Auswertung

December 3, 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
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import linregress
```

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

[3]:
```
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
```

[4]:
```
plt.rcParams.keys()
```

[4]:
```
KeysView(RcParams({'_internal.classic_mode': False,
          'agg.path.chunksize': 0,
          'animation.bitrate': -1,
          'animation.codec': 'h264',
          'animation.convert_args': ['-layers', 'OptimizePlus'],
          'animation.convert_path': 'convert',
          'animation.embed_limit': 20.0,
          'animation.ffmpeg_args': [],
          'animation.ffmpeg_path': 'ffmpeg',
          'animation.frame_format': 'png',
          'animation.html': 'none',
          'animation.writer': 'ffmpeg',
          'axes.autolimit_mode': 'data',
          'axes.axisbelow': True,
          'axes.edgecolor': '.8',
```

```
          'axes.facecolor': 'white',
          'axes.formatter.limits': [-5, 6],
          'axes.formatter.min_exponent': 0,
          'axes.formatter.offset_threshold': 4,
          'axes.formatter.use_locale': False,
          'axes.formatter.use_mathtext': False,
          'axes.formatter.useoffset': True,
          'axes.grid': True,
          'axes.grid.axis': 'both',
          'axes.grid.which': 'major',
          'axes.labelcolor': '.15',
          'axes.labelpad': 4.0,
          'axes.labelsize': 13.0,
          'axes.labelweight': 'normal',
          'axes.linewidth': 1.0,
          'axes.prop_cycle': cycler('color', [(0.2980392156862745,
0.4470588235294118, 0.6901960784313725), (0.8666666666666667,
0.5176470588235295, 0.3215686274509804), (0.3333333333333333,
0.6588235294117647, 0.40784313725490196), (0.7686274509803922,
0.3058823529411765, 0.3215686274509804), (0.5058823529411764,
0.4470588235294118, 0.7019607843137254), (0.5764705882352941,
0.47058823529411764, 0.3764705882352941), (0.8549019607843137,
0.5450980392156862, 0.7647058823529411), (0.5490196078431373,
0.5490196078431373, 0.5490196078431373), (0.8, 0.7254901960784313,
0.4549019607843137), (0.39215686274509803, 0.7098039215686275,
0.803921568627451)]),
          'axes.spines.bottom': True,
          'axes.spines.left': True,
          'axes.spines.right': True,
          'axes.spines.top': True,
          'axes.titlecolor': 'auto',
          'axes.titlelocation': 'center',
          'axes.titlepad': 6.0,
          'axes.titlesize': 13.0,
          'axes.titleweight': 'normal',
          'axes.titley': None,
          'axes.unicode_minus': True,
          'axes.xmargin': 0.05,
          'axes.ymargin': 0.05,
          'axes.zmargin': 0.05,
          'axes3d.grid': True,
          'axes3d.xaxis.panecolor': (0.95, 0.95, 0.95, 0.5),
          'axes3d.yaxis.panecolor': (0.9, 0.9, 0.9, 0.5),
          'axes3d.zaxis.panecolor': (0.925, 0.925, 0.925, 0.5),
          'backend': 'module://matplotlib_inline.backend_inline',
          'backend_fallback': True,
          'boxplot.bootstrap': None,
```

```
'boxplot.boxprops.color': 'black',
'boxplot.boxprops.linestyle': '-',
'boxplot.boxprops.linewidth': 1.0,
'boxplot.capprops.color': 'black',
'boxplot.capprops.linestyle': '-',
'boxplot.capprops.linewidth': 1.0,
'boxplot.flierprops.color': 'black',
'boxplot.flierprops.linestyle': 'none',
'boxplot.flierprops.linewidth': 1.0,
'boxplot.flierprops.marker': 'o',
'boxplot.flierprops.markeredgecolor': 'black',
'boxplot.flierprops.markeredgewidth': 1.0,
'boxplot.flierprops.markerfacecolor': 'none',
'boxplot.flierprops.markersize': 6.0,
'boxplot.meanline': False,
'boxplot.meanprops.color': 'C2',
'boxplot.meanprops.linestyle': '--',
'boxplot.meanprops.linewidth': 1.0,
'boxplot.meanprops.marker': '^',
'boxplot.meanprops.markeredgecolor': 'C2',
'boxplot.meanprops.markerfacecolor': 'C2',
'boxplot.meanprops.markersize': 6.0,
'boxplot.medianprops.color': 'C1',
'boxplot.medianprops.linestyle': '-',
'boxplot.medianprops.linewidth': 1.0,
'boxplot.notch': False,
'boxplot.patchartist': False,
'boxplot.showbox': True,
'boxplot.showcaps': True,
'boxplot.showfliers': True,
'boxplot.showmeans': False,
'boxplot.vertical': True,
'boxplot.whiskerprops.color': 'black',
'boxplot.whiskerprops.linestyle': '-',
'boxplot.whiskerprops.linewidth': 1.0,
'boxplot.whiskers': 1.5,
'contour.algorithm': 'mpl2014',
'contour.corner_mask': True,
'contour.linewidth': None,
'contour.negative_linestyle': 'dashed',
'date.autoformatter.day': '%Y-%m-%d',
'date.autoformatter.hour': '%m-%d %H',
'date.autoformatter.microsecond': '%M:%S.%f',
'date.autoformatter.minute': '%d %H:%M',
'date.autoformatter.month': '%Y-%m',
'date.autoformatter.second': '%H:%M:%S',
'date.autoformatter.year': '%Y',
```

```
'date.converter': 'auto',
'date.epoch': '1970-01-01T00:00:00',
'date.interval_multiples': True,
'docstring.hardcopy': False,
'errorbar.capsize': 0.0,
'figure.autolayout': False,
'figure.constrained_layout.h_pad': 0.04167,
'figure.constrained_layout.hspace': 0.02,
'figure.constrained_layout.use': False,
'figure.constrained_layout.w_pad': 0.04167,
'figure.constrained_layout.wspace': 0.02,
'figure.dpi': 100.0,
'figure.edgecolor': 'white',
'figure.facecolor': 'white',
'figure.figsize': [6.4, 4.8],
'figure.frameon': True,
'figure.hooks': [],
'figure.labelsize': 'large',
'figure.labelweight': 'normal',
'figure.max_open_warning': 20,
'figure.raise_window': True,
'figure.subplot.bottom': 0.11,
'figure.subplot.hspace': 0.2,
'figure.subplot.left': 0.125,
'figure.subplot.right': 0.9,
'figure.subplot.top': 0.88,
'figure.subplot.wspace': 0.2,
'figure.titlesize': 'large',
'figure.titleweight': 'normal',
'font.cursive': ['Apple Chancery',
                 'Textile',
                 'Zapf Chancery',
                 'Sand',
                 'Script MT',
                 'Felipa',
                 'Comic Neue',
                 'Comic Sans MS',
                 'cursive'],
'font.family': ['sans-serif'],
'font.fantasy': ['Chicago',
                 'Charcoal',
                 'Impact',
                 'Western',
                 'xkcd script',
                 'fantasy'],
'font.monospace': ['DejaVu Sans Mono',
                   'Bitstream Vera Sans Mono',
```

```
                   'Computer Modern Typewriter',
                   'Andale Mono',
                   'Nimbus Mono L',
                   'Courier New',
                   'Courier',
                   'Fixed',
                   'Terminal',
                   'monospace'],
'font.sans-serif': ['Arial',
                    'DejaVu Sans',
                    'Liberation Sans',
                    'Bitstream Vera Sans',
                    'sans-serif'],
'font.serif': ['DejaVu Serif',
               'Bitstream Vera Serif',
               'Computer Modern Roman',
               'New Century Schoolbook',
               'Century Schoolbook L',
               'Utopia',
               'ITC Bookman',
               'Bookman',
               'Nimbus Roman No9 L',
               'Times New Roman',
               'Times',
               'Palatino',
               'Charter',
               'serif'],
'font.size': 9.600000000000001,
'font.stretch': 'normal',
'font.style': 'normal',
'font.variant': 'normal',
'font.weight': 'normal',
'grid.alpha': 1.0,
'grid.color': '.8',
'grid.linestyle': '-',
'grid.linewidth': 0.8,
'hatch.color': 'black',
'hatch.linewidth': 1.0,
'hist.bins': 10,
'image.aspect': 'equal',
'image.cmap': 'rocket',
'image.composite_image': True,
'image.interpolation': 'antialiased',
'image.lut': 256,
'image.origin': 'upper',
'image.resample': True,
'interactive': False,
```

```
'keymap.back': ['left', 'c', 'backspace', 'MouseButton.BACK'],
'keymap.copy': ['ctrl+c', 'cmd+c'],
'keymap.forward': ['right', 'v', 'MouseButton.FORWARD'],
'keymap.fullscreen': ['f', 'ctrl+f'],
'keymap.grid': ['g'],
'keymap.grid_minor': ['G'],
'keymap.help': ['f1'],
'keymap.home': ['h', 'r', 'home'],
'keymap.pan': ['p'],
'keymap.quit': ['ctrl+w', 'cmd+w', 'q'],
'keymap.quit_all': [],
'keymap.save': ['s', 'ctrl+s'],
'keymap.xscale': ['k', 'L'],
'keymap.yscale': ['l'],
'keymap.zoom': ['o'],
'legend.borderaxespad': 0.5,
'legend.borderpad': 0.4,
'legend.columnspacing': 2.0,
'legend.edgecolor': '0.8',
'legend.facecolor': 'inherit',
'legend.fancybox': True,
'legend.fontsize': 11.0,
'legend.framealpha': 0.8,
'legend.frameon': True,
'legend.handleheight': 0.7,
'legend.handlelength': 2.0,
'legend.handletextpad': 0.8,
'legend.labelcolor': 'None',
'legend.labelspacing': 0.5,
'legend.loc': 'best',
'legend.markerscale': 1.0,
'legend.numpoints': 1,
'legend.scatterpoints': 1,
'legend.shadow': False,
'legend.title_fontsize': 11.0,
'lines.antialiased': True,
'lines.color': 'C0',
'lines.dash_capstyle': <CapStyle.butt: 'butt'>,
'lines.dash_joinstyle': <JoinStyle.round: 'round'>,
'lines.dashdot_pattern': [6.4, 1.6, 1.0, 1.6],
'lines.dashed_pattern': [3.7, 1.6],
'lines.dotted_pattern': [1.0, 1.65],
'lines.linestyle': '-',
'lines.linewidth': 1.2000000000000002,
'lines.marker': 'None',
'lines.markeredgecolor': 'auto',
'lines.markeredgewidth': 1.0,
```

```
'lines.markerfacecolor': 'auto',
'lines.markersize': 4.800000000000001,
'lines.scale_dashes': True,
'lines.solid_capstyle': <CapStyle.round: 'round'>,
'lines.solid_joinstyle': <JoinStyle.round: 'round'>,
'macosx.window_mode': 'system',
'markers.fillstyle': 'full',
'mathtext.bf': 'sans:bold',
'mathtext.bfit': 'sans:italic:bold',
'mathtext.cal': 'cursive',
'mathtext.default': 'it',
'mathtext.fallback': 'cm',
'mathtext.fontset': 'dejavusans',
'mathtext.it': 'sans:italic',
'mathtext.rm': 'sans',
'mathtext.sf': 'sans',
'mathtext.tt': 'monospace',
'patch.antialiased': True,
'patch.edgecolor': 'w',
'patch.facecolor': 'C0',
'patch.force_edgecolor': True,
'patch.linewidth': 0.8,
'path.effects': [],
'path.simplify': True,
'path.simplify_threshold': 0.111111111111,
'path.sketch': None,
'path.snap': True,
'pcolor.shading': 'auto',
'pcolormesh.snap': True,
'pdf.compression': 6,
'pdf.fonttype': 3,
'pdf.inheritcolor': False,
'pdf.use14corefonts': False,
'pgf.preamble': '',
'pgf.rcfonts': True,
'pgf.texsystem': 'xelatex',
'polaraxes.grid': True,
'ps.distiller.res': 6000,
'ps.fonttype': 3,
'ps.papersize': 'letter',
'ps.useafm': False,
'ps.usedistiller': None,
'savefig.bbox': None,
'savefig.directory': '~',
'savefig.dpi': 'figure',
'savefig.edgecolor': 'auto',
'savefig.facecolor': 'auto',
```

```
'savefig.format': 'png',
'savefig.orientation': 'portrait',
'savefig.pad_inches': 0.1,
'savefig.transparent': False,
'scatter.edgecolors': 'face',
'scatter.marker': 'o',
'svg.fonttype': 'path',
'svg.hashsalt': None,
'svg.image_inline': True,
'text.antialiased': True,
'text.color': '.15',
'text.hinting': 'force_autohint',
'text.hinting_factor': 8,
'text.kerning_factor': 0,
'text.latex.preamble': '',
'text.parse_math': True,
'text.usetex': False,
'timezone': 'UTC',
'tk.window_focus': False,
'toolbar': 'toolbar2',
'webagg.address': '127.0.0.1',
'webagg.open_in_browser': True,
'webagg.port': 8988,
'webagg.port_retries': 50,
'xaxis.labellocation': 'center',
'xtick.alignment': 'center',
'xtick.bottom': False,
'xtick.color': '.15',
'xtick.direction': 'out',
'xtick.labelbottom': True,
'xtick.labelcolor': 'inherit',
'xtick.labelsize': 11.0,
'xtick.labeltop': False,
'xtick.major.bottom': True,
'xtick.major.pad': 3.5,
'xtick.major.size': 4.800000000000001,
'xtick.major.top': True,
'xtick.major.width': 1.0,
'xtick.minor.bottom': True,
'xtick.minor.ndivs': 'auto',
'xtick.minor.pad': 3.4,
'xtick.minor.size': 3.2,
'xtick.minor.top': True,
'xtick.minor.visible': False,
'xtick.minor.width': 0.8,
'xtick.top': False,
'yaxis.labellocation': 'center',
```

```
 'ytick.alignment': 'center_baseline',
 'ytick.color': '.15',
 'ytick.direction': 'out',
 'ytick.labelcolor': 'inherit',
 'ytick.labelleft': True,
 'ytick.labelright': False,
 'ytick.labelsize': 11.0,
 'ytick.left': False,
 'ytick.major.left': True,
 'ytick.major.pad': 3.5,
 'ytick.major.right': True,
 'ytick.major.size': 4.800000000000001,
 'ytick.major.width': 1.0,
 'ytick.minor.left': True,
 'ytick.minor.ndivs': 'auto',
 'ytick.minor.pad': 3.4,
 'ytick.minor.right': True,
 'ytick.minor.size': 3.2,
 'ytick.minor.visible': False,
 'ytick.minor.width': 0.8,
 'ytick.right': False}))
```

[5]:
```
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 $10^3\ \frac{A}{m}$'
```

[6]:
```
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')
```

[7]:
```
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

Overview

[8]:
```
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')
```

[9]:
```
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
```

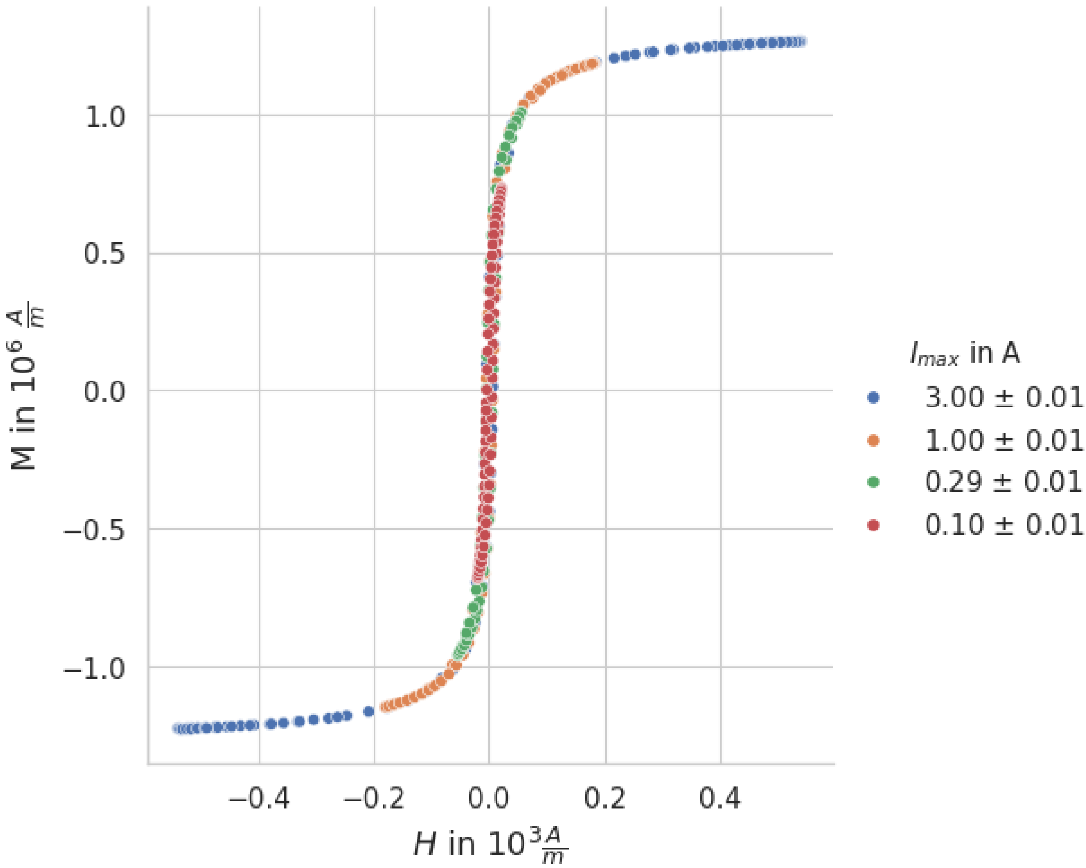
[10]:
```
def M(U):
    nu = 50 # Hz
    n_s = 17
    q = 0.9/10000 # m^2
    mu_0 = 4* math.pi * 1e-7
    return U / (47*nu*n_s*q*mu_0) / 1e6
```

[11]:
```
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'
```

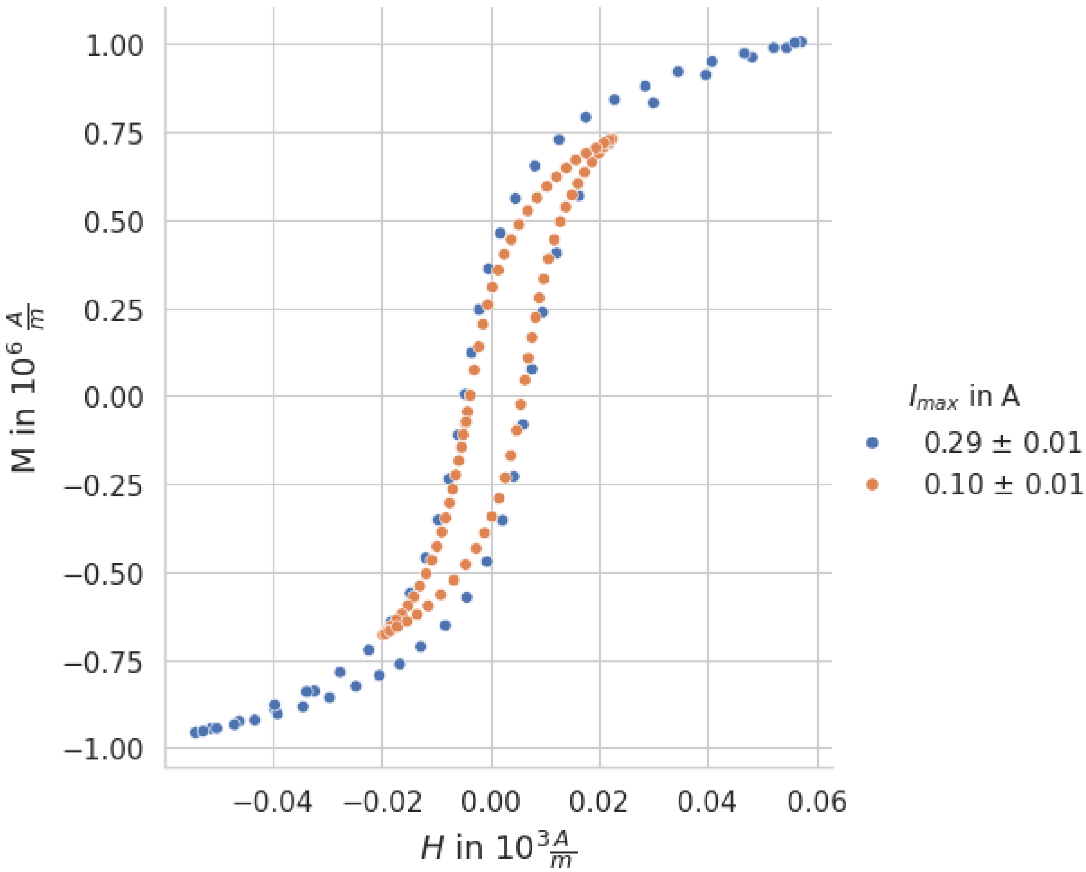
[12]:
```
heizbar_a[H_column] = heizbar_a['H'].apply(H)
heizbar_b[H_column] = heizbar_b['H'].apply(H)
heizbar_c[H_column] = heizbar_c['H'].apply(H)
heizbar_d[H_column] = heizbar_d['H'].apply(H)

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

[13]:
```
heizbar_all = pd.concat([heizbar_a,heizbar_b,heizbar_c,heizbar_d])
plot(heizbar_all)
```



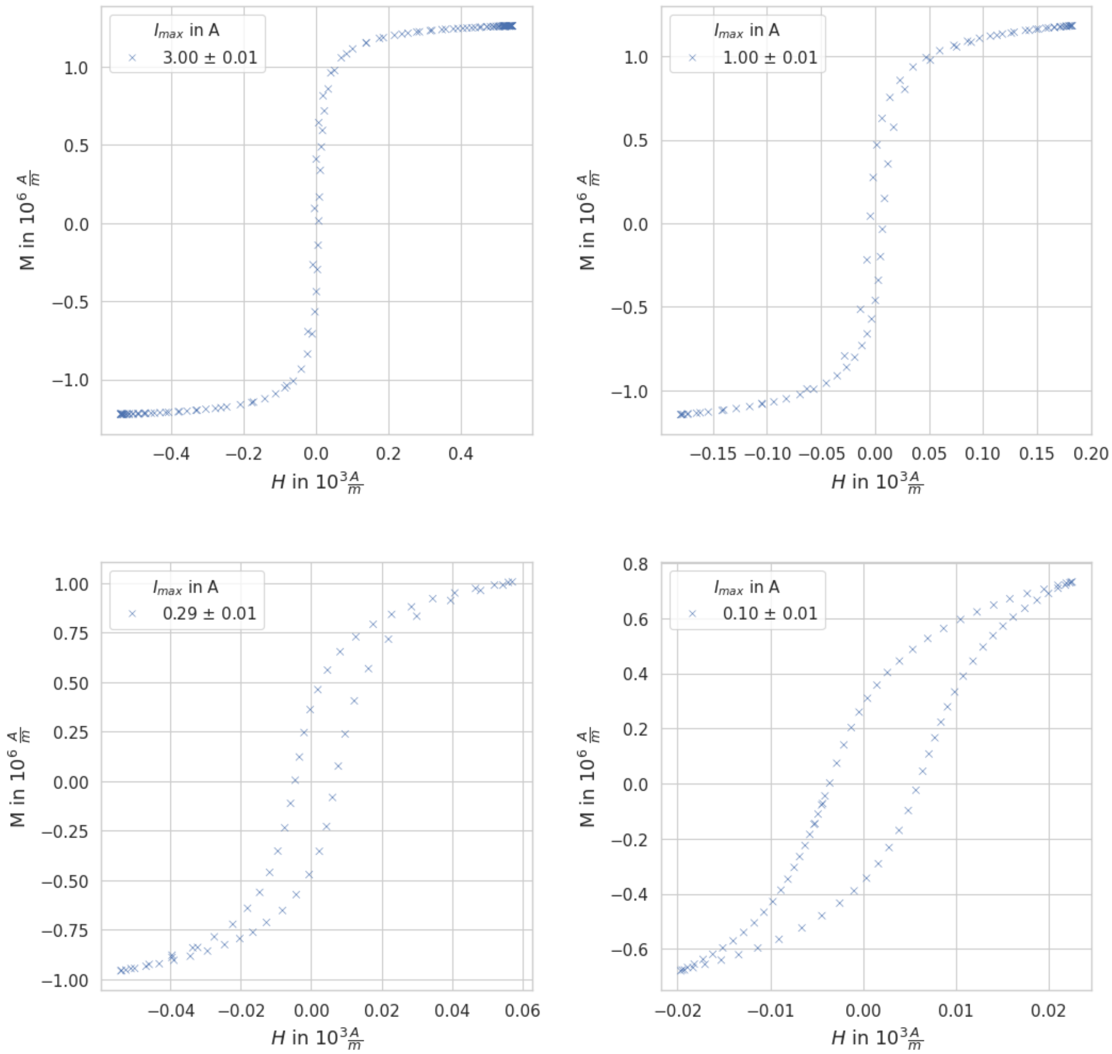
[14]:
```
plot(pd.concat([heizbar_c,heizbar_d]))
```



Alle Messungen in verschiedenen Plots

[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, 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()
```

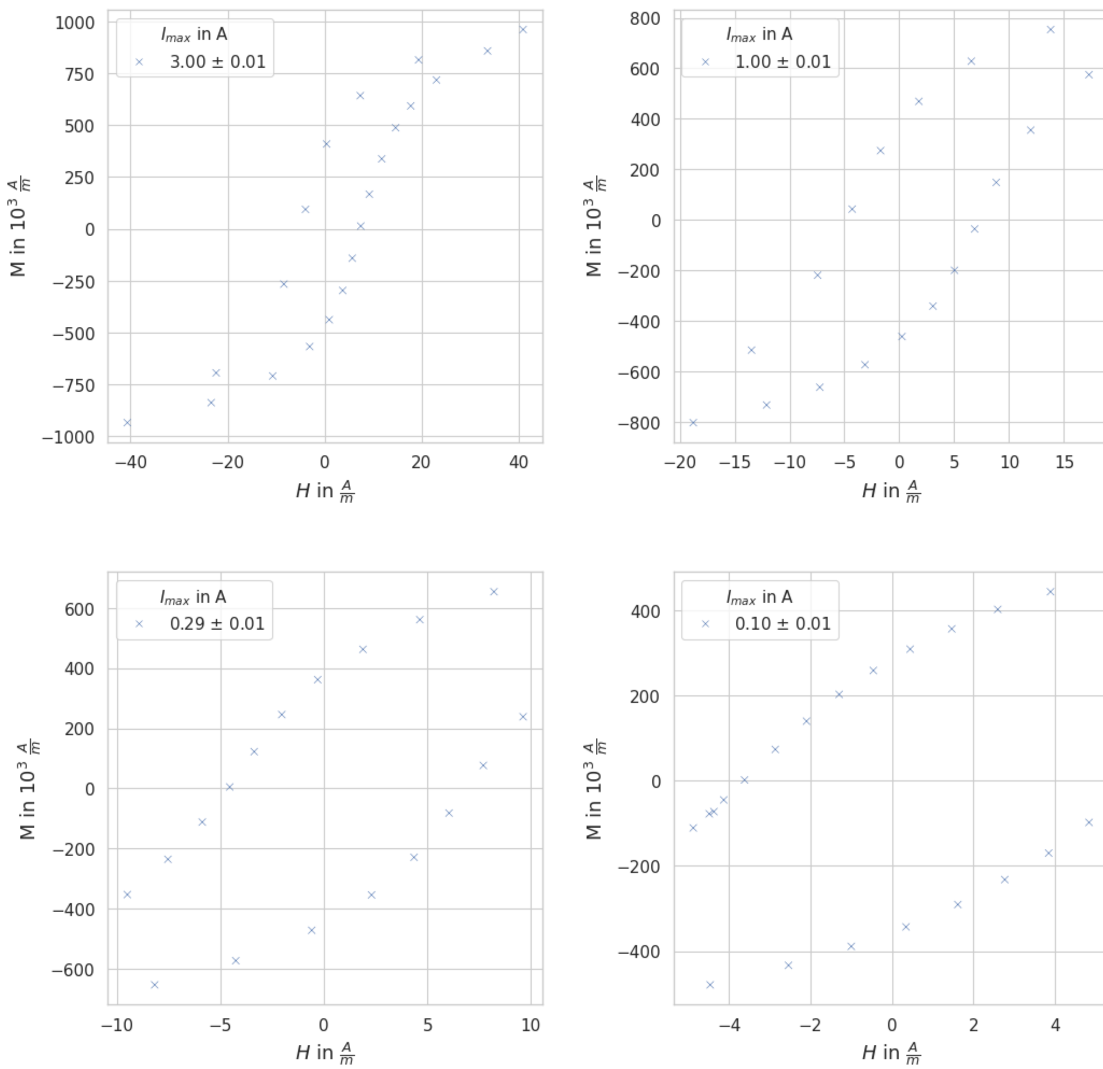


details & values

[16]:
```
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
```

[17]:
```
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()
```



[18]:
```
heizbar_a['Ringkern'] = 'ohne Spalt'
```

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.

[19]:
```
df = heizbar_d
threshold = 0.7

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

[19]:
```
33   -343.188088
73    309.343892
74    259.168560
Name: M in $10^3\ \frac{A}{m}$, dtype: float64
```

[20]:
```
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
```

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.

[21]:
```
df = heizbar_d
threshold = 50

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

[21]:

	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

[22]:
```
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
```

$M_{\max}$

[23]:
```
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

[24]:
```
komm_a = pd.read_csv('3.3.2.a.csv', sep='\t')
komm_b = pd.read_csv('3.3.2.b.csv', sep='\t')
```

[25]:
```
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)
```

[26]:
```
komm_a[I_column] = r'3.00 $\pm$ 0.01'
komm_b[I_column] = r'0.10 $\pm$ 0.01'
```

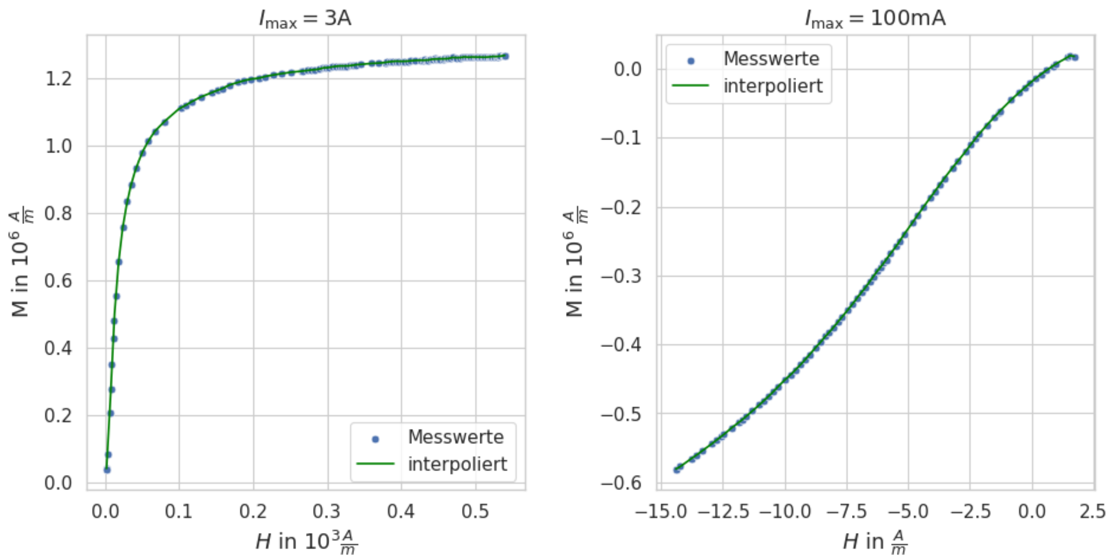
[27]:
```
komm_a = komm_a.sort_values(by=H_column)
komm_b = komm_b.sort_values(by=H_column)
```

[28]:
```
x_range_a = np.linspace(komm_a[H_column].min(), komm_a[H_column].max(), 100)
interp_a = np.interp(x_range_a, komm_a[H_column], komm_a[M_column])
```

[29]:
```
x_range_b = np.linspace(komm_b[H_column].min(), komm_b[H_column].max(), 100)
interp_b = np.interp(x_range_b, komm_b[H_column], komm_b[M_column])
```

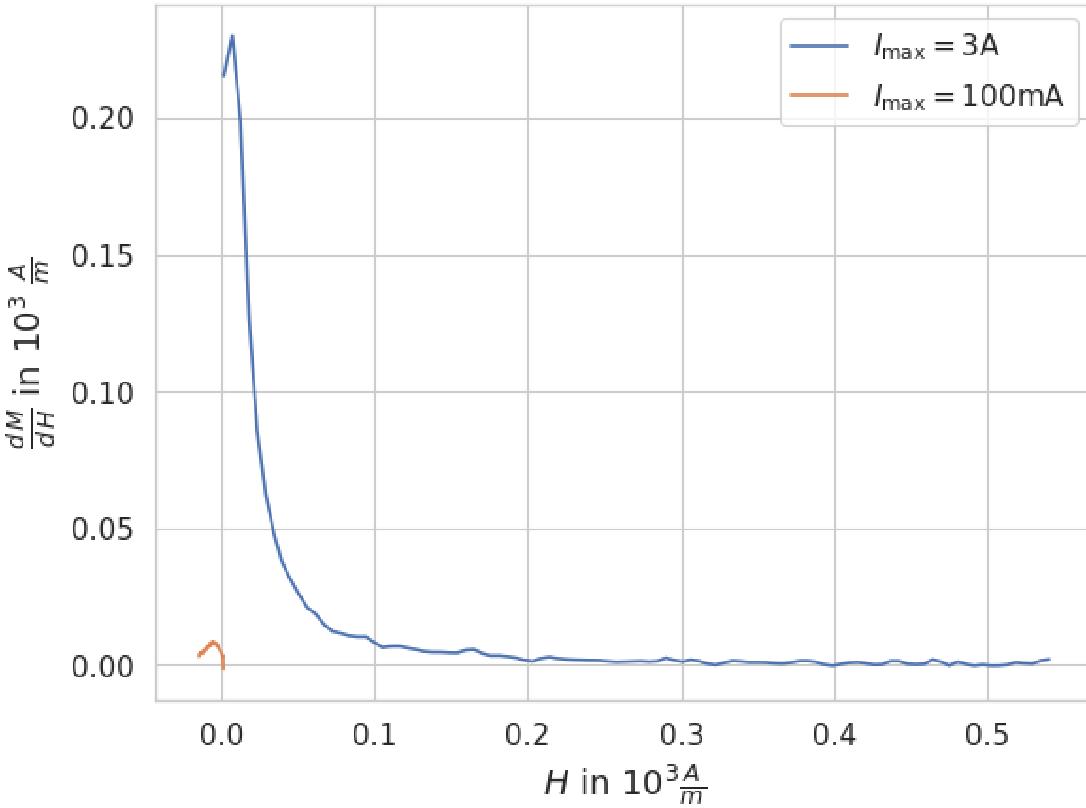
[30]:
```
komm_b[H_column_detailed] = komm_b[H_column] * 1000
```

[31]:
```
fig = plt.figure(figsize=(11,5))
fig.subplots_adjust(hspace=0.3, wspace=0.3)

ax = fig.add_subplot(1, 2, 1)
plt.title(r'$I_\mathrm{max} = 3$A')
sns.scatterplot(
    data=komm_a,
    x=H_column,
    y=M_column,
    ax=ax,
    label='Messwerte'
)
plt.plot(x_range_a, interp_a, color='green', label='interpoliert')
plt.legend()

ax = fig.add_subplot(1, 2, 2)
plt.title(r'$I_\mathrm{max} = 100$mA')
sns.scatterplot(
    data=komm_b,
    x=H_column_detailed,
    y=M_column,
    ax=ax,
    label='Messwerte'
)
plt.plot(x_range_b*1000, interp_b, color='green', label='interpoliert')
plt.legend()

fig.savefig('../../media/B2.4/3.3.2_Messung.svg', bbox_inches='tight')
```



Ableitung

[32]:
```
_, y_a = np.gradient([x_range_a, interp_a])
_, y_b = np.gradient([x_range_b, interp_b])

fig = plt.figure()

plt.plot(x_range_a, y_a[1], label=r'$I_\mathrm{max} = 3$A')
plt.plot(x_range_b, y_b[1], label=r'$I_\mathrm{max} = 100$mA')

plt.xlabel(H_column)
plt.ylabel(r'$\frac{d\,M}{d\,H}$ in $10^3\ \frac{A}{m}$')

plt.legend()

fig.savefig('../../media/B2.4/3.3.2_Ableitung.svg', bbox_inches='tight')
```

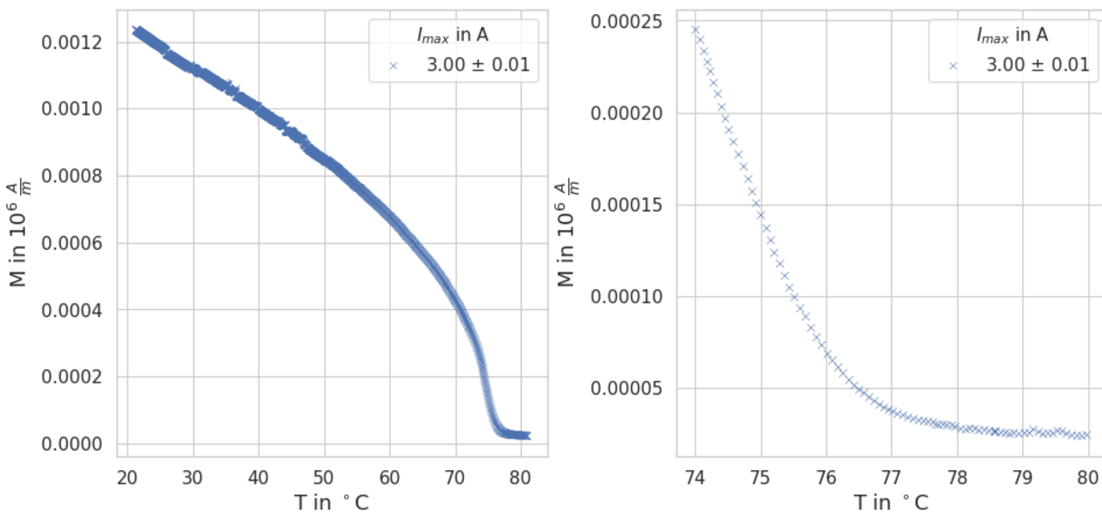


0.3 3.3.3

[33]:
```
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
```

[34]:
```
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

[35]:
```
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')
spalt_g = pd.read_csv('3.4.7.csv', sep='\t')
```

Fixme: Die Länge des Spalts muss eingerechnet werden.

[36]:
```
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
```

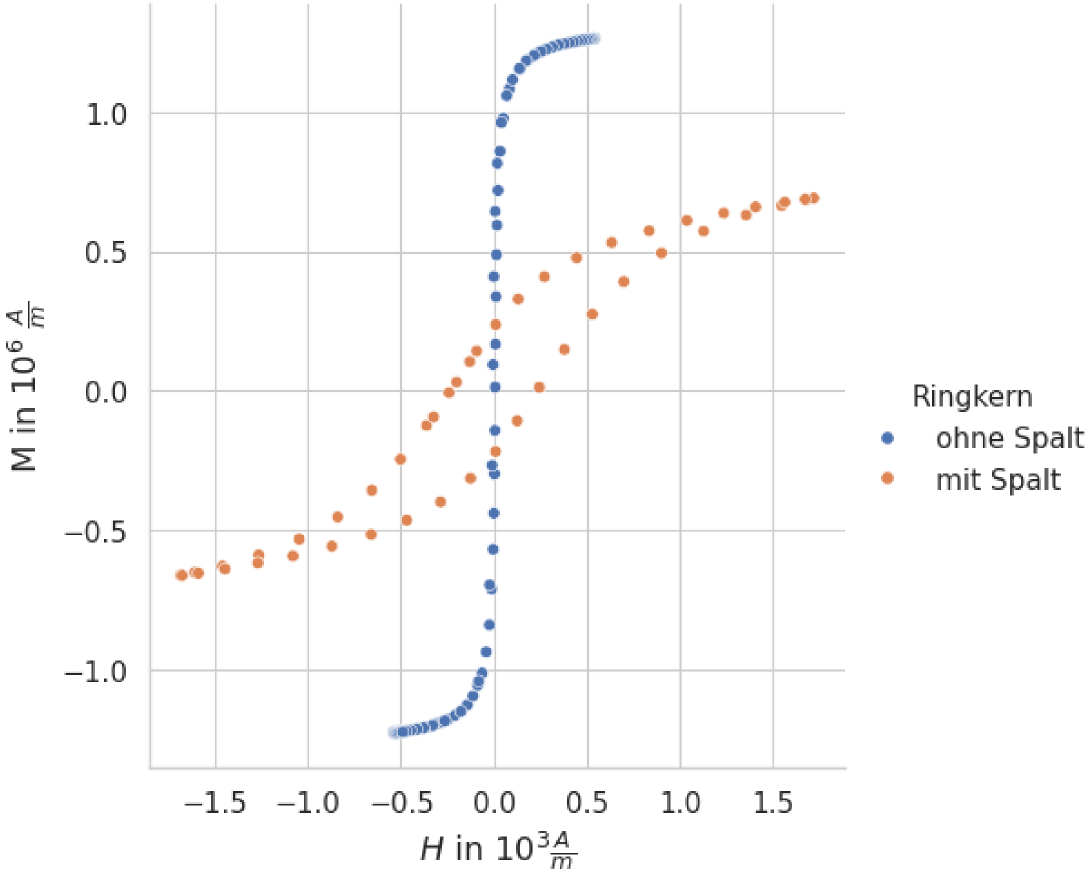
[37]:
```
spalt_a[H_column] = spalt_a['H'].apply(H_spalt)
spalt_b[H_column] = spalt_b['H'].apply(H_spalt)
spalt_c[H_column] = spalt_c['H'].apply(H_spalt)
spalt_d[H_column] = spalt_d['H'].apply(H_spalt)
spalt_e[H_column] = spalt_e['H'].apply(H_spalt)
spalt_f[H_column] = spalt_f['H'].apply(H_spalt)
spalt_g[H_column] = 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)
```

[38]:
```
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'
```

Vergleich

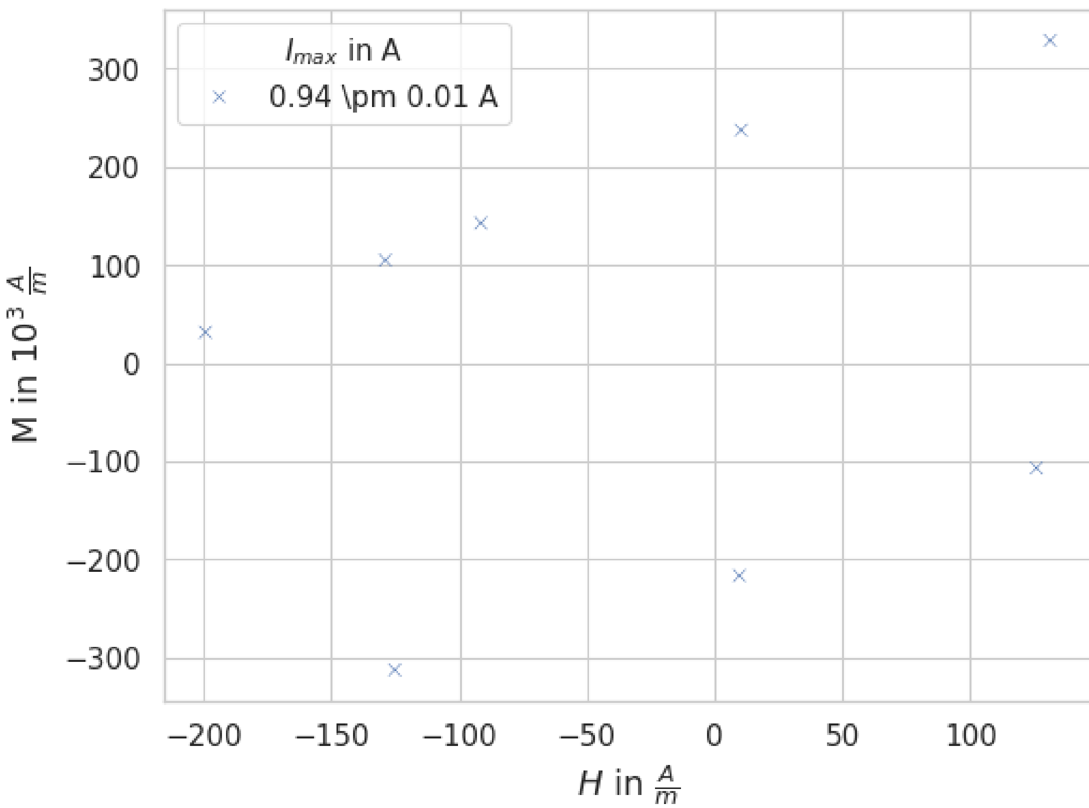
[39]:
```
def plot(data, hue_column=I_column, filename=None):
    img = sns.relplot(
        data=data,
        x=H_column,
        y=M_column,
        hue=hue_column,
        height=5,
        legend='full',

    )
    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')
```



[40]:
```
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'
```

[41]:
```
subplot(spalt_a[spalt_a[H_column_detailed].abs() < 200],
    x_column=H_column_detailed, y_column=M_column_detailed);
```



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.

[42]:
```
df = spalt_a
threshold = 100

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

[42]:
```
21   -216.736368
43    237.407872
44    142.691131
Name: M in $10^3\ \frac{A}{m}$, dtype: float64
```

[43]:
```
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
```

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.

[44]:
```
df = spalt_a
threshold = 50

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

[44]:
```
    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
```

[45]:
```
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
```

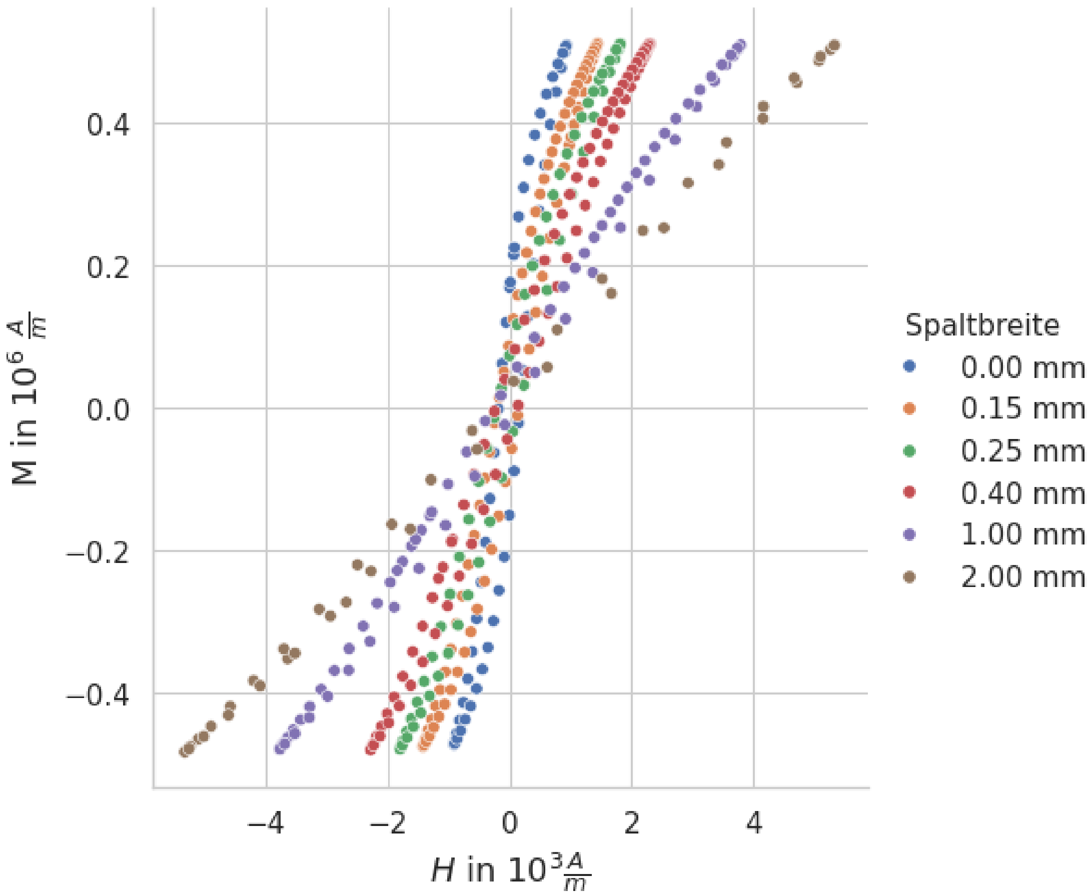
M_{max}

[46]:
```
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
```

Entmagnetisierungsfaktor

[47]:
```
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

[48]:
```
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
```

M_{max}

[49]:
```
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
```

N experimentell Hier ist M um einen Faktor 10^3 verändert, da die Größenordnung von H und M sich um diesen Faktor unterscheidet.

Die zurückgegebenen Werte werden in $10^{-3}\frac{A}{m}$ angegeben.

[50]:
```
def N(delta_H):
    M_max = 0.495 * 1e3

    avg = delta_H / M_max
    return round(avg * 1e3, 3)
```

[51]:
```
def N_err(delta_H, delta_H_err):
    # fix magnitude
    h = delta_H * 1e3
    err_h = delta_H_err * 1e3

    # constants
    M_max = 0.495 * 1e6
    err_M = 0.025 * 1e6

    err_squared = (err_h/M_max)**2 + (h*err_M/(M_max**2))**2
    return round(math.sqrt(err_squared) * 1e3, 3)
```

[52]:
```
def N_theo(l_L):
    R = 15 # mm
    return round(l_L / (2*math.pi*R + l_L) * 1e3, 3)
```

[53]:
```
spaltbreiten = [0, 0.15, 0.25, 0.4, 1, 2]
delta_H = [(0,0), (0.508, 0.001), (0.885, 0.001), (1.368, 0.001), (2.860, 0.
↪004), (4.410, 0.009)]

N_exp_result = [ N(h) for h, err in delta_H ]
N_exp_err = [ N_err(h, err) for h, err in delta_H ]
N_theo_result = [ N_theo(d) for d in spaltbreiten ]
```

[54]:
```
df = pd.DataFrame(
    {
        '$l_L$ in [mm]': spaltbreiten,
        '$N$': N_exp_result,
        r'$\Delta N$': N_exp_err,
        r'$N_\mathrm{theo}$': N_theo_result
    }
)
```

```
df
```

[54]:

	l_L in [mm]	N	ΔN	N_theo
0	0.00	0.000	0.000	0.000
1	0.15	1.026	0.052	1.589
2	0.25	1.788	0.090	2.646
3	0.40	2.764	0.140	4.226
4	1.00	5.778	0.292	10.499
5	2.00	8.909	0.450	20.780

[55]:
```
ax = sns.scatterplot(df, x='$l_L$ in [mm]', y='$N$', label=r'$N_\mathrm{exp}$',
 ↪color='blue')
sns.scatterplot(df, x='$l_L$ in [mm]', y=r'$N_\mathrm{theo}$',
 ↪label=r'$N_\mathrm{theo}$', color='red', ax=ax)
sns.mpl.pyplot.errorbar(x=df['$l_L$ in [mm]'], y=df['$N$'], yerr=df[r'$\Delta
 ↪N$'], color='blue')

ax.figure.savefig('../../media/B2.4/3.3.4_N.svg', bbox_inches='tight')
```