Data Analysis Langmuir Probe

October 4, 2018

1 Data Analysis

1.1 Langmuir Probe

14th September, 2018
Kitty Harris and Idriss Kacou

```
In [2]: import pandas as pd
        import scipy
        from scipy import optimize
        import scipy.constants
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        from bokeh.plotting import figure, output_notebook, show
        import math
```

```
In [76]: #Now we're gonna define some stuff.
         df = pd.read_csv('340mTorr Data Master.csv')
         x=df.iloc[:,0].values
         y=df.iloc[:,1].values
```

```
In [58]: #I took this from Experimental Setup.
         #It has been modified it slightly to fit the defined variables.


         # Specifying that we want the plot to be displayed within the notebook
         output_notebook(hide_banner=True)

         # create a new plot with a title and axis labels.
         #DON'T FORGET TO CHANGE THE TITLE IF YOU CHANGED THE DATA
         p = figure(title="592mTorr 1020V Master Dataset", x_axis_label='Voltage (V)', y_axis_

         # specify the data to plot and define the line type - here we will use a circle for t
         p.circle(x, y, fill_color="white", size=8)

         # show the results
         show(p)
```

```
In [77]: #Now we want the logY plot.
         #DON'T FORGET TO CHANGE THE TITLE IF YOU CHANGED THE DATA
         p = figure(title="592mTorr 1020V Master Dataset logY", x_axis_label='Voltage [V]', y_a

         # specify the data to plot and define the line type - here we will use a circle for t
         p.circle(x, y, fill_color="white", size=8)

         # show the results
         show(p)
```

```
In [24]: #Now we want to show multiple data on the same plot.
         #First we need to define more data.
         af = pd.read_csv("340mTorr Data Master.csv")
         bf = pd.read_csv("Data/592mTorr/KHIK592mtorr1400V.csv")
         cf = pd.read_csv("585mtorr Data Master.csv")
         #df = pd.read_csv("Data/592mTorr/KHIK592mtorr1500V.csv")
         #copy and paste this, changing the first letter until you have as many as you need.
         #Now we define X and Y
         xa=af.iloc[:,0].values
         ya=af.iloc[:,1].values
         xb=bf.iloc[:,0].values
         yb=bf.iloc[:,1].values
         xc=cf.iloc[:,0].values
         yc=cf.iloc[:,1].values
         #xd=df.iloc[:,0].values
         #yd=df.iloc[:,1].values
         #etc.
```

```
In [26]: #Now we can plot our graph.
         #This is mainly the same as above.

         #DON'T FORGET TO CHANGE THE TITLE IF YOU CHANGED THE DATA
         p = figure(title="~1400V", x_axis_label='Voltage [V]', y_axis_label='logCurrent (log[

         # REMEMBER TO UPDATE THE LEGEND ACCORDINGLY
         p.circle(xa, ya, fill_color="blue", size=4, legend="340mTorr")
         p.square(xb, yb, fill_color="green", size=4, legend="592mTorr")
         p.triangle(xc, yc, fill_color="red", size=8, legend="585mTorr")
         #p.diamond(xd, yd, fill_color="yellow", size=8, legend="1500V")

         p.legend.location = "top_left"

         show(p)
```

```
In [48]: #This is our setup for finding our curve fits for ion- and electron-dominated regions
         #ions = pd.read_csv("585mTorr Ions Only.csv")
         electrons = pd.read_csv("592mTorr1020V Electrons Only.csv")
         #this deletes stored arrays so that we can redefine smaller ones if necessary.
         ye = []
         xe = []
         yi = []
         xe = []

         #these need to be cleaned. I'm not sure why, but the code below should clean them.
         #yidirty = ions.iloc[:,1].values
         yedirty = electrons.iloc[:,1].values
         #xidirty = ions.iloc[:,0].values
         xedirty = electrons.iloc[:,0].values
         ye = [x for x in yedirty if str(x) != 'nan']
         xe = [x for x in xedirty if str(x) != 'nan']
         #yi = [x for x in yidirty if str(x) != 'nan']
         #xi = [x for x in xidirty if str(x) != 'nan']

         #logi = np.zeros(len(xi))
         loge = np.zeros(len(xe))
         i=0
         #while i < len(xi):
          #   logi[i] = math.log(yi[i])
           #  i += 1

         i=0
         while i < len(xe):
             loge[i] = math.log(ye[i])
             i += 1
```

```
In [49]: #here we get the parameters for our curve fit for Ions.
         print("Ions:")
         np.polyfit(xi, logi, 1, rcond=None, full=False, w=None, cov=False)
```

```
Ions:



Out[49]: array([ -0.03818926, -12.57272928])
```

```
In [50]: #and here we get parameters for Electrons.
         print("Electrons:")
         np.polyfit(xe, loge, 1, rcond=None, full=False, w=None, cov=False)
```

```
Electrons:



Out[50]: array([0.03069946, 1.07237903])
```

```
In [51]: #Now we define arrays...
         emb = np.polyfit(xe, loge, 1)
         imb = np.polyfit(xi, logi, 1)
```

```
In [236]: #We want to define some arrays so that we can plot our lines. Since they're perfectl
          fix = [-210,-200]
          fiy = [0,0]
          fex = [-180,-130]
          fey = [0,0]
          #we use e below because our graph takes the log of our y-axis
          i = 0
          while i < 2:
              fiy[i] = math.e**(imb[0] * fix[i] + imb[1])
              i += 1

          i = 0
          while i < 2:
              fey[i] = math.e**(emb[0] * fex[i] + emb[1])
              i += 1
```

```
In [238]: #Plot with curve fits
          p = figure(title="340 mTorr logY", x_axis_label='Voltage [V]', y_axis_label='logCurr

          p.circle(xe, ye, fill_color="white", size=8, legend="Electron-Dominated")
          p.square(xi, yi, fill_color="orange", size=8, legend="Ion-Dominated")
          p.line(fex, fey, line_color="blue", legend="m=0.05167206, b=-1.71136101")
          p.line(fix, fiy, line_color="red", legend="m=0.2357951, b=33.99766579")

          show(p)
```

```
In [250]: #Here we can plot multiple data with curve fits.

          #DON'T FORGET TO CHANGE THE TITLE IF YOU CHANGED THE DATA
          p = figure(title="~1000V", x_axis_label='Voltage [V]', y_axis_label='logCurrent (log

          # REMEMBER TO UPDATE THE LEGEND ACCORDINGLY
          p.circle(xa, ya, fill_color="blue", size=4, legend="340mTorr")
          p.square(xb, yb, fill_color="green", size=4, legend="592mTorr")
          p.triangle(xc, yc, fill_color="red", size=8, legend="585mTorr")
          #p.diamond(xd, yd, fill_color="yellow", size=8, legend="1500V")
          p.line(fex, fey, line_color="yellow", legend="m=0.05167206, b=-1.71136101", line_wid
          p.line(fix, fiy, line_color="red", legend="m=0.2357951, b=33.99766579")
          p.legend.location="top_left"
          show(p)
```

```
In [52]: print(emb[0])

0.03069945924921889
```