APL_2_cell

May 27, 2021

[122]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[123]:
```
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=False, grid=False)
```

[124]:
```
import matplotlib as mpl
```

[125]:
```
# Edit the font, font size, and axes width
# mpl.rcParams['font.family'] = 'Avenir'
plt.rcParams['font.size'] = 24
plt.rcParams['axes.linewidth'] = 2
```

[126]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[127]:
```
import sys
sys.path.insert(0, '/home/trevormjs/Documents/Science/APL/Lab')
```

[128]:
```
from Helper.plotting import my_graph
```

0.0.1 Part A

Measurements

[129]:
```
silicon_thickness = 0.54  # mm
silicon_dimensions = [32.33, 9.04]
wire_colors = {
    1: 'purple',
    2: 'blue',
    3: 'green',
    4: 'black',
    5: 'white',
    6: 'grey'
}
color_wires = {i:k for k, i in wire_colors.items()}
```

[130]:
```
resistances = pd.DataFrame({
    'first':[
        'green', 'green', 'green',
        'green', 'green', 'blue',
        'blue', 'blue', 'blue',
        'grey', 'grey', 'grey',
        'white', 'white', 'black',
    ],
    'second':[
        'blue', 'grey', 'white',
        'black', 'purple', 'grey',
        'white', 'black', 'purple',
        'white', 'black', 'purple',
        'black', 'purple', 'purple',
    ],
    'resistance':[
        22.76e3, 24.55e3, 66.43e3,
        26.94e3, 28.57e3, 11.213e3,
        12.445e3, 12.06e3, 11.665e3,
        51.39e3, 8.8936e3, 9.566e3,
        171.76e3, 176.26e3, 12.296e3,
    ]
})
```

[131]:
```
resistances.insert(2, '#1', [color_wires[color] for color in␣
 ↪resistances['first']])
resistances.insert(3, '#2', [color_wires[color] for color in␣
 ↪resistances['second']])
```

Data

[264]:
```
voltages = pd.DataFrame(columns=['current', 'voltage', 'resistance', 'power'])


def add_row(c, v, r=12296.0):
    voltages.loc[voltages.shape[0], :] = [c, v, r, c*v]
    display(voltages.iloc[-1])
```

Adding

[265]:
```
add_row(0.0022, 0.0183)

add_row(0.0071, 0.0836)

add_row(0.0128, 0.2103)

add_row(0.0201, 0.4462)
```

```
add_row(0.0441, 1.0425)

add_row(0.0656, 1.501)

add_row(-0.0011, -0.0196)

add_row(-0.0044, -0.0603)

add_row(-0.0105, -0.1550)

add_row(-0.0133, -0.2201)

add_row(-0.0236, -0.5823)

add_row(-0.0306,
        -0.9142)

add_row(-0.0352,
        -1.1616)

add_row(-0.0383,
        -1.3490)

add_row(-0.0424,
        -1.5880)

add_row(-0.0482,
        -1.9278)

voltages
```

```
current        0.0022
voltage        0.0183
resistance    12296.0
power         0.00004
Name: 0, dtype: object
```

```
current         0.0071
voltage         0.0836
resistance     12296.0
power         0.000594
Name: 1, dtype: object
```

```
current         0.0128
voltage         0.2103
resistance     12296.0
power         0.002692
Name: 2, dtype: object
```

```
current         0.0201
voltage         0.4462
resistance     12296.0
power         0.008969
Name: 3, dtype: object

current         0.0441
voltage         1.0425
resistance     12296.0
power         0.045974
Name: 4, dtype: object

current         0.0656
voltage          1.501
resistance     12296.0
power         0.098466
Name: 5, dtype: object

current        -0.0011
voltage        -0.0196
resistance     12296.0
power         0.000022
Name: 6, dtype: object

current        -0.0044
voltage        -0.0603
resistance     12296.0
power         0.000265
Name: 7, dtype: object

current        -0.0105
voltage         -0.155
resistance     12296.0
power         0.001628
Name: 8, dtype: object

current        -0.0133
voltage        -0.2201
resistance     12296.0
power         0.002927
Name: 9, dtype: object

current        -0.0236
voltage        -0.5823
resistance     12296.0
power         0.013742
Name: 10, dtype: object

current        -0.0306
voltage        -0.9142
resistance     12296.0
```

```
power         0.027975
Name: 11, dtype: object

current        -0.0352
voltage        -1.1616
resistance     12296.0
power         0.040888
Name: 12, dtype: object

current        -0.0383
voltage         -1.349
resistance     12296.0
power         0.051667
Name: 13, dtype: object

current        -0.0424
voltage         -1.588
resistance     12296.0
power         0.067331
Name: 14, dtype: object

current       -0.0482
voltage       -1.9278
resistance    12296.0
power         0.09292
Name: 15, dtype: object
```

[265]:

	current	voltage	resistance	power
0	0.0022	0.0183	12296.0	0.00004
1	0.0071	0.0836	12296.0	0.000594
2	0.0128	0.2103	12296.0	0.002692
3	0.0201	0.4462	12296.0	0.008969
4	0.0441	1.0425	12296.0	0.045974
5	0.0656	1.501	12296.0	0.098466
6	-0.0011	-0.0196	12296.0	0.000022
7	-0.0044	-0.0603	12296.0	0.000265
8	-0.0105	-0.155	12296.0	0.001628
9	-0.0133	-0.2201	12296.0	0.002927
10	-0.0236	-0.5823	12296.0	0.013742
11	-0.0306	-0.9142	12296.0	0.027975
12	-0.0352	-1.1616	12296.0	0.040888
13	-0.0383	-1.349	12296.0	0.051667
14	-0.0424	-1.588	12296.0	0.067331
15	-0.0482	-1.9278	12296.0	0.09292

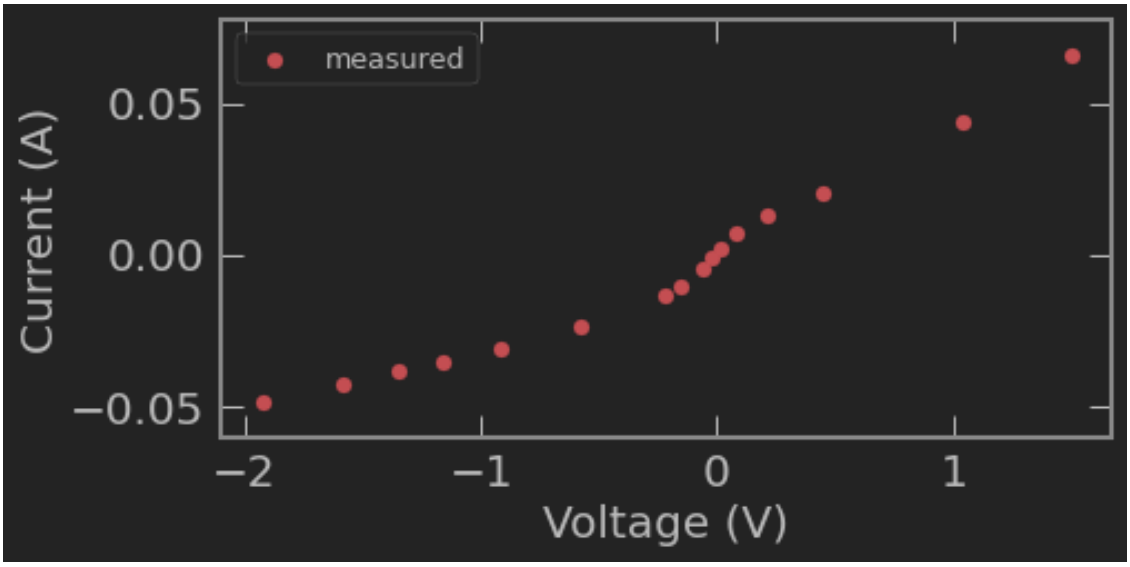
[267]:
```
print(voltages.to_latex())
```

```
\begin{tabular}{lllll}
\toprule
{} & current & voltage & resistance &     power \\
```

```
\midrule
0  &  0.0022 &  0.0183 &    12296.0 &   0.00004 \\
1  &  0.0071 &  0.0836 &    12296.0 &  0.000594 \\
2  &  0.0128 &  0.2103 &    12296.0 &  0.002692 \\
3  &  0.0201 &  0.4462 &    12296.0 &  0.008969 \\
4  &  0.0441 &  1.0425 &    12296.0 &  0.045974 \\
5  &  0.0656 &   1.501 &    12296.0 &  0.098466 \\
6  & -0.0011 & -0.0196 &    12296.0 &  0.000022 \\
7  & -0.0044 & -0.0603 &    12296.0 &  0.000265 \\
8  & -0.0105 &  -0.155 &    12296.0 &  0.001628 \\
9  & -0.0133 & -0.2201 &    12296.0 &  0.002927 \\
10 & -0.0236 & -0.5823 &    12296.0 &  0.013742 \\
11 & -0.0306 & -0.9142 &    12296.0 &  0.027975 \\
12 & -0.0352 & -1.1616 &    12296.0 &  0.040888 \\
13 & -0.0383 &  -1.349 &    12296.0 &  0.051667 \\
14 & -0.0424 &  -1.588 &    12296.0 &  0.067331 \\
15 & -0.0482 & -1.9278 &    12296.0 &   0.09292 \\
\bottomrule
\end{tabular}
```

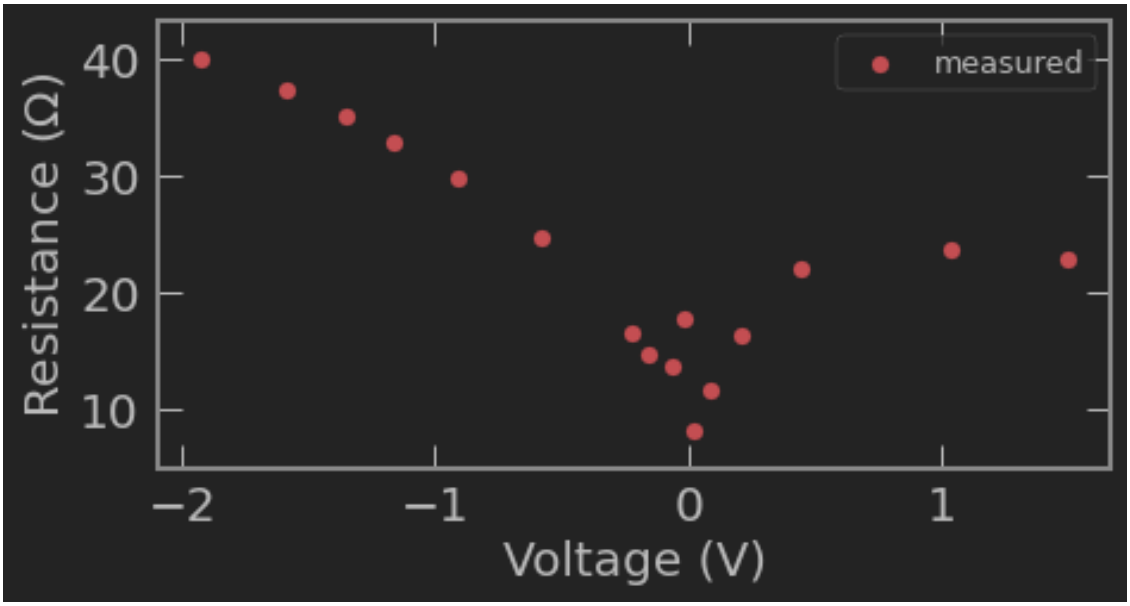
Results

[135]:
```
my_graph(voltages.voltage, voltages.current, 'linear','linear','Voltage (V)',␣
↪'Current (A)', 'l2_a_1', '')
```



[136]:
```
rohms = 'Resistance ($\\mathregular{\\Omega}$)'
```

[137]:
```
my_graph(voltages.voltage, voltages.voltage/voltages.current,
         'linear', 'linear', 'Voltage (V)', rohms,  'l2_a_2', '', )
```



0.0.2 Part B

Measurements

[174]:
```
volts = 0.9083
amps = 0.0396e-3

fourwire = pd.DataFrame({
    'start':[
        'blue','white',
    ],
    'end':[
        'green','grey',
    ],
    'voltage':[
        5.84e-3, 9.33e-3
    ]
})
```

Calculations

[175]:
```
fourwire.insert(3, 'resistance', fourwire.voltage/amps)
fourwire.insert(4, 'distance', [9.46, 11.63])
fourwire
```

[175]:

	start	end	voltage	resistance	distance
0	blue	green	0.00584	147.474747	9.46
1	white	grey	0.00933	235.606061	11.63

[178]:
```
csa = cross_sectional_area = silicon_dimensions[1]*1e-3 * silicon_thickness*1e-3
csa
```

[178]: 4.8816e-06

$$R = \rho \cdot \frac{L}{A}$$
$$\rho = \frac{R \cdot A}{L}$$

[177]:
```
fourwire.insert(5, 'rho', fourwire.resistance * csa / (fourwire.distance*1e-3))

fourwire.insert(6, 'measured_R', [20.2e3, 213.6e3])

fourwire
```

[177]:

	start	end	voltage	resistance	distance	rho	measured_R
0	blue	green	0.00584	147.474747	9.46	0.076101	20200.0
1	white	grey	0.00933	235.606061	11.63	0.098894	213600.0

[167]:
```
from Helper.numbers import print_unc
```

[183]:
```
3.5 *1e-6*1e-2
```

[183]: 3.5e-08

[182]:
```
_ = print_unc(fourwire.rho.mean(), fourwire.rho.std())
```

```
0.09 +- 0.02
```

0.0.3 Hall effect, carrier concentration and mobility

Calibration of the Electromagnet

[142]:
```
data = pd.read_excel('./Lab2_HALL_alpha_new.xlsx', skiprows = 1)
```

Fit

[143]:
```
alpha, beta = data.iloc[3, -2:]
data = data.iloc[:,1:3]
data
```

[143]:

	Im (A)	B (T)
0	-3.00	-0.487
1	-2.50	-0.408
2	-2.00	-0.327

```
3    -1.50 -0.246
4    -1.00 -0.164
5    -0.50 -0.082
6     0.00  0.001
7     0.50  0.078
8     1.00  0.160
9     1.50  0.240
10    2.00  0.321
11    2.49  0.402
12    3.00  0.481
```

Cross-voltages

[144]:
```
current = None # Need this I think
```

[145]:
```
cross_voltage = {'grey-blue': 73.9e-3,
                 'white-green': .2906}
cross_voltage
```

[145]:
```
{'grey-blue': 0.0739, 'white-green': 0.2906}
```

Measure Hall Voltage

Data

[146]:
```
h_v = pd.DataFrame(columns=['Is', 'Im', 'Vh'])

def add_row(Is, Im, Vh):
    h_v.loc[h_v.shape[0], :] = [Is, Im, Vh]
#     display(h_v.iloc[-1])

add_row(7.86e-3, 0, 71.22e-3)
add_row(7.87e-3, .18, 70.23e-3)
add_row(7.87e-3, .36, 68.91e-3)
add_row(7.917e-3, .66, 68.15e-3)
add_row(7.906e-3, 1.06, 64.91e-3)
add_row(7.906e-3, 1.53, 62.28e-3)
add_row(7.912e-3, 2.01, 59.28e-3)
add_row(7.911e-3, 2.32, 57.37e-3)
add_row(7.874e-3, 2.69, 54.48e-3)
add_row(7.901e-3, 2.98, 53.01e-3)
add_row(7.913e-3, -1e-10, 72.67e-3)
add_row(7.913e-3, -.32, 74.46e-3)
add_row(7.940e-3, -.56, 76.44e-3)
add_row(7.933e-3, -.85, 77.53e-3)
add_row(7.950e-3, -1.01, 79.10e-3)
add_row(7.929e-3, -1.30, 81.22e-3)
add_row(7.918e-3, -1.67, 83.04e-3)
```

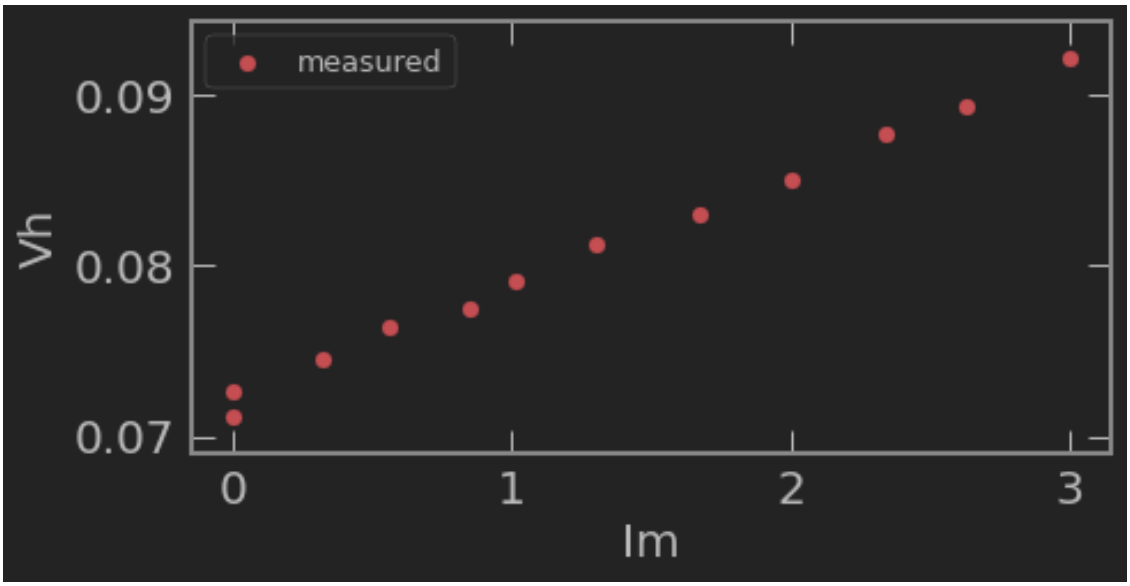
```
add_row(7.925e-3, -2.00, 85.10e-3)
add_row(7.941e-3, -2.34, 87.73e-3)
add_row(7.927e-3, -2.63, 89.43e-3)
add_row(7.952e-3, -3.00, 92.15e-3)

h_v
```

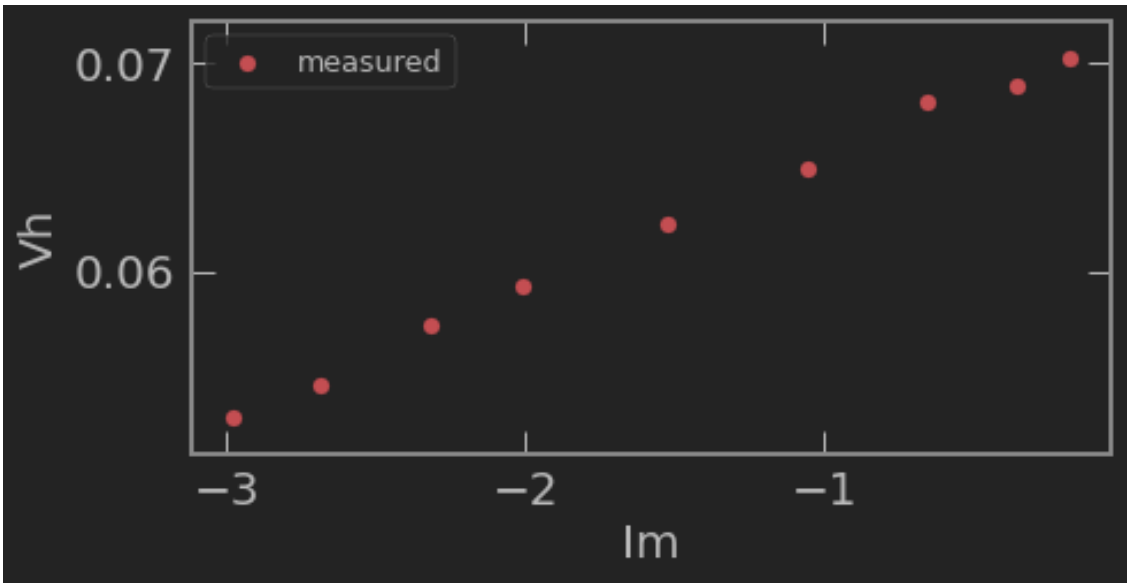
[146]:

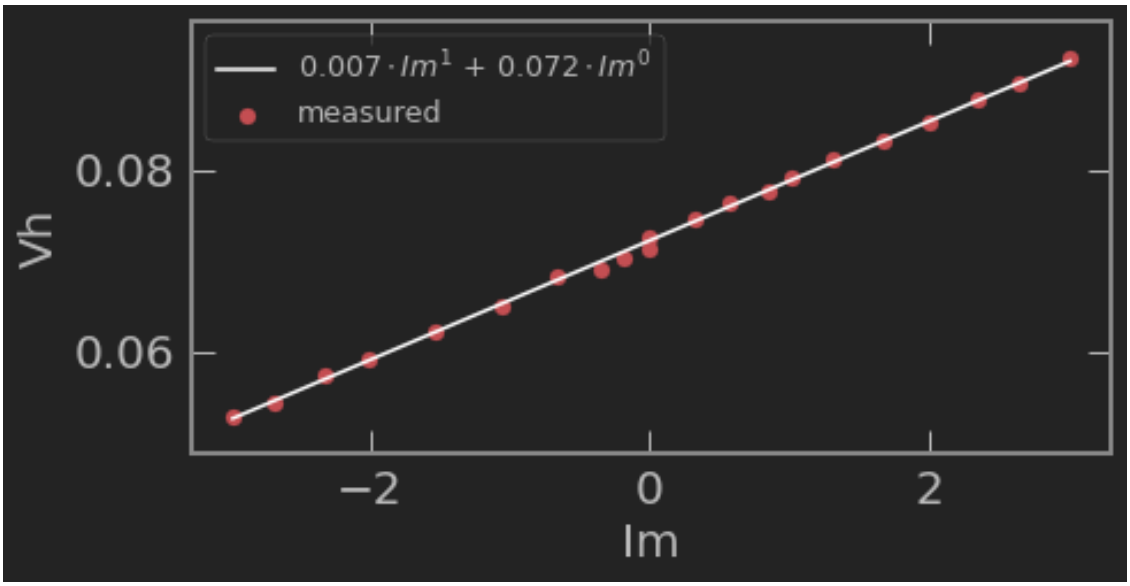
	Is	Im	Vh
0	0.00786	0	0.07122
1	0.00787	0.18	0.07023
2	0.00787	0.36	0.06891
3	0.007917	0.66	0.06815
4	0.007906	1.06	0.06491
5	0.007906	1.53	0.06228
6	0.007912	2.01	0.05928
7	0.007911	2.32	0.05737
8	0.007874	2.69	0.05448
9	0.007901	2.98	0.05301
10	0.007913	-0.0	0.07267
11	0.007913	-0.32	0.07446
12	0.00794	-0.56	0.07644
13	0.007933	-0.85	0.07753
14	0.00795	-1.01	0.0791
15	0.007929	-1.3	0.08122
16	0.007918	-1.67	0.08304
17	0.007925	-2.0	0.0851
18	0.007941	-2.34	0.08773
19	0.007927	-2.63	0.08943
20	0.007952	-3.0	0.09215

[147]:
```
h_v.Im *= -1
```

[148]:
```
my_graph(h_v['Im'].loc[h_v.Im >= 0], h_v['Vh'].loc[h_v.Im >=
                                                   0],
         'linear', 'linear', 'Im', 'Vh', 'l2_d_4a_pos', '')
my_graph(h_v['Im'].loc[h_v.Im < 0], h_v['Vh'].loc[h_v.Im <
                                                  0],
         'linear', 'linear', 'Im', 'Vh', 'l2_d_4a_neg', '')
my_graph(h_v['Im'], h_v['Vh'],
         'linear', 'linear', 'Im', 'Vh', 'l2_d_4a_all', '', 1)
```


measured
0.09
0.08
0.07
Vh
0
1
2
3
Im


measured
0.07
0.06
Vh
−3
−2
−1
Im



[148]:
```
array([0.00655619, 0.07220566])
```

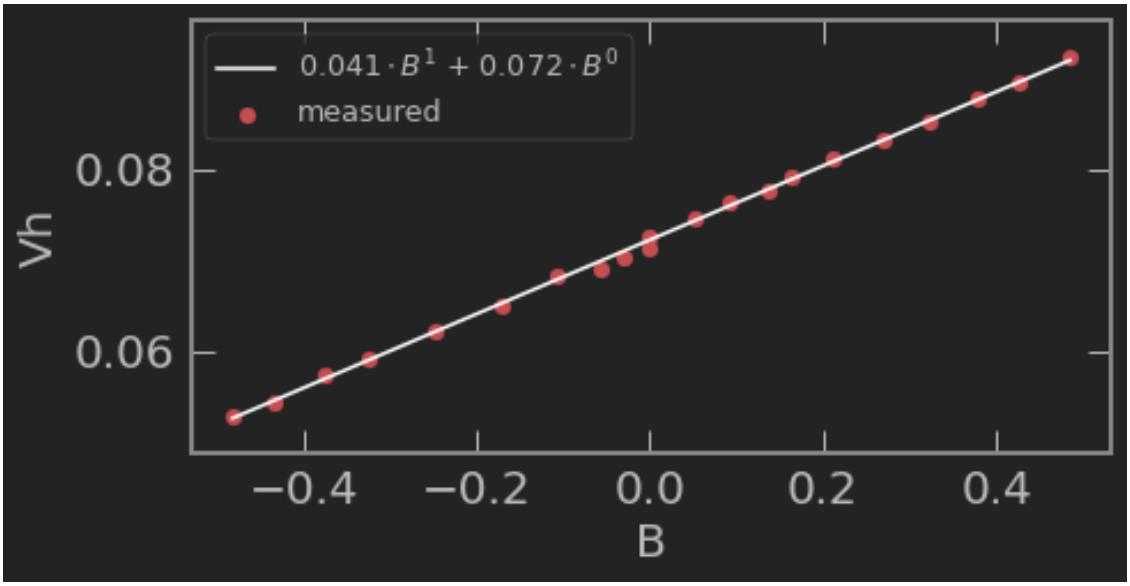
[149]:
```
h_v.insert(3, 'B', alpha*h_v.Im)
h_v
```

[149]:

	Is	Im	Vh	B
0	0.00786	0	0.07122	0.0
1	0.00787	-0.18	0.07023	-0.029125
2	0.00787	-0.36	0.06891	-0.058249
3	0.007917	-0.66	0.06815	-0.10679
4	0.007906	-1.06	0.06491	-0.171511
5	0.007906	-1.53	0.06228	-0.247558
6	0.007912	-2.01	0.05928	-0.325224
7	0.007911	-2.32	0.05737	-0.375383
8	0.007874	-2.69	0.05448	-0.43525
9	0.007901	-2.98	0.05301	-0.482173
10	0.007913	0.0	0.07267	0.0
11	0.007913	0.32	0.07446	0.051777
12	0.00794	0.56	0.07644	0.09061
13	0.007933	0.85	0.07753	0.137532
14	0.00795	1.01	0.0791	0.163421
15	0.007929	1.3	0.08122	0.210344
16	0.007918	1.67	0.08304	0.270211
17	0.007925	2.0	0.0851	0.323606
18	0.007941	2.34	0.08773	0.378619
19	0.007927	2.63	0.08943	0.425542
20	0.007952	3.0	0.09215	0.485409

[150]:
```
from Helper.numbers import get_leading_figure
```

[151]:
```
[dvh_db, _] = my_graph(h_v['B'], h_v['Vh'],
         'linear', 'linear', 'B', 'Vh', 'l2_d_4b', '', 1)
dvh_db
```



[151]:
```
0.04051960131985295
```

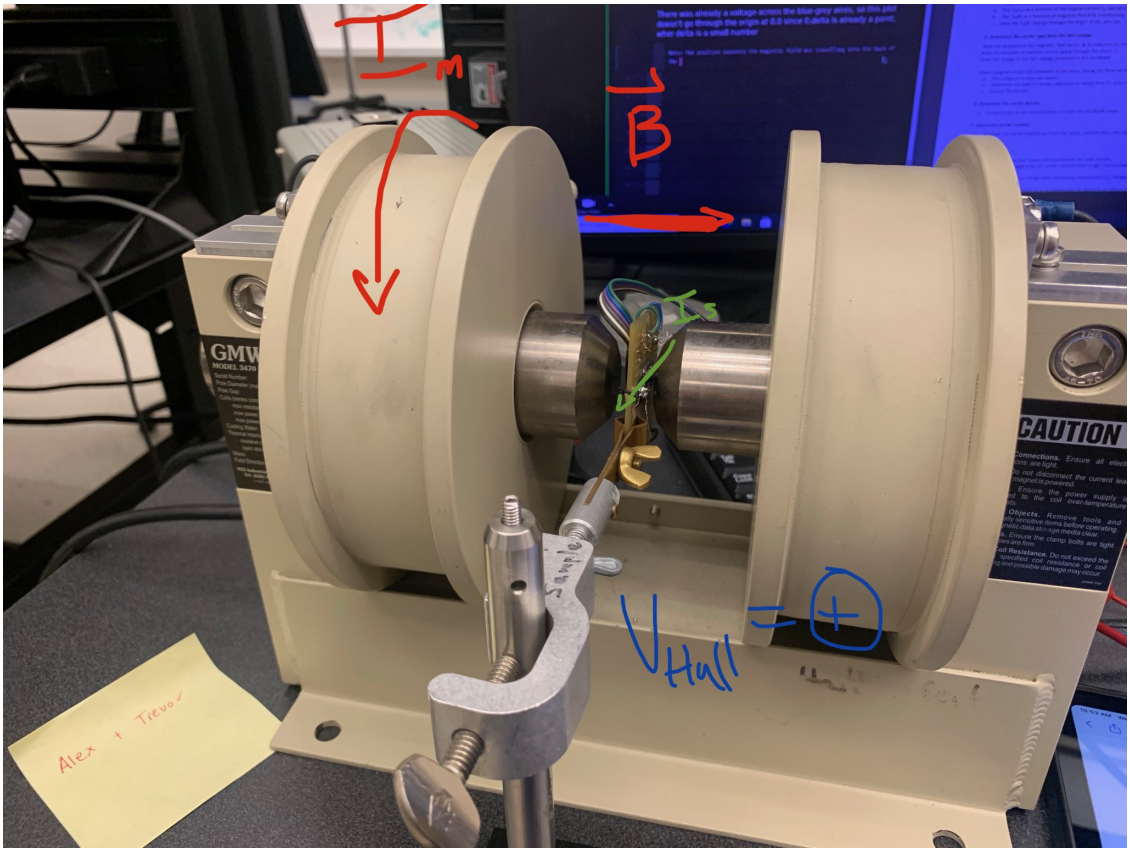
There was already a voltage across the blue-grey wires, so this plot doesn't go through the origin at 0,0 since 0,delta is already a point, wher delta is a small number

Note: For positive currents the magnetic field was travelling into the back of the chip, for negative currents the field was travelling into the top of the chip.

Current was travelling from blue to black

[152]:
```
from IPython.display import Image
Image("./messages_0.jpeg", width = 900)
```

[152]:



Determine carrier density

[252]:
```
errs = v_per_b*h_v['B'] - h_v['Vh']
abs(errs - errs.mean()).mean()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-252-7296535775cb> in <module>
      1 errs = v_per_b*h_v['B'] - h_v['Vh']
----> 2 abs(errs - errs.mean()).mean()

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/generic.py in␣
 ↪mean(self, axis, skipna, level, numeric_only, **kwargs)
  11116         )
  11117         def mean(self, axis=None, skipna=None, level=None,␣
 ↪numeric_only=None, **kwargs):
> 11118             return NDFrame.mean(self, axis, skipna, level, numeric_only,␣
 ↪**kwargs)
  11119
  11120         # pandas\core\generic.py:10924: error: Cannot assign to a method
```

```
~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/generic.py in␣
↪mean(self, axis, skipna, level, numeric_only, **kwargs)
 10724
 10725     def mean(self, axis=None, skipna=None, level=None, numeric_only=None,␣
↪**kwargs):
> 10726         return self._stat_function(
 10727             "mean", nanops.nanmean, axis, skipna, level, numeric_only,␣
↪**kwargs
 10728         )

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/generic.py in␣
↪_stat_function(self, name, func, axis, skipna, level, numeric_only, **kwargs)
 10709         if level is not None:
 10710             return self._agg_by_level(name, axis=axis, level=level,␣
↪skipna=skipna)
> 10711         return self._reduce(
 10712             func, name=name, axis=axis, skipna=skipna,␣
↪numeric_only=numeric_only
 10713         )

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/series.py in␣
↪_reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwds)
  4180                 )
  4181             with np.errstate(all="ignore"):
-> 4182                 return op(delegate, skipna=skipna, **kwds)
  4183
  4184     def _reindex_indexer(self, new_index, indexer, copy):

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/nanops.py in␣
↪_f(*args, **kwargs)
    71             try:
    72                 with np.errstate(invalid="ignore"):
---> 73                     return f(*args, **kwargs)
    74             except ValueError as e:
    75                 # we want to transform an object array

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/nanops.py in␣
↪f(values, axis, skipna, **kwds)
   133                     result = alt(values, axis=axis, skipna=skipna, **kwds
   134             else:
--> 135                 result = alt(values, axis=axis, skipna=skipna, **kwds)
   136
   137             return result

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/nanops.py in␣
↪new_func(values, axis, skipna, mask, **kwargs)
   392             mask = isna(values)
```

```
    393
--> 394         result = func(values, axis=axis, skipna=skipna, mask=mask,␣
 ↪**kwargs)
    395
    396         if datetimelike:

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/nanops.py in␣
 ↪nanmean(values, axis, skipna, mask)
    631
    632     count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
--> 633     the_sum = _ensure_numeric(values.sum(axis, dtype=dtype_sum))
    634
    635     if axis is not None and getattr(the_sum, "ndim", False):

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/pandas/core/nanops.py in␣
 ↪_ensure_numeric(x)
   1535     elif not (is_float(x) or is_integer(x) or is_complex(x)):
   1536         try:
-> 1537             x = float(x)
   1538         except ValueError:
   1539             # e.g. "1+1j" or "foo"

~/anaconda3/envs/conda_env/lib/python3.8/site-packages/uncertainties/core.py in␣
 ↪raise_error(self)
   2700     for coercion_type in ('complex', 'int', 'long', 'float'):
   2701         def raise_error(self):
-> 2702             raise TypeError("can't convert an affine function (%s)"
   2703                             ' to %s; use x.nominal_value'
   2704                             # In case AffineScalarFunc is sub-classed:

TypeError: can't convert an affine function (<class 'uncertainties.core.
 ↪AffineScalarFunc'>) to float; use x.nominal_value
```

[253]:
```
v_per_b = ufloat(.04051960131985295, 0.00035729666185760475)
```

$$n = \frac{BI}{V_H e d}$$

$$n = \frac{B}{V_H}\frac{I}{ed}$$

$$n = \frac{dB}{dV_H}\frac{I}{ed}$$

[254]:
```
I_s_mean, I_s_unc = h_v.Is.mean(), h_v.Is.std()
```

[255]:
```
I_s = ufloat(I_s_mean, I_s_unc)
```

[256]:
```
e = 1.60217662e-19 # coulombs
```

[258]:
```
d = ufloat(silicon_thickness, .02)*1e-3
print_unc(d)
```

```
0.00054 +- 0.00002
```

[258]:
```
(0.00054, 2e-05, 5)
```

[259]:
```
v_per_b, I_s, e, d
```

[259]:
```
(0.04051960131985295+/-0.00035729666185760475,
 0.007912761904761903+/-2.618569220376792e-05,
 1.60217662e-19,
 0.00054+/-2e-05)
```

[260]:
```
n = v_per_b**-1 * I_s_mean / (e*d)
n # per m^3
```

[260]:
```
2.2571414938221858e+21+/-8.593447778863827e+19
```

[234]:
```
from uncertainties import ufloat
```

[261]:
```
rho = ufloat(fourwire.rho.mean(), fourwire.rho.std())
rho
```

[261]:
```
0.08749724057934033+/-0.01611712651379376
```

$$\rho = \frac{1}{n\, e\, \mu}$$

$$\mu = \frac{1}{n\, e\, \rho}$$

[262]:
```
mu = 1/(n*e*rho) # m^2/(Vs)
mu
```

[262]:
```
0.031603593614801424+/-0.005944475884310966
```

[263]:
```
mu * 100*100
```

[263]:
```
316.03593614801423+/-59.44475884310966
```

[224]:
```
rho * 100, 'cm'
```

[224]:
```
8.749724057934033+/-1.611712651379376
```

[232]:
```
print(n*1e-6)
```

```
(1.348+/-0.012)e+14
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: