Notebook

February 23, 2023

[1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
d = 0.5 # mm
dd = 0.01 # mm

# Messwerte Versuch 1
exp1 = pd.DataFrame({
    "n":  [1, 2, 3, 4, 5, 6],
    "l":  [52, 52, 52, 52,  36, 36 ],
    "U":  [1.53, 1.45, 0.99, 1.00, 0.23, 0.3 ],
    "I":  [0.52, 1.2, 0.6, 1.6, 0.88, 1.15],
})
exp1["dl"] = 0.5

# digital: 0.5% + 1, analog: 1.5%
exp1["dU"] = (exp1["U"] * 0.005 + 0.001) \
    .where(cond = exp1['n']%2==1, other = exp1["U"] * 0.015)

# digital: 1.5% + 1, analog: 2.5%
exp1["dI"] = (exp1["I"] * 0.015 + 0.001) \
    .where(cond = exp1['n']%2==0, other = exp1["I"] * 0.02)

exp1["R"] = exp1["U"] / exp1["I"]
exp1["dR"] = ((exp1["dU"]/exp1["I"])**2 + (exp1["U"]/exp1["I"]**2 *␣
 ↪exp1["dI"])**2).pow(0.5)

# exp1["Rprol"] = exp1["R"] / exp1["l"]
# exp1["dRprol"] = ((exp1["dR"]/exp1["l"])**2 + (exp1["R"]/exp1["l"]**2 *␣
 ↪exp1["dl"])**2).pow(0.5)

exp1["rho"] = exp1["U"] * math.pi * (d/2)**2 / (exp1["I"] * exp1["l"]) * 1e-4 #␣
 ↪Ohm×m
```

```
exp1["drhoU"] = exp1["dU"] * math.pi * (d/2)**2 / (exp1["I"]    * exp1["l"]) ␣
↪* 1e-4
exp1["drhoI"] = exp1["U"]  * math.pi * (d/2)**2 / (exp1["I"]**2 * exp1["l"]) ␣
↪* 1e-4 * exp1["dI"]
exp1["drhol"] = exp1["U"]  * math.pi * (d/2)**2 / (exp1["I"]    * exp1["l"]**2)␣
↪* 1e-4 * exp1["dl"]
exp1["drhod"] = exp1["U"]  * math.pi * (d/2)    / (exp1["I"]    * exp1["l"]) ␣
↪* 1e-4 * dd * 2
exp1["drho"] = (exp1["drhoU"]**2 + exp1["drhoI"]**2 + exp1["drhol"]**2 +␣
↪exp1["drhod"]**2)**0.5
#exp1["drho2"] = np.sum(np.square(np.array([drhoU + drhoI, drhol, drhod])))**0.5


exp1
```

[3]:
```
with open("table1.tex","w") as f:
    f.write('''
\\begingroup
\\sisetup{round-mode=uncertainty,round-precision=2}
\\begin{tabular}{rSSSS}
\\toprule
{n} & {$l$ (\\unit{\\cm})} & {$U$ (\\unit{\\V})} & {$I$ (\\unit{\\A})} &␣
↪{$\\varrho$ (\\unit{\\micro\\ohm\\m})} \\\\
\\midrule
''')
    for index, row in exp1.iterrows():
        f.write(f"{int(row['n'])} & ")
        f.write(f"{row['l']}\\pm{row['dl']} & ")
        f.write(f"{row['U']}\\pm{row['dU']} & ")
        f.write(f"{row['I']}\\pm{row['dI']} & ")
        f.write(f"{row['rho']*1e6}\\pm{row['drho']*1e6} ")
        f.write(" \\\\\n")

    f.write('''
\\bottomrule
\\end{tabular}
\\endgroup
''')
```

[4]:
```
# Auswahl: Schaltung 5

e = exp1.iloc[4]

rho = e["rho"] * 1e6
drhoU = e["drhoU"] * 1e6
drhoI = e["drhoI"] * 1e6
```

```
drhol = e["drhol"] * 1e6
drhod = e["drhod"] * 1e6
drho = e["drho"] * 1e6

tab3 = pd.DataFrame({
    r"$\varrho$ ($\mathrm{\mu\Omega m}$)": [rho],
    r"$|\frac{\partial \varrho}{\partial U}|\Delta U$": [drhoU],
    r"$|\frac{\partial \varrho}{\partial I}|\Delta I$": [drhoI],
    r"$|\frac{\partial \varrho}{\partial l}|\Delta l$": [drhol],
    r"$|\frac{\partial \varrho}{\partial d}|\Delta d$": [drhod],
    r"$\Delta \varrho$": [drho],
#   r"$\Delta \rho'$": [drho2],
})
tab3.style.format(precision=4).hide(axis='index') \
    .to_latex(buf = f"table3.tex", hrules = True, siunitx=True)
tab3
```

[5]:
```
# rho_lit = 0.124 # μ Ohm m, https://www.spektrum.de/lexikon/physik/tantal/
 ↪14319 (ohne Angabe der Temperatur)
rho_lit = 0.131 # μ Ohm m, Wikipedia

t = abs(rho-rho_lit)/drho

t
```

[6]:
```
# Messwerte Versuch 2

exp2 = pd.DataFrame({
    "l":  [44.5, 39.5, 35.0, 31.5, 28.5, 25.5, 23.0, 20.5, 16.5, 14.5, 12.3, 16.
 ↪5 ],
    "U":  [0.33, 0.296, 0.264, 0.239, 0.206, 0.192, 0.187, 0.169, 0.152, 0.115,␣
 ↪0.097, 0.128],
})

# Rechnung
exp2["dl"] = 0.5
exp2["dU"] = exp2["U"] * 0.005 + 0.001 # 0.5% + 1
I = 1.0
dI = I * 0.02 # 2%

exp2["R"] = exp2["U"] / I
exp2["dR"] = ((exp2["dU"]/I)**2 + (exp2["U"]/I*dI)**2).pow(0.5)

exp2["rho"] = exp2["U"] * math.pi * (d/2)**2 / (I * exp2["l"]) * 1e-4 # Ohm×m

exp2["drhoU"] = exp2["dU"] * math.pi * (d/2)**2 / (I    * exp2["l"])    * 1e-4
```

```
exp2["drhoI"] = exp2["U"]  * math.pi * (d/2)**2 / (I**2 * exp2["l"])    * 1e-4␣
↪* dI
exp2["drhol"] = exp2["U"]  * math.pi * (d/2)**2 / (I    * exp2["l"]**2) * 1e-4␣
↪* exp2["dl"]
exp2["drhod"] = exp2["U"]  * math.pi * (d/2)    / (I    * exp2["l"])    * 1e-4␣
↪* dd * 2
exp2["drho"] = (exp2["drhoU"]**2 + exp2["drhoI"]**2 + exp2["drhol"]**2 +␣
↪exp2["drhod"]**2)**0.5


exp2
```

[7]:
```
# Bestimmung \varrho aus exp2

rho2 = (exp2["rho"]/exp2["drho"]**2).sum() / (1/exp2["drho"]**2).sum() * 1e6
drho2 = 1 / (1/exp2["drho"]**2).sum()**0.5  * 1e6
t2 = abs(rho2-rho_lit)/drho2

(rho2, drho2, t2)
```

[8]:
```
with open("table2.tex","w") as f:
    f.write('''
\\begingroup
\\sisetup{round-mode=uncertainty,round-precision=2}
\\begin{tabular}{SSS}
\\toprule
{$l$ (\\unit{\\cm})} & {$U$ (\\unit{\\V})} & {$\\varrho$␣
↪(\\unit{\\micro\\ohm\\m})} \\\\
\\midrule
''')
    for index, row in exp2.iterrows():
        f.write(f"{row['l']}\\pm{row['dl']} & ")
        f.write(f"{row['U']}\\pm{row['dU']} & ")
        f.write(f"{row['rho']*1e6}\\pm{row['drho']*1e6} ")
        f.write(" \\\\\\\n")

    f.write('''
\\bottomrule
\\end{tabular}
\\endgroup
''')
```

[9]:
```
# Lineare Regression
# Quelle: https://home.uni-leipzig.de/prakphys/pdf/
↪LA_EP1_Einf%C3%BChrung_WS2014_2.pdf
```

```python
x = exp2["l"]
y = exp2["R"]

n = len(x)
xs  = 1/n * x.sum()
x2s = 1/n * x.pow(2).sum()
ys  = 1/n * y.sum()
xys = 1/n *(x*y).sum()

a1 = (x2s*ys - xs*xys)/(x2s - xs**2)
b1 = (xys - xs*ys)/(x2s - xs**2)
s = (1/(n-2) * (y - (a1 + b1*x)).pow(2).sum())**0.5
da1 = s * (x2s / (n *(x2s - xs**2)))**0.5
db1 = s * (1 / (n *(x2s - xs**2)))**0.5

pd.DataFrame({
    r"$a$ ($\Omega$)": [a1],
    r"$b$ ($\frac{\Omega}{\mathrm{cm}}$)": [b1],
    r"$\Delta a$": [da1],
    r"$\Delta b$": [db1]})
```

[10]:
```python
plt.figure(figsize=(10, 4))
plt.ylim(0, 0.4)
plt.xlim(0, 50)
#plt.margins(x=0, y =0)
plt.xlabel(r'$l$ ($\mathrm{cm}$)')
plt.ylabel(r'$R$ ($\Omega$)')
plt.plot([0,50],[a1, a1+50*b1], label=f'Regressionsgerade')
plt.errorbar(exp2["l"], exp2["R"], exp2["dU"], exp2["dl"], label='Messdaten',␣
  ↪marker = "o", ms=4, ls='none')
#plt.errorbar(df["m"], df["s"], 0.01, 0, label='Fehlerbalken', ms=4, ls='none')
plt.legend(loc='upper right')
plt.savefig(f"plot1.pdf")
plt.show()
```

[11]:
```python
plt.figure(figsize=(10, 2))
#plt.margins(x=0, y =0)
plt.xlabel(r'$l$ (cm)')
plt.ylabel(r'$R$-Residuen ($\Omega$)')
plt.plot([0,50],[0, 0], label=f'Regressionsgerade')
plt.plot(exp2["l"], exp2["R"] - (a1+exp2["l"]*b1), label='Residuen', marker =␣
  ↪"o", ms=4, ls='none')
plt.errorbar(exp2["l"], exp2["R"] - (a1+exp2["l"]*b1), exp2["dR"], exp2["dl"],␣
  ↪label='Fehlerbalken', ms=4, ls='none')
#plt.legend(loc='upper right')
plt.savefig(f"plot1residuen.pdf")
plt.show()
```

[29]:

```python
# Temperaturberechnung

dens = 16.65 # g / cm³
wkap = 140.0 # J / kgK


incr = (exp2.iloc[-1]["rho"] / exp2.iloc[0]["rho"] - 1) * 100
incrT = incr/0.5
power = exp2.iloc[0]["U"] * I
vol = exp2.iloc[0]["l"] * math.pi * (d/2 * 0.1) **2 # cm³
mass = dens * vol # g
time = (wkap * (mass * 1e-3) * incrT) / power
time
```

[30]:

```python
# Exporting all locals

outfile = open("defs.tex", "w")
outfile.write(r"""
\newcommand{\DefVal}[2]{%
  \expandafter\newcommand\csname val-#1\endcsname{#2}%
}
\newcommand{\Val}[1]{\csname val-#1\endcsname}
""")
for (n, x) in locals().items():
    if type(x) in [float, np.float64]:
        outfile.write(f"\\DefVal{{{n}}}{{{np.format_float_positional(x,␣
 ↪trim='-')}}}\n")
outfile.close()
#print(open("defs.tex").read())
```