Langmuir Probe

James Amidei, Zach Stedman, Max Markgraf

RUN CELLS IN ORDER

In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import scipy.stats as stats

def is_float(string):
    try:
        float(string)
        return True
    except ValueError:
        return False

data1 = np.genfromtxt('data\data4_26_24\data_good\p420v1200s3.csv', delim
data2 = np.genfromtxt('data\data4_26_24\data_good\p415v1300s3.csv', delim
data3 = np.genfromtxt('data\data4_26_24\data_good\p417v1400s3.csv', delim

v_data_1 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_1 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_1 = np.asarray(i_data_1, dtype=np.float64)*0.001
v_data_2 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_2 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_2 = np.asarray(i_data_1, dtype=np.float64)*0.001
v_data_3 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_3 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_3 = np.asarray(i_data_1, dtype=np.float64)*0.001

# constants
k = 1.380649e-23 # Joules/Kelvin
e = 1.602176634e-19 # Coulombs

# linear fit function
def func(x, a, b):
    return a*x + b
```

Important Note:

----From what it seems like from this write up (https://download.tek.com/document/LowCurtMsmntsAppNote.pdf (https://download.tek.com/document/LowCurtMsmntsAppNote.pdf)), we can estimate the error in the current to be roughly 0.01 of the current measurement. This has been calculated from our data above.----

I just used the variance from the matrix given to use through pcov in curve_fit to calculate the

$$I_e(V_P) = I_{es} e^{\frac{-q_e(V_P - V_B)}{kT_e}}$$

$$ln(I_e) = \frac{q_e V_B}{kT_e} - \frac{q_e V_P}{kT_e} + ln(V_{es}) = \frac{q_e V_B}{kT_e} + b$$

$$I_{es} = \frac{1}{4} e n_e v_e A_{probe} = e n_e \left(\frac{kT_e}{2\pi m_e} \right)^{1/2} A_{probe}$$

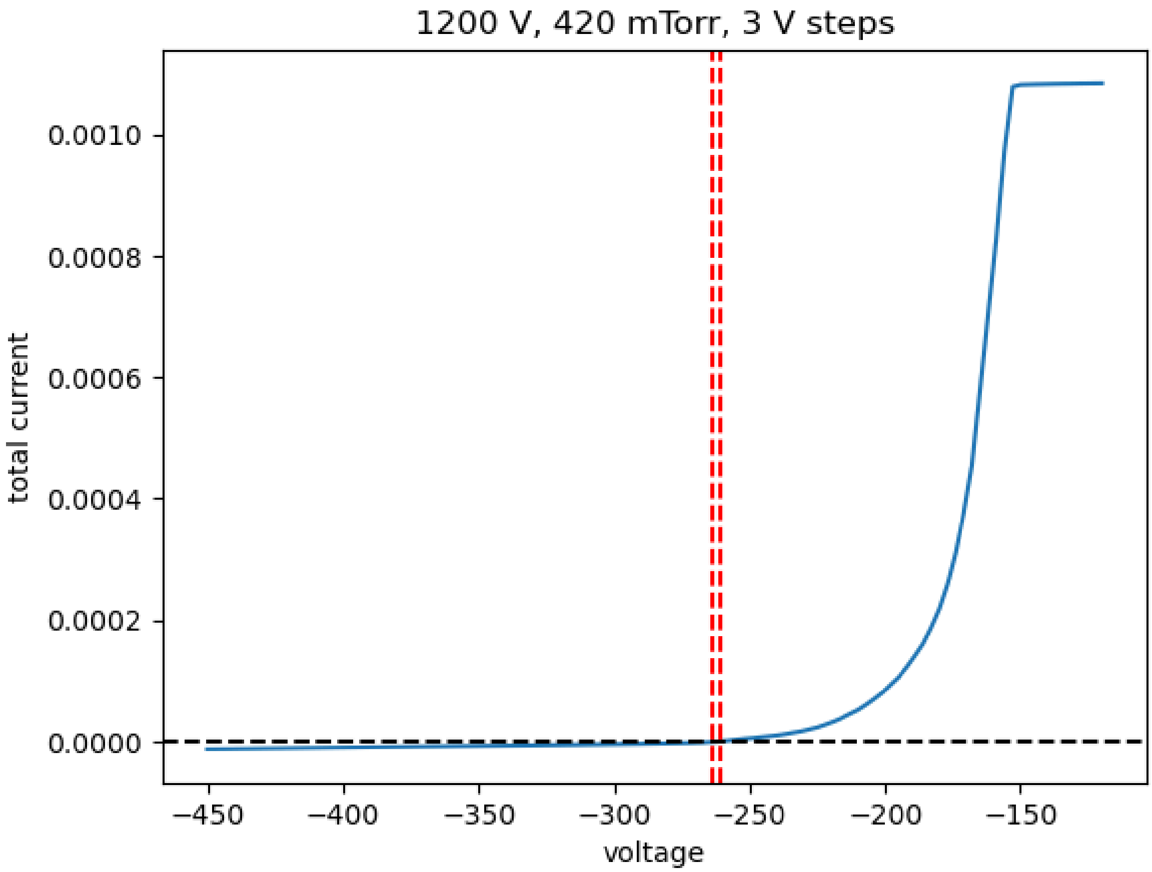
$$n_e = e I_{es} \left(\frac{kT_e}{2\pi m_e} \right)^{1/2} A_{probe}$$

$$f_c = 9\sqrt{n_{e,max}}$$

1200 V, 420 mTorr, 3 V steps

In [2]:

```
# 1200 - initial plot, non-fit
# from this we can find the floating potential V_f
# and the ion and electron saturation currents (I_is and I_es).
# The floating potential is the voltage where
# the total current is equal to zero.
# The ion saturation current is the negative current given
# by the first plateau. The electron saturation current
# is given by the second plateau after the exponential transition.

V_f_1_max = v_data_1[62]
V_f_1_min = v_data_1[63]

plt.figure()
plt.plot(v_data_1, i_data_1)
plt.axhline(y=0, color='black', linestyle='--')
plt.axvline(x=V_f_1_max, color='red', linestyle='--')
plt.axvline(x=V_f_1_min, color='red', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1200 V, 420 mTorr, 3 V steps')
plt.show()

print(f'V_f_max = {V_f_1_max} V')
print(f'V_f_min = {V_f_1_min} V')
print(f'V_f = {(V_f_1_max + V_f_1_min)/2} V +/- {abs(V_f_1_max - V_f_1_mi
print()
print(f'I_is_min = {min(i_data_1)} A')
print(f'I_is_max = {i_data_1[10]} A')
print(f'I_is = {(i_data_1[10] + min(i_data_1))/2:0.11f} A +/- {abs(i_data
print()
print(f'I_es_min = {i_data_1[100]} A')
print(f'I_es_max = {max(i_data_1)} A')
print(f'I_es = {(max(i_data_1) + i_data_1[100])/2:0.9f} A +/- {abs(max(i_
```

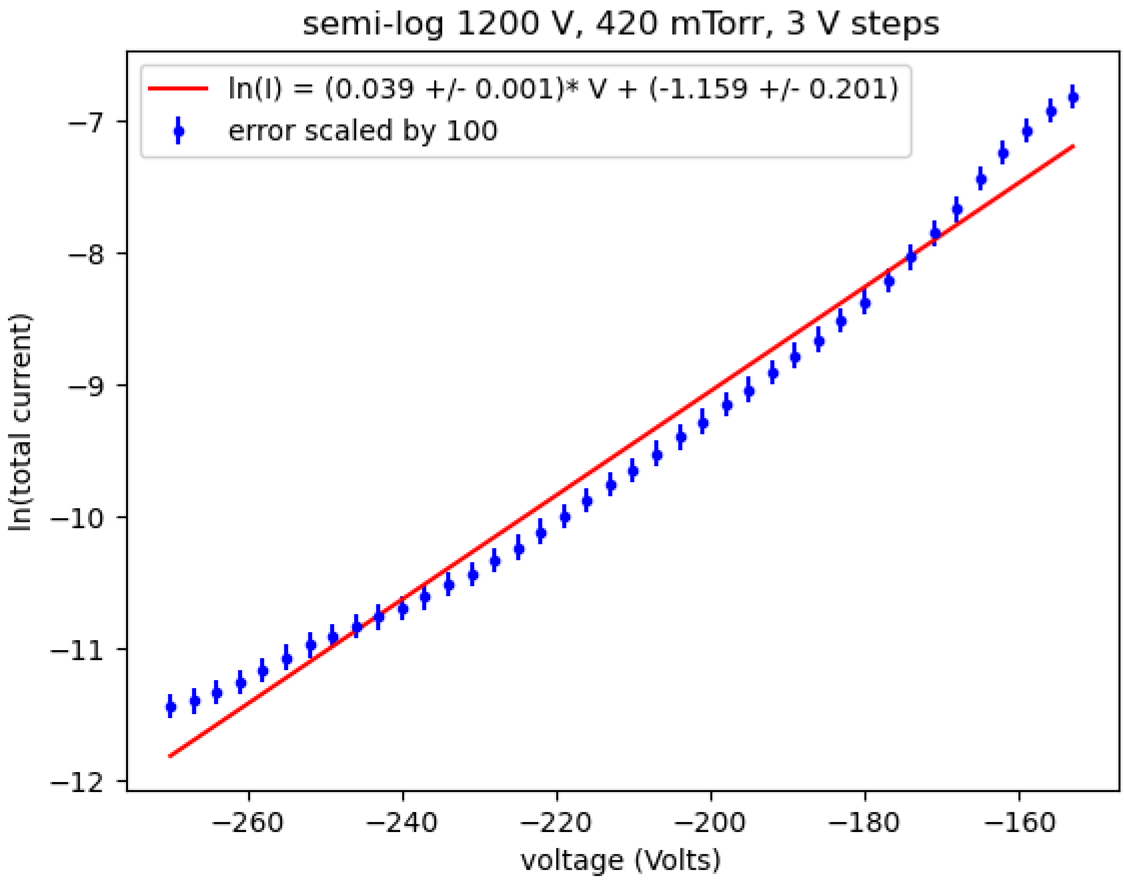



```
V_f_max = -264.0 V
V_f_min = -261.0 V
V_f = -262.5 V +/- 1.5 V

I_is_min = -1.284397e-05 A
I_is_max = -1.131792e-05 A
I_is = -0.00001208095 A +/- 0.00000076303 A

I_es_min = 0.001081897 A
I_es_max = 0.001084314 A
I_es = 0.001083106 A +/- 0.000001209 A
```

In [3]:

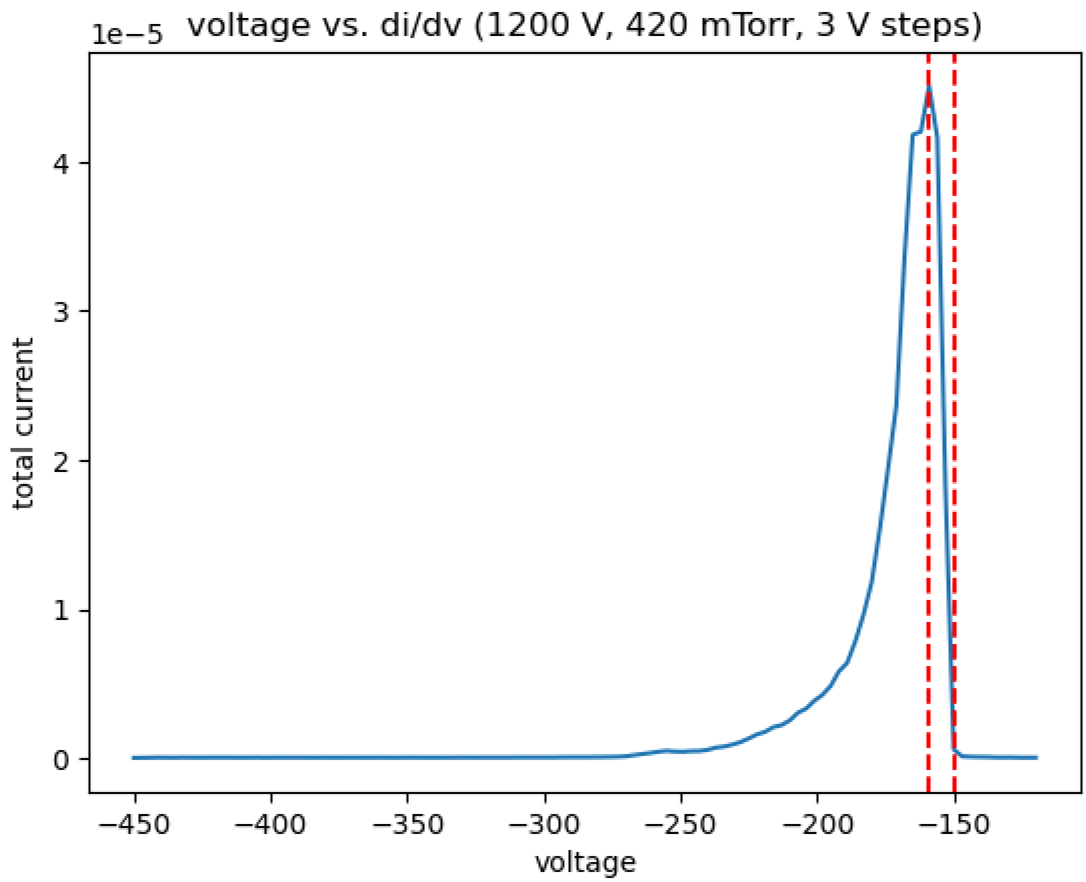
```
I_is_1 = -1.284397e-05 # Ion saturation current for data1

I_data_1 = [x - I_is_1 for x in i_data_1]

I_data_1_filtered = I_data_1[60:100]
V_data_1 = v_data_1[60:100]
logI_data_1 = np.log(I_data_1_filtered)

popt, pcov = curve_fit(func, V_data_1, logI_data_1, p0=[1.0, 1.0])

V_data_1arr = np.asarray(V_data_1, dtype=np.float64)
logI_fit_1 = func(V_data_1arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))
fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

plt.figure()
plt.errorbar(V_data_1, logI_data_1, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_1arr, logI_fit_1, color='r', label= f'{fit}')
plt.xlabel('voltage (Volts)')
plt.ylabel('ln(total current)')
plt.title('semi-log 1200 V, 420 mTorr, 3 V steps')
plt.legend()
plt.show()

T_e1 = e/(k*a)
sigma_T_e1 = abs(sigma_a*(e/(k*a**2)))

print(f'Electron temperature (K): {T_e1:.0f} K +/- {sigma_T_e1:.0f} K')
print(f'Electron temperature (eV): {T_e1/11606:.2f} eV +/- {sigma_T_e1/11
```

```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.159 +/- 0.20
1)
```




```
Electron temperature (K): 294038 K +/- 6999 K
Electron temperature (eV): 25.34 eV +/- 0.60 eV
```

In [4]:
```
# plasma potential
# this is the voltage where we hit the peak plateau
# we can find this by plotting the derivative of the current
# and find the value of the voltage around the maximum and the
# next zero value

didv = np.gradient(i_data_1, v_data_1)
didv_max = max(didv)
#mask = np.abs(didv - didv_max).argmin()
max_index = np.argmax(didv)
min_index = 100
V_P_max = v_data_1[max_index]
V_P_min = v_data_1[min_index]

plt.figure()
plt.plot(v_data_1, didv)
#plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('voltage vs. di/dv (1200 V, 420 mTorr, 3 V steps)')
plt.axvline(x=V_P_max, color='red', linestyle='--')
plt.axvline(x=V_P_min, color='red', linestyle='--')
#plt.legend()
plt.show()

print(f'V_P_max = {v_data_1[max_index]} V')
print(f'V_P_min = {v_data_1[min_index]} V')
print(f'V_P = {(v_data_1[max_index] + v_data_1[min_index])/2} V +/- {abs(
```



```
V_P_max = -159.0 V
V_P_min = -150.0 V
V_P = -154.5 V +/- 4.5 V
```

In [5]:

```
# calculations for the floating potential.
# This is when the total current is zero
# and V_B < V_P.

I_is_min = min(i_data_1)
I_is_max = i_data_1[10]

I_es_min = i_data_1[100]
I_es_max = max(i_data_1)

I_is = (I_is_max + I_is_min)/2
I_es = (I_es_max + I_es_min)/2

sigma_Iis = abs(I_is_max - I_is_min)/2
sigma_Ies = abs(I_es_max - I_es_min)/2

V_P = (v_data_1[max_index] + v_data_1[min_index])/2
sigma_V_P = abs(v_data_1[max_index] - v_data_1[min_index])/2

sigma_V_f_calc = np.sqrt( (sigma_a*(1/a**2)*np.log(-I_is/I_es))**2 + (sig

V_f_calc = (1/a)*np.log(-I_is/I_es) + V_P
print(f'The calculated floating potential using the electron temperature:
```

```
The calculated floating potential using the electron temperature: -268.42 K
+/- 5.49 K
```

In [6]:
```
# abandoned for now, may come back later

# electron density calculation
# needs more work


d = 3.0e-3 # mm (probe diameter estimate from Merlino)
A = 2*np.pi*(d/2)**2 # area of probe

m_e = 9.11e-31 # kg (electron mass)

v_e = np.sqrt(8*k*T_e1/(np.pi*m_e)) # electron speed
sigma_v_e = np.sqrt((sigma_T_e1*np.sqrt(2*k/(np.pi*m_e*T_e1)))**2)

n_e = (4*I_es)/(e*A*v_e) # electron density

sigma_n_e = np.sqrt((sigma_Ies*4/(e*A*v_e)**2 + (sigma_v_e*(2*I_es/(e*A*v

P = 420*(1/1000)*(133.3) # mTorr to Pascals
R = 8.3145 #mol^-1 K^-1

molecular_density = P/(R*T_e1)

print(f'electron density: n_e = {round(n_e)} +/- {round(sigma_n_e)}')
print(f'Total molecualr density: n/V = {molecular_density}')
```

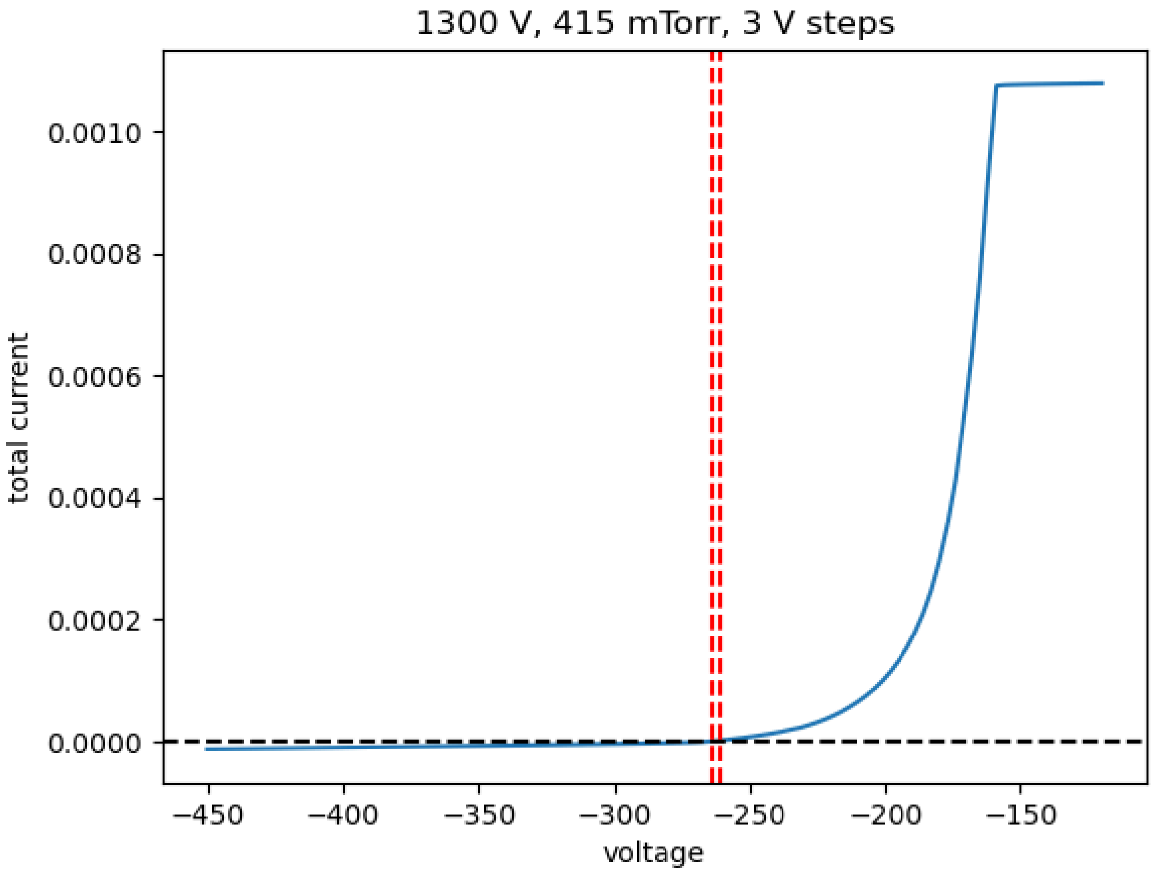
```
electron density: n_e = 567810752146647 +/- 288174825282869
Total molecualr density: n/V = 2.2900217217644483e-05
```

In [ ]:
```

```

1300 V, 415 mTorr, 3 V steps

In [7]:

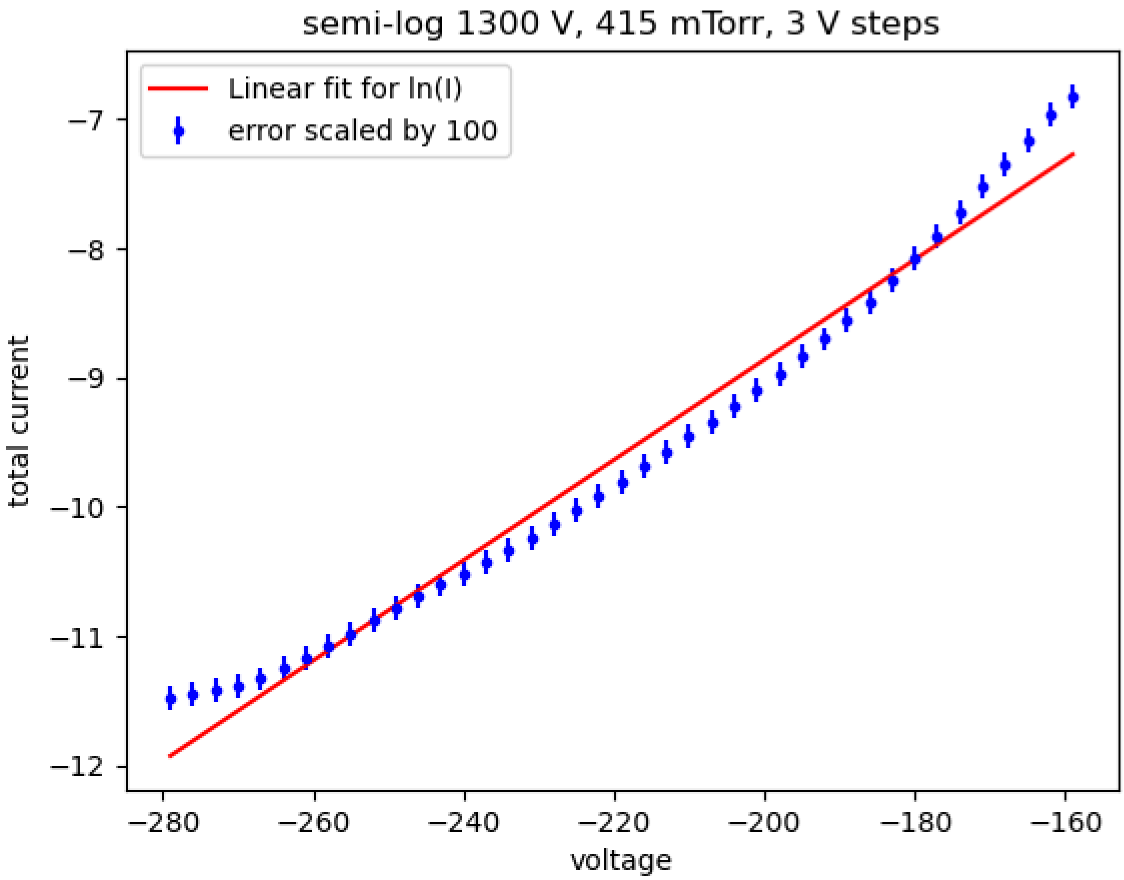
```
# 1300 - initial plot, non-fit

V_f_2_max = v_data_2[62]
V_f_2_min = v_data_2[63]

plt.figure()
plt.plot(v_data_2, i_data_2)
plt.axhline(y=0, color='black', linestyle='--')
plt.axvline(x=V_f_2_min, color='red', linestyle='--')
plt.axvline(x=V_f_2_max, color='red', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1300 V, 415 mTorr, 3 V steps')
plt.show()

print(f'V_f_max = {V_f_2_max} V')
print(f'V_f_min = {V_f_2_min} V')
print(f'V_f = {(V_f_2_max + V_f_2_min)/2} V +/- {abs(V_f_2_max - V_f_2_mi
print()
print(f'I_is_min = {min(i_data_2)} A')
print(f'I_is_max = {i_data_2[10]} A')
print(f'I_is = {(i_data_2[10] + min(i_data_2))/2:0.11f} A +/- {abs(i_data
print()
print(f'I_es_min = {i_data_2[100]} A')
print(f'I_es_max = {max(i_data_2)} A')
print(f'I_es = {(max(i_data_2) + i_data_2[100])/2:0.9f} A +/- {abs(max(i_

```




```
V_f_max = -264.0 V
V_f_min = -261.0 V
V_f = -262.5 V +/- 1.5 V

I_is_min = -1.315807e-05 A
I_is_max = -1.166607e-05 A
I_is = -0.00001241207 A +/- 0.00000074600 A

I_es_min = 0.00107665 A
I_es_max = 0.001078651 A
I_es = 0.001077651 A +/- 0.000001000 A
```

In [8]:
```
I_is_2 = -1.315807e-05
#print(i_data_2)

I_data_2 = [x - I_is_2 for x in i_data_2]

I_data_2_filtered = I_data_2[57:98]
V_data_2 = v_data_2[57:98]
logI_data_2 = np.log(I_data_2_filtered)

popt, pcov = curve_fit(func, V_data_2, logI_data_2, p0=[1.0,1.0])

V_data_2arr = np.asarray(V_data_2, dtype=np.float64)
logI_fit_2 = func(V_data_2arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))

fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

T_e2 = e/(k*a)
sigma_T_e2 = abs(sigma_a*(e/(k*a**2)))

plt.figure()
#plt.plot(V_data_2, logI_data_2)
plt.errorbar(V_data_2, logI_data_2, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_2arr, logI_fit_2, color='r', label= 'Linear fit for ln(I)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('semi-log 1300 V, 415 mTorr, 3 V steps')
plt.legend()
plt.show()

print(f'Electron temperature: {T_e2:.0f} K +/- {sigma_T_e2:.0f} K')
print(f'Electron temperature (eV): {T_e2/11606:.2f} eV +/- {sigma_T_e2/11
```

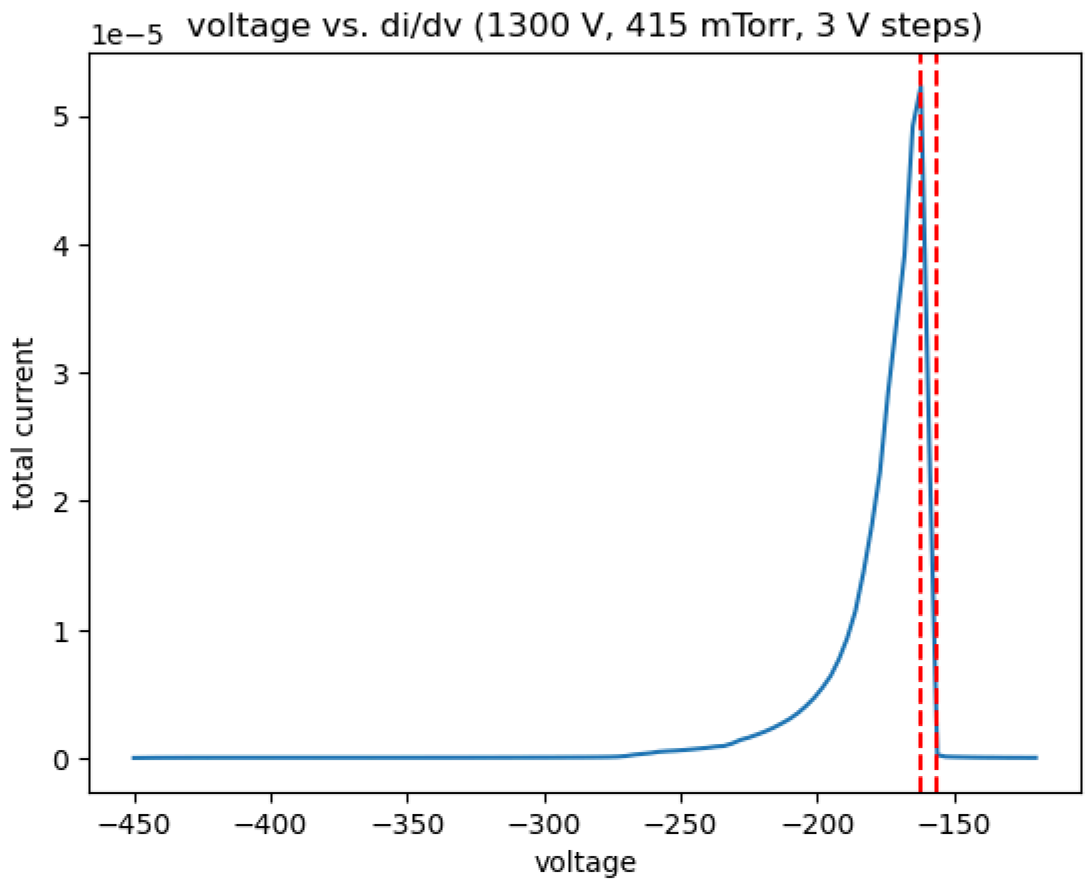
```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.111 +/- 0.20
4)
```




```
Electron temperature: 299393 K +/- 7114 K
Electron temperature (eV): 25.80 eV +/- 0.61 eV
```

In [9]:

```
didv = np.gradient(i_data_2, v_data_2)
didv_max = max(didv)
max_index = np.argmax(didv)
min_index = 98
V_P_max = v_data_2[max_index]
V_P_min = v_data_2[min_index]

plt.figure()
plt.plot(v_data_2, didv)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('voltage vs. di/dv (1300 V, 415 mTorr, 3 V steps)')
plt.axvline(x=V_P_max, color='red', linestyle='--')
plt.axvline(x=V_P_min, color='red', linestyle='--')
plt.show()

print(f'V_P_max = {v_data_2[max_index]} V')
print(f'V_P_min = {v_data_2[min_index]} V')
print(f'V_P = {(v_data_2[max_index] + v_data_2[min_index])/2} V +/- {abs(
```




```
V_P_max = -162.0 V
V_P_min = -156.0 V
V_P = -159.0 V +/- 3.0 V
```

In [10]:

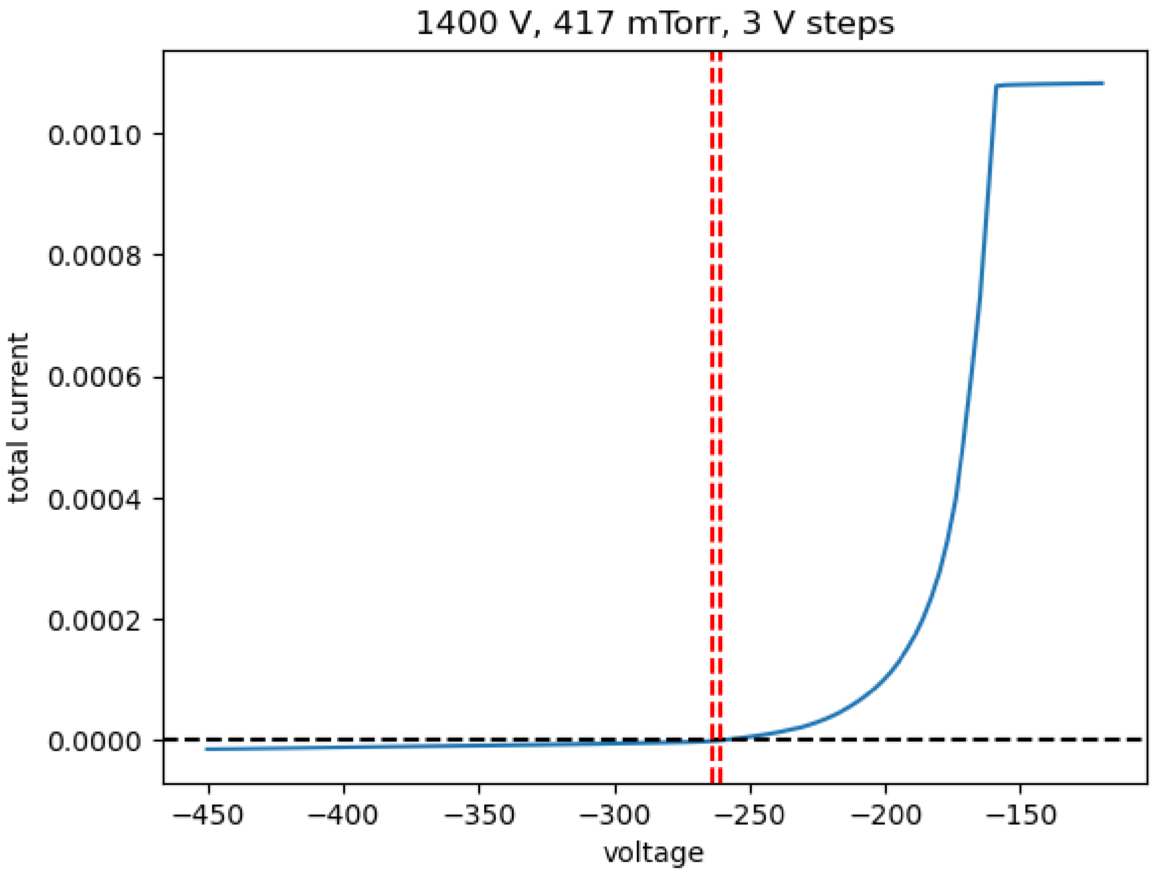
```
# calculations for the floating potential.

I_is_min = min(i_data_2)
I_is_max = i_data_2[10]

I_es_min = i_data_2[100]
I_es_max = max(i_data_2)

I_is = (I_is_max + I_is_min)/2
I_es = (I_es_max + I_es_min)/2

sigma_Iis = abs(I_is_max - I_is_min)/2
sigma_Ies = abs(I_es_max - I_es_min)/2

V_P = (v_data_2[max_index] + v_data_2[min_index])/2
sigma_V_P = abs(v_data_2[max_index] - v_data_1[min_index])/2

sigma_V_f_calc = np.sqrt( (sigma_a*(1/a**2)*np.log(-I_is/I_es))**2 + (sig

V_f_calc = (1/a)*np.log(-I_is/I_es) + V_P
print(f'The calculated floating potential using the electron temperature:
```

```
The calculated floating potential using the electron temperature: -274.17 K
+/- 4.35 K
```

1400 V, 417 mTorr, 3 V steps

In [11]:

```
# 1200 - initial plot, non-fit

V_f_3_min = v_data_3[62]
V_f_3_max = v_data_3[63]

plt.figure()
plt.plot(v_data_3, i_data_3)
plt.axhline(y=0, color='black', linestyle='--')
plt.axvline(x=V_f_3_min, color='red', linestyle='--')
plt.axvline(x=V_f_3_max, color='red', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1400 V, 417 mTorr, 3 V steps')
plt.show()

print(f'V_f_max = {V_f_3_max} V')
print(f'V_f_min = {V_f_3_min} V')
print(f'V_f = {(V_f_3_max + V_f_3_min)/2} V +/- {abs(V_f_3_max - V_f_3_mi
print()
print(f'I_is_min = {min(i_data_3)} A')
print(f'I_is_max = {i_data_3[10]} A')
print(f'I_is = {(i_data_3[10] + min(i_data_3))/2:0.11f} A +/- {abs(i_data
print()
print(f'I_es_min = {i_data_3[100]} A')
print(f'I_es_max = {max(i_data_3)} A')
print(f'I_es = {(max(i_data_3) + i_data_3[100])/2:0.9f} A +/- {abs(max(i_

```

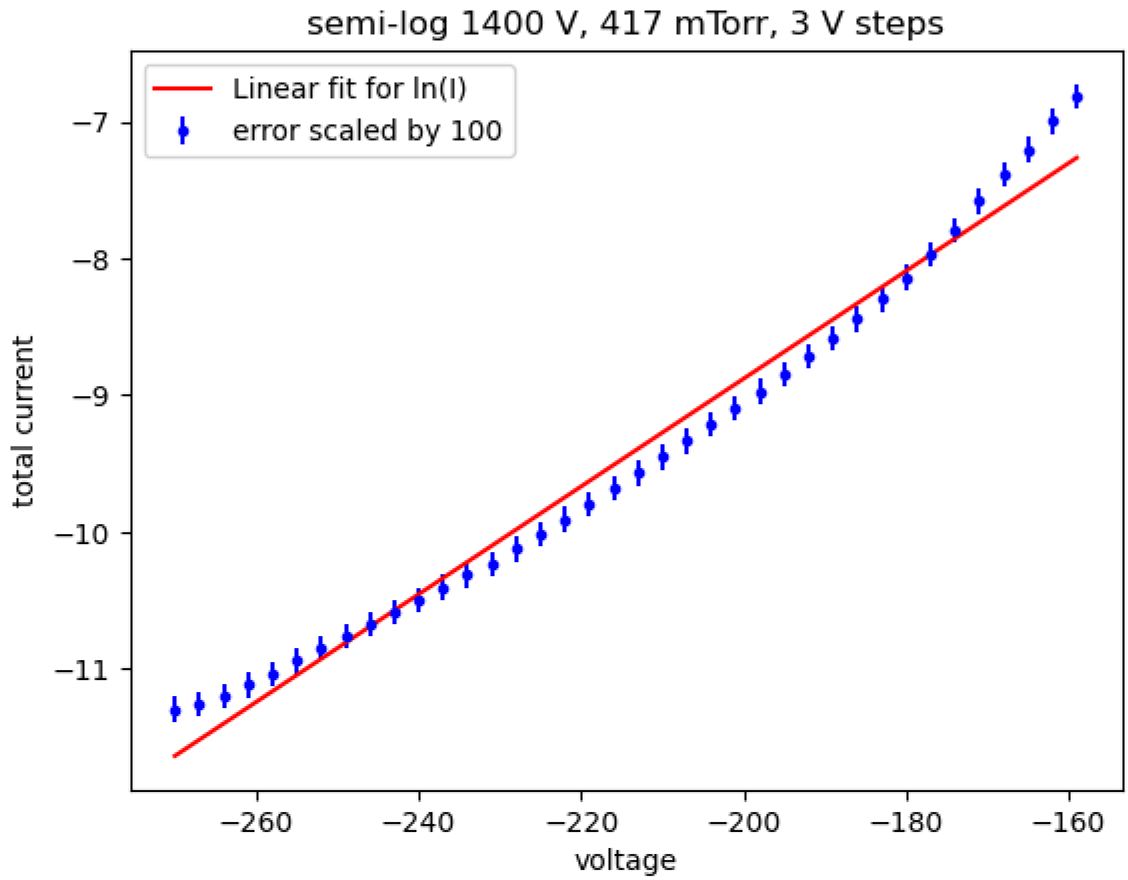



```
V_f_max = -261.0 V
V_f_min = -264.0 V
V_f = -262.5 V +/- 1.5 V

I_is_min = -1.478699e-05 A
I_is_max = -1.307172e-05 A
I_is = -0.00001392935 A +/- 0.00000085763 A

I_es_min = 0.001080552 A
I_es_max = 0.001082654 A
I_es = 0.001081603 A +/- 0.000001051 A
```

In [12]:

```
I_is_3 = -1.478699e-05

I_data_3 = [x - I_is_3 for x in i_data_3]

I_data_3_filtered = I_data_3[60:98]
V_data_3 = v_data_3[60:98]
logI_data_3 = np.log(I_data_3_filtered)

popt, pcov = curve_fit(func, V_data_3, logI_data_3, p0=[1.0,1.0])

V_data_3arr = np.asarray(V_data_3, dtype=np.float64)
logI_fit_3 = func(V_data_3arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))
fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

T_e3 = e/(k*a)
sigma_T_e3 = abs(sigma_a*(e/(k*a**2)))


plt.figure()
#plt.plot(V_data_3, logI_data_3)
plt.errorbar(V_data_3, logI_data_3, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_3arr, logI_fit_3, color='r', label= 'Linear fit for ln(I)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('semi-log 1400 V, 417 mTorr, 3 V steps')
plt.legend()
plt.show()

print(f'Electron temperature: {T_e3:.0f} K +/- {sigma_T_e3:.0f} K')
print(f'Electron temperature (eV): {T_e3/11606:.2f} eV +/- {sigma_T_e3/11
```

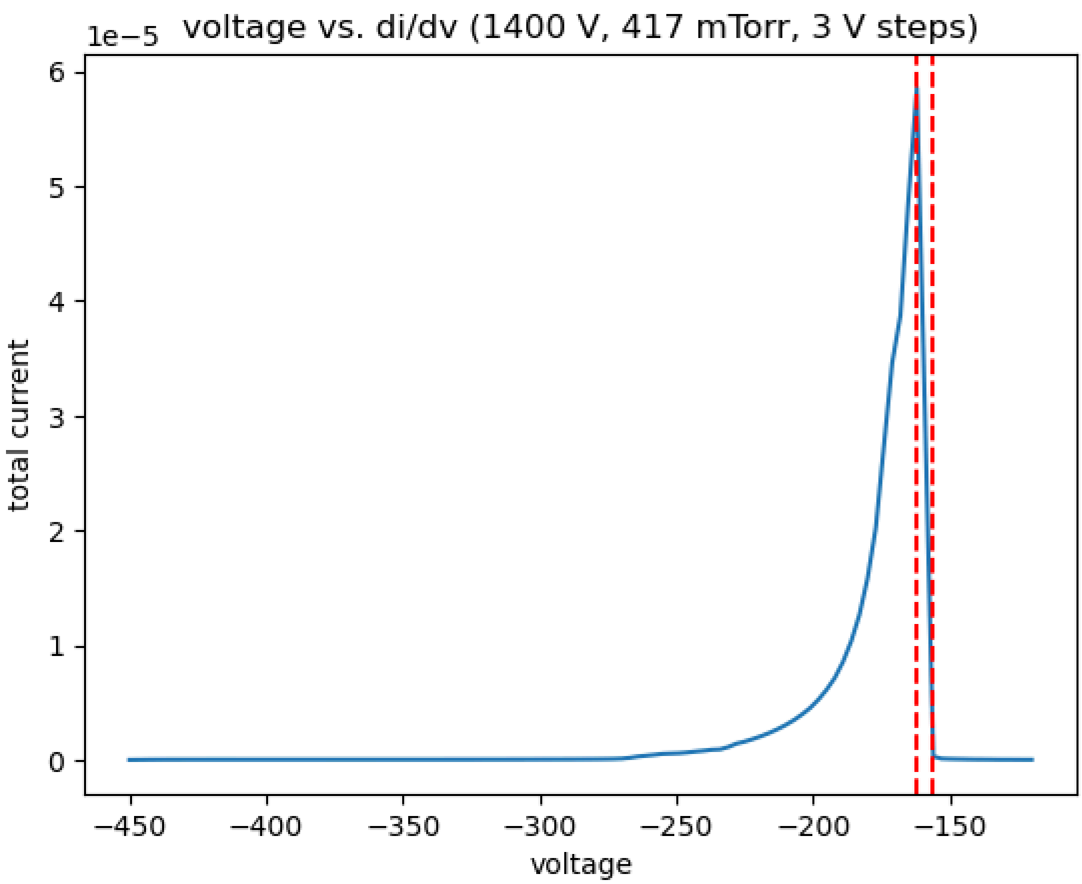
```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.001 +/- 0.19
9)
```




```
Electron temperature: 294587 K +/- 6854 K
Electron temperature (eV): 25.38 eV +/- 0.59 eV
```

In [13]:

```python
didv = np.gradient(i_data_3, v_data_3)
didv_max = max(didv)
#mask = np.abs(didv - didv_max).argmin()
max_index = np.argmax(didv)
min_index = 98
V_P_max = v_data_3[max_index]
V_P_min = v_data_3[min_index]

plt.figure()
plt.plot(v_data_3, didv)
#plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('voltage vs. di/dv (1400 V, 417 mTorr, 3 V steps)')
plt.axvline(x=V_P_max, color='red', linestyle='--')
plt.axvline(x=V_P_min, color='red', linestyle='--')
#plt.legend()
plt.show()

print(f'V_P_max = {v_data_3[max_index]} V')
print(f'V_P_min = {v_data_3[min_index]} V')
print(f'V_P = {(v_data_3[max_index] + v_data_3[min_index])/2} V +/- {abs(
```




```
V_P_max = -162.0 V
V_P_min = -156.0 V
V_P = -159.0 V +/- 3.0 V
```

In [14]:

```
# calculations for the floating potential.

I_is_min = min(i_data_2)
I_is_max = i_data_2[10]

I_es_min = i_data_2[100]
I_es_max = max(i_data_2)

I_is = (I_is_max + I_is_min)/2
I_es = (I_es_max + I_es_min)/2

sigma_Iis = abs(I_is_max - I_is_min)/2
sigma_Ies = abs(I_es_max - I_es_min)/2

V_P = (v_data_2[max_index] + v_data_2[min_index])/2
sigma_V_P = abs(v_data_2[max_index] - v_data_1[min_index])/2

sigma_V_f_calc = np.sqrt( (sigma_a*(1/a**2)*np.log(-I_is/I_es))**2 + (sig

V_f_calc = (1/a)*np.log(-I_is/I_es) + V_P
print(f'The calculated floating potential using the electron temperature:
```

```
The calculated floating potential using the electron temperature: -272.32 K
+/- 4.28 K
```

In [ ]:

```

```