```
 1 import numpy as np
 2 import matplotlib.pyplot as plt
 3 from sklearn.linear_model import LinearRegression
 4
 5 #linefit and quadratic fit for troughs
 6 n_min = np.array([1, 2, 3]).reshape((-1, 1))
 7 tr = np.array([21.1, 38.1, 57])
 8 y_min = []
 9 n_max = np.array([1, 2, 3, 4]).reshape((-1, 1))
10 pe = np.array([13.8, 30.8, 48.7, 68.7])
11 y_max = []
12
13 reg = LinearRegression().fit(n_min, tr)
14 reg.score(n_min,tr)
15 xfit = np.linspace(0, 4, 100)
16 yfit = reg.predict(xfit[:, np.newaxis])
17
18 f = plt.figure(1, figsize=(12,8))
19 plt.grid()
20 plt.title('Neon Franck-Hertz Analysis', fontsize=14)
21 plt.xlabel('Peak/Trough Number', fontsize=14)
22 plt.ylabel('Peak/Trough Position (V)', fontsize=14)
23 plt.scatter(n_min,tr, color = "blue")
24 plt.plot(xfit, yfit, label = 'Trough Linear Fit')
25
26
27 reg = LinearRegression().fit(n_max, pe)
28 reg.score(n_max,pe)
29 xfit = np.linspace(0, 4, 100)
30 yfit = reg.predict(xfit[:, np.newaxis])
31 plt.scatter(n_max,pe, color = "black")
32 plt.plot(xfit, yfit, label = "Peak Linear Fit", color
    = "black")
33
34 model = np.poly1d(np.polyfit(n_min.flatten(), tr.
   flatten(), 2))
35 polyline = np.linspace(0, 4, 100)
36 plt.plot(polyline, model(polyline), label = "Trough
   Quadratic Fit", color = "lightblue")
37
38 model2 = np.poly1d(np.polyfit(n_max.flatten(), pe.
   flatten(), 2))
39 polyline = np.linspace(0, 4, 100)
40 plt.plot(polyline, model2(polyline), label = 'Peak
```

```
40 Quadratic Fit', color = "gray")
41 plt.legend()
42
43 n_min = np.array([1, 2]).reshape((-1, 1))
44 tr = [21.1, 38.1, 57]
45 y_min = []
46
47 for i in range(len(tr) - 1):
48     l = tr[i + 1] - tr[i]
49     y_min.append([l])
50 y_min = np.array(y_min)
51 print("List of Neon trough differences(V): ", y_min.
   flatten())
52
53
54 reg = LinearRegression().fit(n_min, y_min)
55 reg.score(n_min,y_min)
56 slope_tr = reg.coef_
57 int_tr = reg.intercept_
58 xfit = np.linspace(0, 4, 100)
59 yfit = reg.predict(xfit[:, np.newaxis])
60
61
62 g = plt.figure(2, figsize=(12,8))
63 plt.grid()
64 plt.title('Neon Franck-Hertz Analysis', fontsize=14)
65 plt.xlabel('Peak/Trough Number', fontsize=14)
66 plt.ylabel('Peak/Trough Position Difference (V)',
   fontsize=14)
67 plt.scatter(n_min,y_min, color = "black")
68 plt.plot(xfit, yfit, label = "Trough linear fit",
   color = "black")
69
70 n_max = np.array([1, 2, 3]).reshape((-1, 1))
71 pe = [13.8, 30.8, 48.7, 68.7]
72 y_max = []
73
74 for i in range(len(pe) - 1):
75     l = pe[i + 1] - pe[i]
76     y_max.append([l])
77 y_max = np.array(y_max)
78 print("List of Neon peak differences (V): ", y_max.
   flatten())
79
```

```
 80 reg = LinearRegression().fit(n_max, y_max)
 81 reg.score(n_max,y_max)
 82 slope_pk = reg.coef_
 83 int_pk = reg.intercept_
 84 xfit = np.linspace(0, 4, 100)
 85 yfit = reg.predict(xfit[:, np.newaxis])
 86 plt.scatter(n_max,y_max)
 87 plt.plot(xfit, yfit, label = "Peak linear fit")
 88 plt.legend()
 89 plt.show()
 90
 91 trough = np.poly1d(np.polyfit(n_min.flatten(), y_min
    .flatten(), 2))
 92 peak = np.poly1d(np.polyfit(n_max.flatten(), y_max.
    flatten(), 2))
 93
 94
 95 #Actual Analysis of FH
 96
 97
 98
 99 print('CONISISTENCY CHECK OF FIT COEFFICIENTS IN RSB
     METHOD')
100 print('Troughs analysis:')
101 a1 = 0.95
102 b1 = 14.15
103 c1 = 6
104 p1 = b1 - a1
105 m1 = 2 * a1
106 print('p = b-a: p = {:.2f}'.format(b1-a1))
107 print('m = 2a: m = {:.2f}'.format(2*a1))
108
109 print("model1:", model, "model2:", model2)
110
111 print('\nPeaks analysis:')
112 a2 = 0.75
113 b2 = 14.51
114 c2 = 1.4
115 p2 = b2 - a2
116 m2 = 2 * a2
117 print('p = b-a: p = {:.2f}' .format(b2-a2))
118 print('m = 2a: m = {:.2f}' .format(2*a2))
119
120 print('\n\nCALCULATION OF FIRST EXCITED STATE OF
```

```
120 NEON FROM DIFFERENT METHODS')
121
122 Ea_trad_pk = sum(y_max)/len(y_max)
123 Ea_trad_tr = sum(y_min)/len(y_min)
124
125 print('"Traditional" average peak spacing: ',
    Ea_trad_pk, ' V')
126 print('"Traditional" average trough spacing: ',
    Ea_trad_tr, ' V')
127
128 # From quadratic fit, see E_a as parameter b
129 # From RSB n=1/2 method, E_a = p + m/2
130
131 print('\nRSB method using line fit to peak/trough
    position differences')
132
133 E_a_pk = int_pk + slope_pk/2
134 E_a_tr = int_tr + slope_tr/2
135
136 print("\nTrough: ", E_a_tr, "\nPeak: ", E_a_pk )
137
138 print('\nRSB method using quadratic fit to peak/
    trough positions')
139
140 print("\nTrough: ", b1, "\nPeak: ", b2 )
141
142
143
144
145 #calculating teh 3p-3s energy spacing
146
147 small_pk = [39.2, 55.9, 72.7]
148 small_tr = [40, 56.6, 73.8]
149 list_3p3s = []
150
151 for i in range(len(small_pk)):
152     x = np.abs(small_tr[i] - small_pk[i])
153     list_3p3s.append([x])
154
155 print('Energy difference measurements (V)')
156 print(list_3p3s)
157
158 avg = sum(np.array(list_3p3s))/len(np.array(
    list_3p3s))
```

```
159
160 print('Measured average energy difference 3p-3s for
    neon: ',avg , 'eV')
161
```