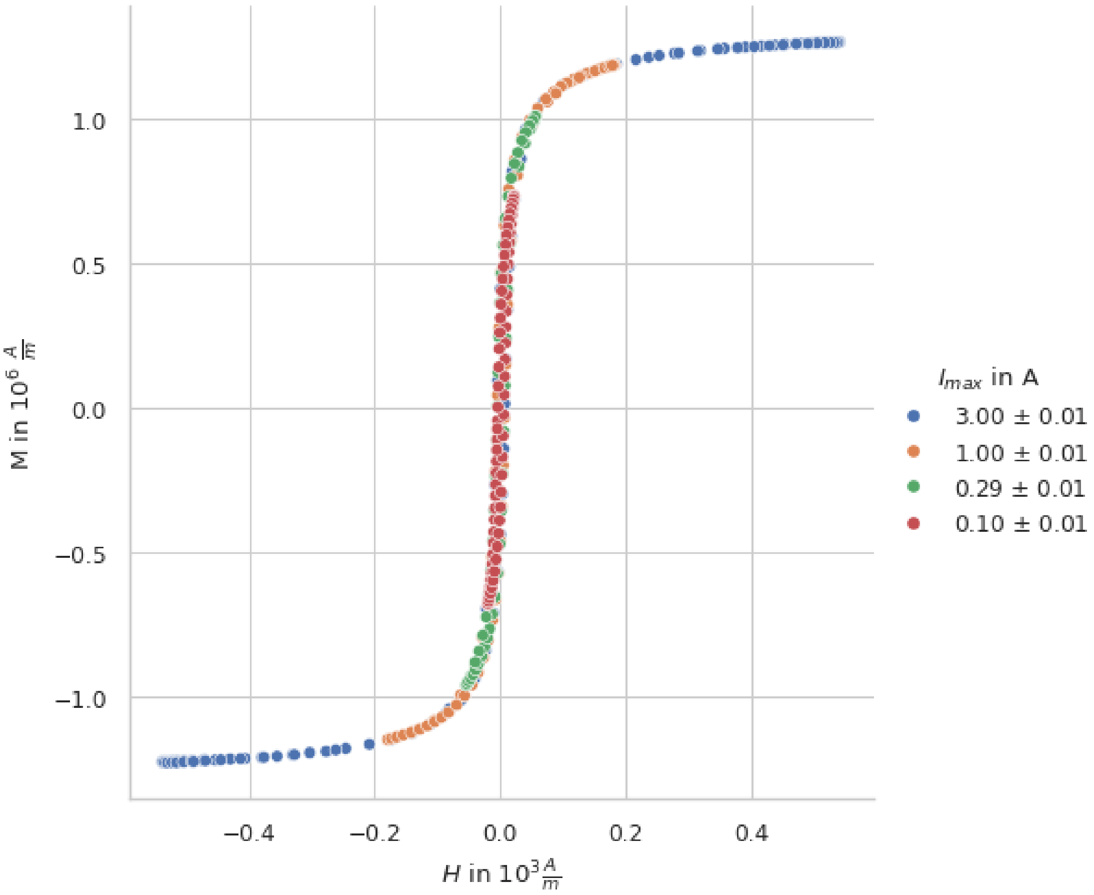
[11]:
```
def M(U):
    nu = 50 # Hz
    n_s = 17
    q = 0.9/10000 # m^2
    mu_0 = 4* math.pi * 1e-7
    return U / (47*nu*n_s*q*mu_0) / 1e6
```

[9]:
```
heizbar_a[I_column] = r'3.00 $\pm$ 0.01'
heizbar_b[I_column] = r'1.00 $\pm$ 0.01'
heizbar_c[I_column] = r'0.29 $\pm$ 0.01'
heizbar_d[I_column] = r'0.10 $\pm$ 0.01'
```

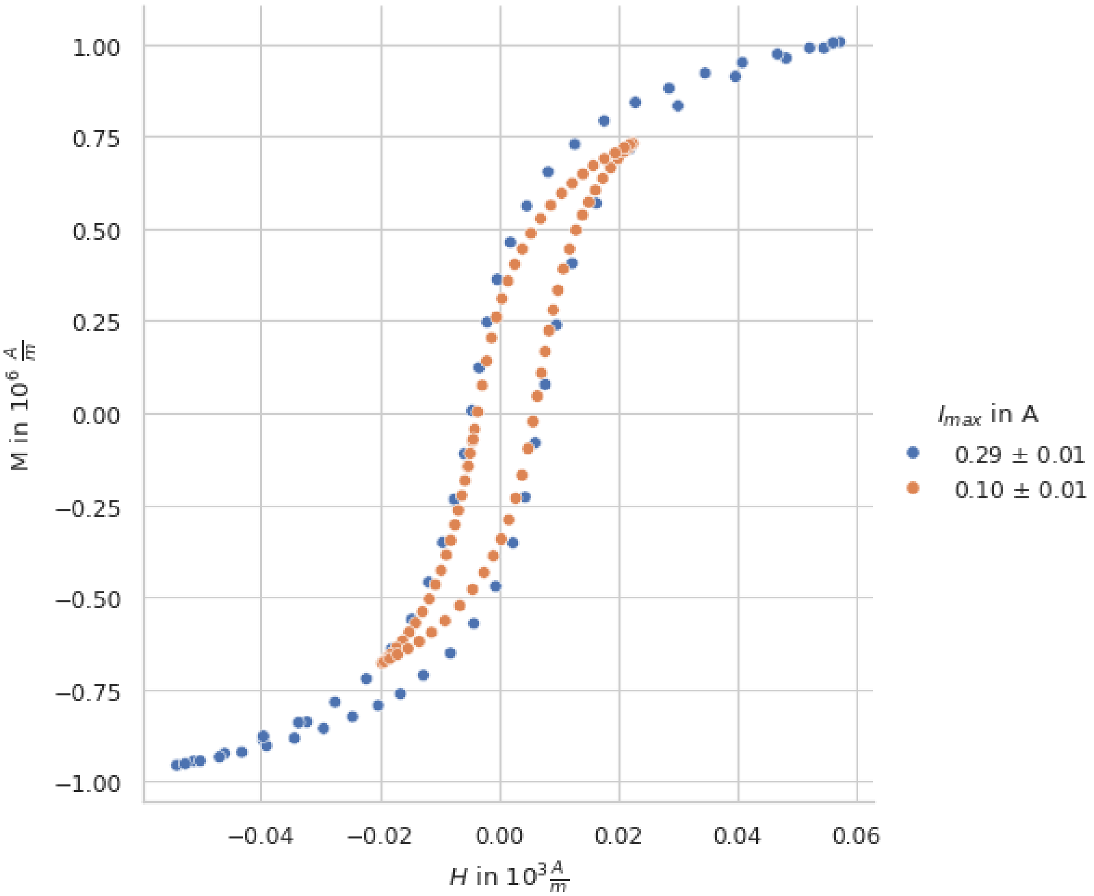
[10]:
```
heizbar_a[H_column] = heizbar_a['H'].apply(H)
heizbar_b[H_column] = heizbar_b['H'].apply(H)
heizbar_c[H_column] = heizbar_c['H'].apply(H)
heizbar_d[H_column] = heizbar_d['H'].apply(H)

heizbar_a[M_column] = heizbar_a['M'].apply(M)
heizbar_b[M_column] = heizbar_b['M'].apply(M)
heizbar_c[M_column] = heizbar_c['M'].apply(M)
heizbar_d[M_column] = heizbar_d['M'].apply(M)
```

Alle Messungen in einem Plot

[11]:
```
heizbar_all = pd.concat([heizbar_a,heizbar_b,heizbar_c,heizbar_d])
plot(heizbar_all)
```



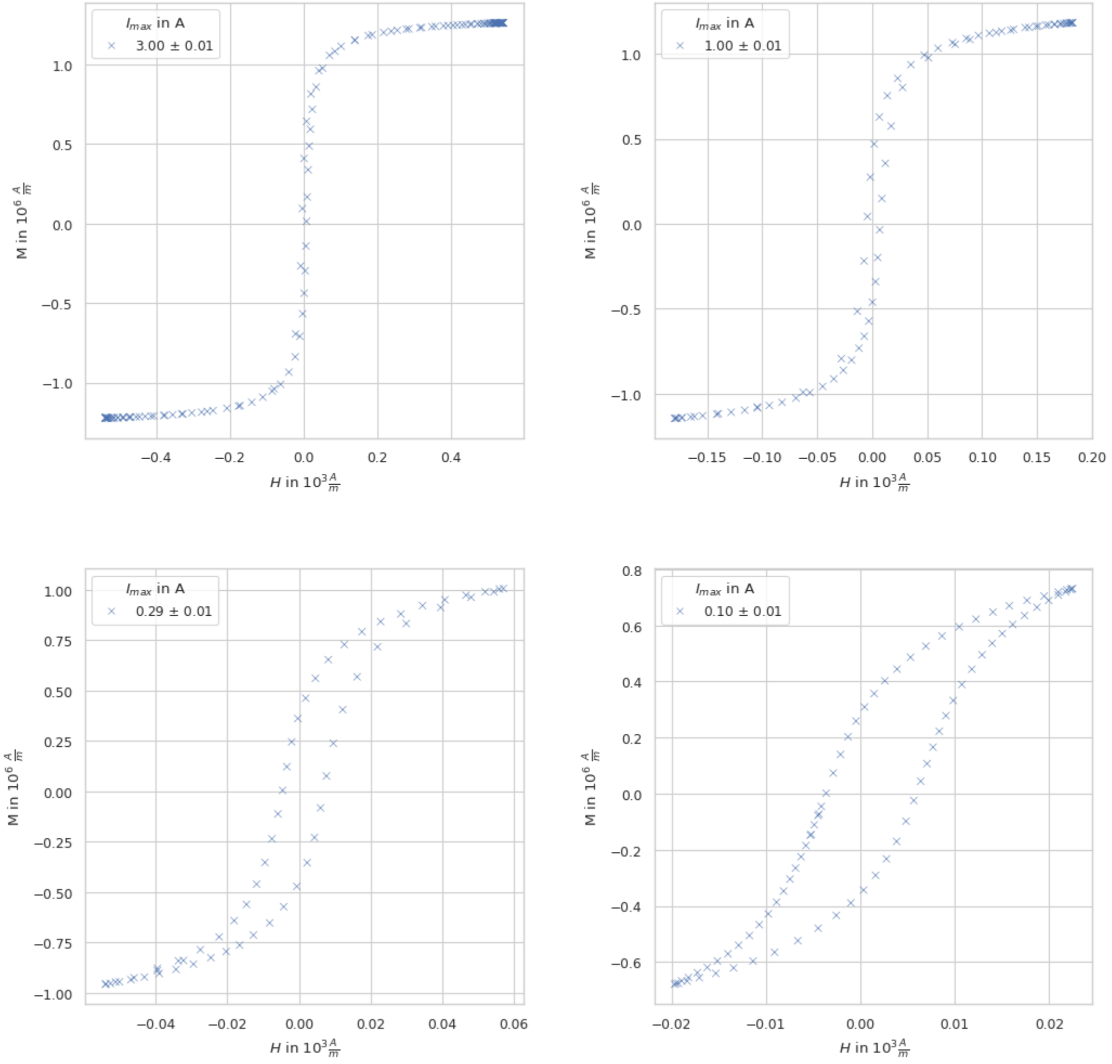
[12]:
```
plot(pd.concat([heizbar_c,heizbar_d]))
```



Alle Messungen in verschiedenen Plots

```
[13]: fig = plt.figure(figsize=(12,12))
      fig.subplots_adjust(hspace=0.3, wspace=0.3)

      # 4 subplots jeweils 1/2 Breite
      # https://matplotlib.org/stable/api/figure_api.html#matplotlib.figure.Figure.
       ↪add_subplot
      ax = fig.add_subplot(2, 2, 1)
      subplot(heizbar_a, axis=ax)

      ax = fig.add_subplot(2, 2, 2)
      subplot(heizbar_b, axis=ax)

      ax = fig.add_subplot(2, 2, 3)
      subplot(heizbar_c, axis=ax)

      ax = fig.add_subplot(2, 2, 4)
      subplot(heizbar_d, axis=ax)
```

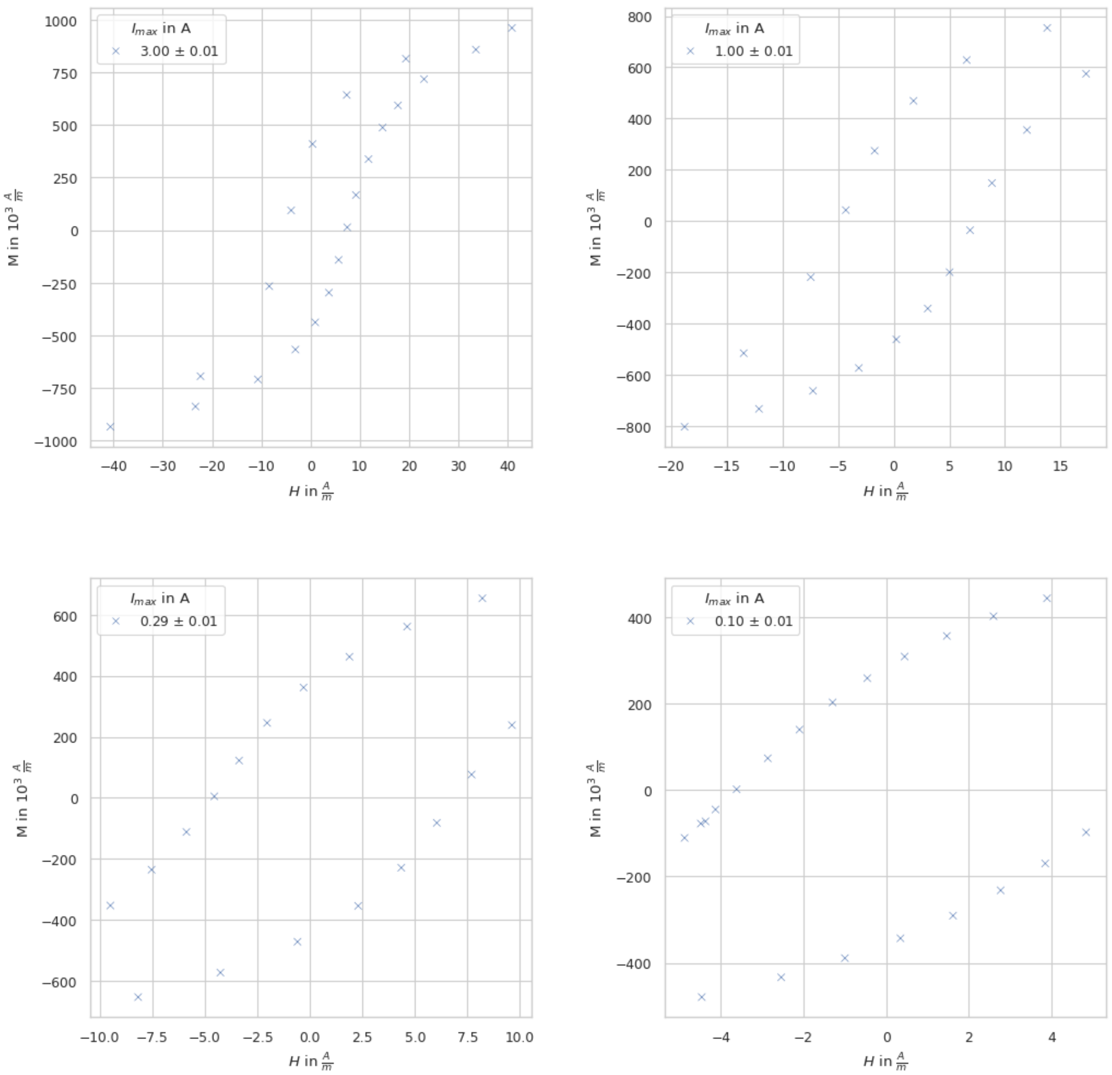
```
fig.savefig('../../media/B2.4/3.3.1_single_measures.svg', bbox_inches='tight')

plt.show()
```



[14]:
```
heizbar_a[H_column_detailed] = heizbar_a[H_column] * 1000
heizbar_b[H_column_detailed] = heizbar_b[H_column] * 1000
heizbar_c[H_column_detailed] = heizbar_c[H_column] * 1000
heizbar_d[H_column_detailed] = heizbar_d[H_column] * 1000

heizbar_a[M_column_detailed] = heizbar_a[M_column] * 1000
heizbar_b[M_column_detailed] = heizbar_b[M_column] * 1000
heizbar_c[M_column_detailed] = heizbar_c[M_column] * 1000
heizbar_d[M_column_detailed] = heizbar_d[M_column] * 1000
```

[15]:

```
fig = plt.figure(figsize=(12,12))
fig.subplots_adjust(hspace=0.3, wspace=0.3)

# 4 subplots jeweils 1/2 Breite
# https://matplotlib.org/stable/api/figure_api.html#matplotlib.figure.Figure.
 ↪add_subplot
ax = fig.add_subplot(2, 2, 1)
subplot(heizbar_a[heizbar_a[H_column].abs() < 0.05], axis=ax,␣
 ↪x_column=H_column_detailed, y_column=M_column_detailed)

ax = fig.add_subplot(2, 2, 2)
subplot(heizbar_b[heizbar_b[H_column].abs() < 0.02], axis=ax,␣
 ↪x_column=H_column_detailed, y_column=M_column_detailed)

ax = fig.add_subplot(2, 2, 3)
subplot(heizbar_c[heizbar_c[H_column].abs() < 0.01], axis=ax,␣
 ↪x_column=H_column_detailed, y_column=M_column_detailed)

ax = fig.add_subplot(2, 2, 4)
subplot(heizbar_d[heizbar_d[H_column].abs() < 0.005], axis=ax,␣
 ↪x_column=H_column_detailed, y_column=M_column_detailed)

# fig.savefig('../../media/B2.4/3.3.1_single_measures_detailed.svg',␣
 ↪bbox_inches='tight')

plt.show()
```



[16]:
```
heizbar_a['Ringkern'] = 'ohne Spalt'
```

0.1.1 ermittle Remanenz

threshold muss so gewählt werden, dass maximal 3 Werte herausgefiltert werden. Ideal wären zwei, falls ein Wert oben und ein Wert unten ist.

[17]:
```
df = heizbar_d
threshold = 0.7

df[df[H_column_detailed].abs() < threshold][M_column_detailed]
```

[17]:
```
33   -343.188088
73    309.343892
```

```
74    259.168560
Name: M in $10^3\ \frac{A}{m}$, dtype: float64
```

[18]:
```
m = df[df[H_column_detailed].abs() < threshold][M_column_detailed].abs().mean()
d = df[df[H_column_detailed].abs() < threshold][M_column_detailed].abs().std()
print(m.round(2), r'\pm', d.round(2))
```

```
303.9 \pm 42.27
```

0.1.2 ermittle H_K

threshold muss so gewählt werden, dass maximal 4 Werte herausgefiltert werden. Ideal wären zwei, falls ein Wert oben und ein Wert unten ist.

[19]:
```
df = heizbar_d
threshold = 50

df[df[M_column_detailed].abs() < threshold][[M_column_detailed,␣
 ↪H_column_detailed]]
```

[19]:
```
    M in $10^3\ \frac{A}{m}$  $H$ in $\frac{A}{m}$
0                 -44.867783             -4.128412
38                -24.091637              5.661485
39                 44.802778              6.398923
78                  1.973907             -3.617576
```

[20]:
```
m = df[df[M_column_detailed].abs() < threshold][H_column_detailed].abs().mean()
d = df[df[M_column_detailed].abs() < threshold][H_column_detailed].abs().std()
print(m.round(2), r'\pm', d.round(2))
```

```
4.95 \pm 1.3
```

0.1.3 M_{max}

[21]:
```
df = heizbar_d
m = (df[M_column_detailed].max() + abs(df[M_column_detailed].min()))/2
d = (df[M_column_detailed].max() - abs(df[M_column_detailed].min()))/2
print(m.round(2), r'\pm', d.round(2))
```

```
705.29 \pm 26.18
```

0.2 3.3.2

[22]:
```
komm_a = pd.read_csv('3.3.2.a.csv', sep='\t')
komm_b = pd.read_csv('3.3.2.b.csv', sep='\t')
```

[23]:
```
komm_a[H_column] = komm_a['H'].apply(H)
komm_b[H_column] = komm_b['H'].apply(H)
```

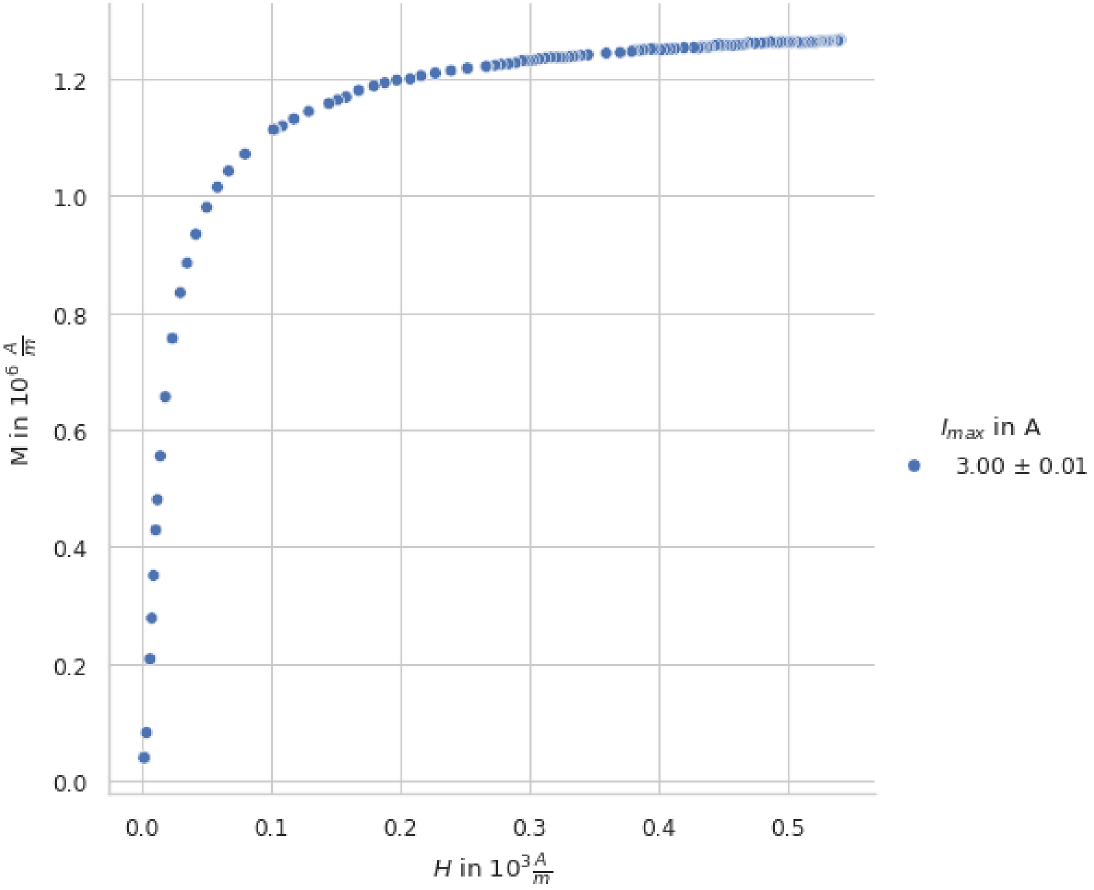
```
komm_a[M_column] = komm_a['M'].apply(M)
komm_b[M_column] = komm_b['M'].apply(M)
```

[24]:

```
komm_a[I_column] = r'3.00 $\pm$ 0.01'
komm_b[I_column] = r'0.10 $\pm$ 0.01'
```

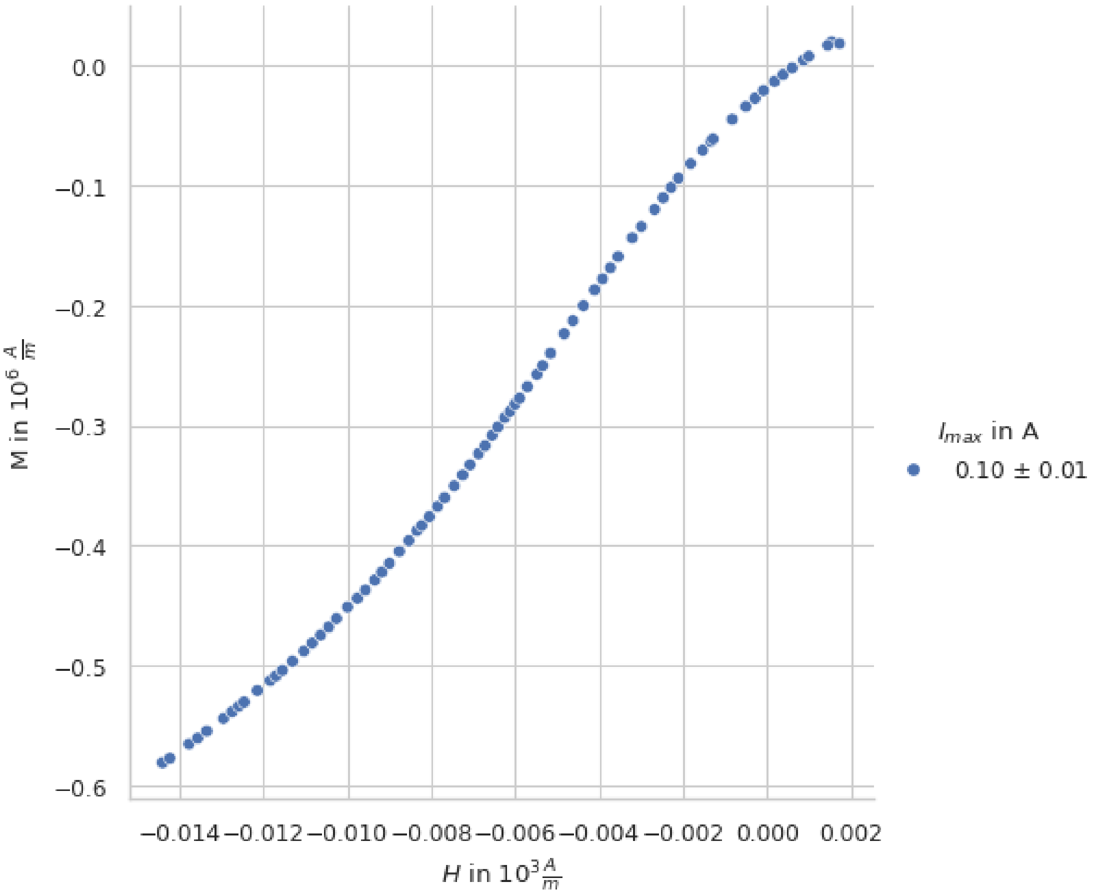
[25]:

```
plot(komm_a)
```



[26]:

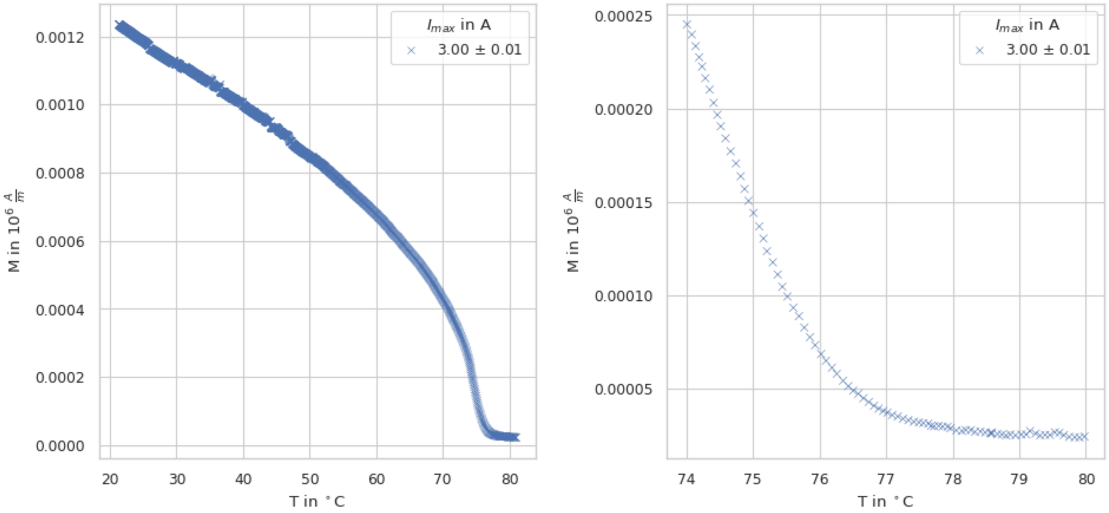
```
plot(komm_b)
```



0.3 3.3.3

[27]:
```
data = pd.read_csv('3.3.3.csv', sep='\t')
data[I_column] = r'3.00 $\pm$ 0.01'
T_column = r'T in $^\circ$C'
data[T_column] = data['T']
data[M_column] = data['M'].apply(M)
data[M_column] /= 1e3
```

[28]:
```
fig = plt.figure(figsize=(11,5))
fig.subplots_adjust(hspace=0.3, wspace=0.3)

ax = fig.add_subplot(1, 2, 1)
sns.scatterplot(
    data=data,
    x=T_column,
    y=M_column,
    hue=I_column,
    marker='x',
    legend='full',
    ax=ax
)

ax = fig.add_subplot(1, 2, 2)
sns.scatterplot(
    data=data[(data[T_column] > 74)&(data[T_column] < 80)],
    x=T_column,
    y=M_column,
    hue=I_column,
    marker='x',
    legend='full',
    ax=ax
)

fig.savefig('../../media/B2.4/3.3.3.svg', bbox_inches='tight')
```



0.4 3.3.4

Messungsdetails: * 3.4.1: 0.94A * 3.4.2: 3.0A, 1mm * 3.4.3: 2.12A, 0.5mm * 3.4.4: 1.27A, 0.2mm * 3.4.5: 1.0A, 0.125mm * 3.4.6: 0.79A, 0.075mm * 3.4.7: 0.50A, 0.0mm

[6]:

```
spalt_a = pd.read_csv('3.4.1.csv', sep='\t')
spalt_b = pd.read_csv('3.4.2.csv', sep='\t')
spalt_c = pd.read_csv('3.4.3.csv', sep='\t')
spalt_d = pd.read_csv('3.4.4.csv', sep='\t')
spalt_e = pd.read_csv('3.4.5.csv', sep='\t')
spalt_f = pd.read_csv('3.4.6.csv', sep='\t')
```

```python
spalt_g = pd.read_csv('3.4.7.csv', sep='\t')
```

Fixme: Die Länge des Spalts muss eingerechnet werden.

[7]:
```python
def H_spalt(U):
    U_max = spalt_a.H.max()
    n_p=54
    r=1.5/100 # m
    return n_p/(2 * math.pi * r) * (3.0/U_max) * U / 1e3
```

[12]:
```python
spalt_a[H_column] = spalt_a['H'].apply(H_spalt)
spalt_b[H_column] = spalt_b['H'].apply(H_spalt)
spalt_c[H_column] = spalt_c['H'].apply(H_spalt)
spalt_d[H_column] = spalt_d['H'].apply(H_spalt)
spalt_e[H_column] = spalt_e['H'].apply(H_spalt)
spalt_f[H_column] = spalt_f['H'].apply(H_spalt)
spalt_g[H_column] = spalt_g['H'].apply(H_spalt)

spalt_a[M_column] = spalt_a['M'].apply(M)
spalt_b[M_column] = spalt_b['M'].apply(M)
spalt_c[M_column] = spalt_c['M'].apply(M)
spalt_d[M_column] = spalt_d['M'].apply(M)
spalt_e[M_column] = spalt_e['M'].apply(M)
spalt_f[M_column] = spalt_f['M'].apply(M)
spalt_g[M_column] = spalt_g['M'].apply(M)
```

[13]:
```python
spalt_a['Ringkern'] = 'mit Spalt'

S_column = 'Spaltbreite'
spalt_b[S_column] = r'2.00 mm'
spalt_c[S_column] = r'1.00 mm'
spalt_d[S_column] = r'0.40 mm'
spalt_e[S_column] = r'0.25 mm'
spalt_f[S_column] = r'0.15 mm'
spalt_g[S_column] = r'0.00 mm'
```

[14]:
```python
def plot(data, hue_column=I_column, filename=None):
    img = sns.relplot(
        data=data,
        x=H_column,
        y=M_column,
        hue=hue_column,
        height=5,
        legend='full',

    )
    if filename is not None:
        img.figure.savefig(filename, bbox_inches='tight')
```

```
plot(pd.concat([heizbar_a, spalt_a]), hue_column='Ringkern') #, filename='../../
 ↪media/B2.4/3.3.3_comparison.svg')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 14
     11     if filename is not None:
     12         img.figure.savefig(filename, bbox_inches='tight')
---> 14 plot(pd.concat([heizbar_a, spalt_a]), hue_column='Ringkern') #,␣
 ↪filename='../../media/B2.4/3.3.3_comparison.svg')

NameError: name 'heizbar_a' is not defined
```

[ ]:
```
spalt_a[H_column_detailed] = spalt_a[H_column] * 1000
spalt_a[M_column_detailed] = spalt_a[M_column] * 1000
spalt_a[I_column] = r'0.94 \pm 0.01 A'
```

[9]:
```
subplot(spalt_a[spalt_a[H_column_detailed].abs() < 200],␣
 ↪x_column=H_column_detailed, y_column=M_column_detailed);
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~/.local/lib/python3.12/site-packages/pandas/core/indexes/base.py:3790, in
 ↪Index.get_loc(self, key)
   3789 try:
-> 3790     return self._engine.get_loc(casted_key)
   3791 except KeyError as err:

File index.pyx:152, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:181, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7088, in pandas._libs.hashtable.
 ↪PyObjectHashTable.get_item()

KeyError: '$H$ in $\\frac{A}{m}$'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[9], line 1
```

```
----> 1 subplot(spalt_a[spalt_a[H_column_detailed].abs() < 200], x_column=H_column_detailed, y_column=M_column_detailed);

File ~/.local/lib/python3.12/site-packages/pandas/core/frame.py:3893, in DataFrame.__getitem__(self, key)
   3891 if self.columns.nlevels > 1:
   3892     return self._getitem_multilevel(key)
-> 3893 indexer = self.columns.get_loc(key)
   3894 if is_integer(indexer):
   3895     indexer = [indexer]

File ~/.local/lib/python3.12/site-packages/pandas/core/indexes/base.py:3797, in Index.get_loc(self, key)
   3792     if isinstance(casted_key, slice) or (
   3793         isinstance(casted_key, abc.Iterable)
   3794         and any(isinstance(x, slice) for x in casted_key)
   3795     ):
   3796         raise InvalidIndexError(key)
-> 3797     raise KeyError(key) from err
   3798 except TypeError:
   3799     # If we have a listlike key, _check_indexing_error will raise
   3800     #  InvalidIndexError. Otherwise we fall through and re-raise
   3801     #  the TypeError.
   3802     self._check_indexing_error(key)

KeyError: '$H$ in $\\frac{A}{m}$'
```

0.4.1 ermittle Remanenz

threshold muss so gewählt werden, dass maximal 3 Werte herausgefiltert werden. Ideal wären zwei, falls ein Wert oben und ein Wert unten ist.

[37]:

```
df = spalt_a
threshold = 100

df[df[H_column_detailed].abs() < threshold][M_column_detailed
]
```

[37]:

```
21   -216.736368
43    237.407872
44    142.691131
Name: M in $10^3\ \frac{A}{m}$, dtype: float64
```

[38]:

```
m = df[df[H_column_detailed].abs() < threshold][M_column_detailed].abs().mean()
d = df[df[H_column_detailed].abs() < threshold][M_column_detailed].abs().std()
print(m.round(2), r'\pm', d.round(2))
```

```
198.95 \pm 49.8
```

0.4.2 ermittle H_K

`threshold` muss so gewählt werden, dass maximal 4 Werte herausgefiltert werden. Ideal wären zwei, falls ein Wert oben und ein Wert unten ist.

```
[39]: df = spalt_a
      threshold = 50

      df[df[M_column_detailed].abs() < threshold][[M_column_detailed,␣
       ↪H_column_detailed]]
```

```
[39]:     M in $10^3\ \frac{A}{m}$  $H$ in $\frac{A}{m}$
      1                   -6.707348          -240.342553
      23                  12.799964           244.102602
      45                  31.351007          -199.387561
```

```
[40]: m = df[df[M_column_detailed].abs() < threshold][H_column_detailed].abs().mean()
      d = df[df[M_column_detailed].abs() < threshold][H_column_detailed].abs().std()
      print(m.round(2), r'\pm', d.round(2))
```

```
227.94 \pm 24.8
```

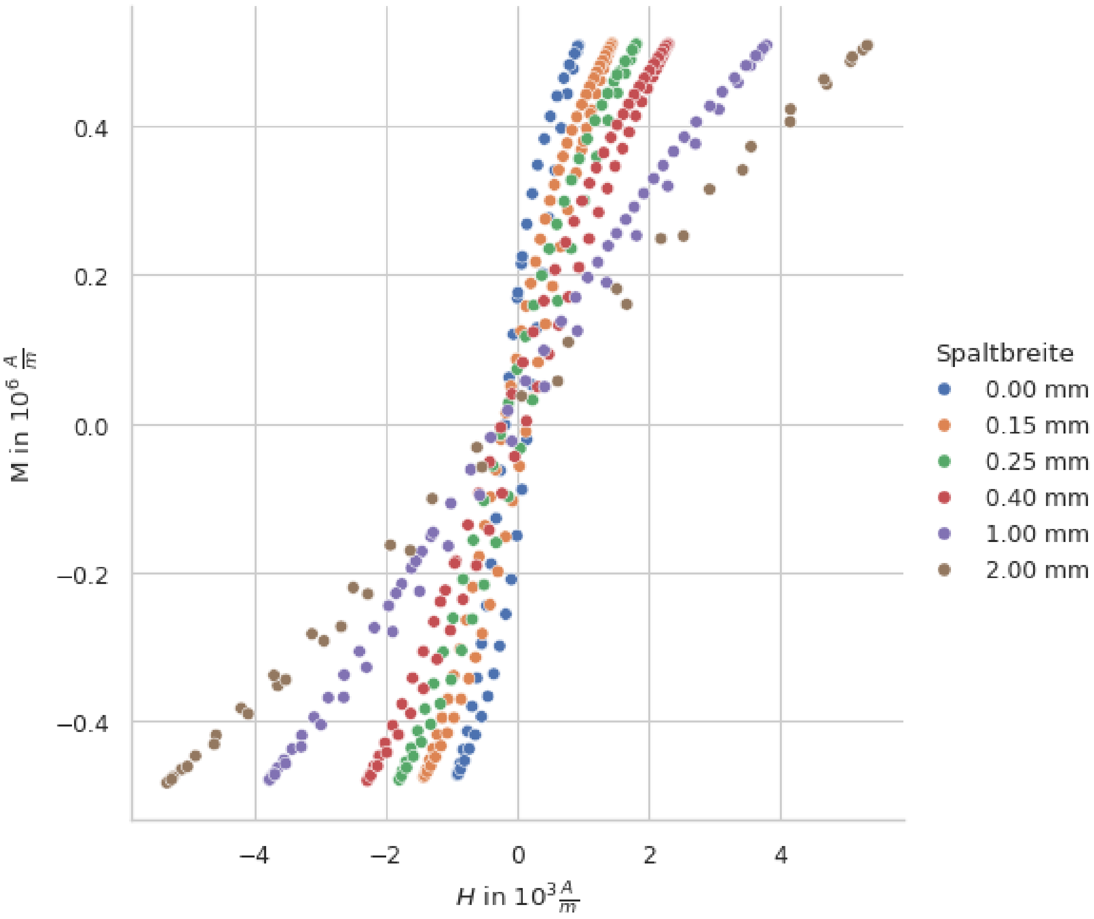
0.4.3 M_{max}

```
[41]: df = spalt_a
      m = (df[M_column_detailed].max() + abs(df[M_column_detailed].min()))/2
      d = (df[M_column_detailed].max() - abs(df[M_column_detailed].min()))/2
      print(m.round(2), r'\pm', d.round(2))
```

```
675.28 \pm 14.92
```

0.4.4 Entmagnetisierungsfaktor

```
[15]: spalt_all = pd.concat([spalt_g,spalt_f,spalt_e,spalt_d,spalt_c,spalt_b])
      plot(spalt_all, hue_column=S_column, filename='../../media/B2.4/3.3.3_overview.
       ↪svg')
```



Entmagnetisierungsfelder

```
[16]: df = spalt_f

avg = (df[H_column].max() - spalt_g[H_column].max() - (df[H_column].min() -␣
 ↪spalt_g[H_column].min()))/2
err = abs((df[H_column].max() - spalt_g[H_column].max() + (df[H_column].min() -␣
 ↪spalt_g[H_column].min()))/2)
print(df['Spaltbreite'][0])
print(avg.round(3), r'\pm', err.round(3))
```

```
0.15 mm
0.508 \pm 0.001
```

0.4.5 M_{max}

[25]:

```
df = spalt_g
m = (df[M_column].max() + abs(df[M_column].min()))/2
d = (df[M_column].max() - abs(df[M_column].min()))/2
print(df['Spaltbreite'][0])
print(m.round(2), r'\pm', d.round(2))
```

```
0.00 mm
0.49 \pm 0.02
```

0.4.6 N experimentell

[58]:

```
def N(delta_H):
    M_max = 0.495 * 1e3
    err_M = 0.025 * 1e3

    avg = delta_H / M_max
    err = delta_H / err_M
    print(round(avg, 3), r'\pm', round(err, 3))
```

[64]:

```
N(0)
```

```
0.0 \pm 0.0
```

0.4.7 N theoretisch

[80]:

```
def N_theo(l_L):
    R = 150 # mm
    return round(l_L / (2*math.pi*R + l_L) * 1e3, 2)
```

[86]:

```
N_theo(2)
```

[86]:

```
2.12
```