

## Analyze CC from raw ABF Files

### User Input

```
In [2]: 1 dir_in = r'\Projects\All Olive\Coupling\ABF'
2 file_in = '2019_08_08_0019.abf'
3 dir_out = dir_in + '\\Analyzed'
4 ignore = list([13,117,65,111,9,8,87, 75, 71,101,121, 43, 147])
```

### Import Packages

```
In [3]: 1 import pyabf
2 import os
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 from IPython.display import display, HTML
```

### Import abf files

```
In [4]: 1 os.chdir(dir_in)
2 abf = pyabf.ABF(file_in)
```

### Declare variables

```
In [5]: 1 sweeps = abf.sweepList
2 included1 = list()
3 included2 = list()
4 count1 = list()
5 count2 = list()
6 sensitivity = 0.5
7
8 sf = abf.dataPointsPerMs # Scale factor for reading the raw data into the graph (in ms - must be integer)
9 sf_s = abf.dataPointsPerMs * 1000 # Scale factor for setting "cursors" in the graph (in seconds)
10 base_start1 = 0
11 base_end1 = 50 * sf
12 hyp_start1 = 172 * sf
13 hyp_end1 = 272 * sf
14 trace_end1 = 450 * sf
15
16 base_start2 = 967 * sf
17 base_end2 = 1067 * sf
18 hyp_start2 = 1120 * sf
19 hyp_end2 = 1220 * sf
20 trace_end2 = 1500 * sf
21
22 means1 = list()
23 means2 = list()
24 mean1 = np.nan
25 mean2 = np.nan
26
27 i = 0
28 for sweep in sweeps:
29     # mean of baselines for each channel
30     if sweep in ignore:
31         pass
32     else:
33         # Get X and Y data for this sweep on Ch1
34         abf.setSweep(sweepNumber = sweep, channel = 0)
35         means1.append(abf.sweepY [base_start1:base_end1])
36         #means1.append(abf.sweepY [(500 * sf):(1500 * sf)])
37
38         # Get X and Y data for this sweep on Ch2
39         abf.setSweep(sweepNumber = sweep, channel = 2)
40         means2.append(abf.sweepY [base_start2:base_end2])
41         #means2.append(abf.sweepY [(0 * sf):(1000 * sf)])
42
43     i = i + 1
44
45 mean1 = np.mean(means1)
46 mean2 = np.mean(means2)
47
48
49
50
```

### Filter Sweeps

In [6]:

```
1 i = 0
2
3
4 for sweep in sweeps:
5
6     # Is it in the ignore List?
7     if sweep in ignore:
8         pass
9
10    # Proceed to analysis
11    else:
12
13        # Get X and Y data for this sweep on Ch1
14        abf.setSweep (sweepNumber = sweep, channel = 0)
15
16        # Get values for region 1 cell 1 baseline region
17        x1 = abf.sweepX [base_start1:trace_end1]
18        y1 = abf.sweepY [base_start1:trace_end1]
19        x_base1 = abf.sweepX [base_start1:base_end1]
20        y_base1 = abf.sweepY [base_start1:base_end1]
21        y_base_mean1 = np.mean(y_base1)
22        y_base_max1 = max(y_base1)
23        y_base_min1 = min(y_base1)
24
25        # Get values for Region 2 cell 1 baseline region
26        x3 = abf.sweepX [base_start2:trace_end2]
27        y3 = abf.sweepY [base_start2:trace_end2]
28        x_base3 = abf.sweepX [base_start2:base_end2]
29        y_base3 = abf.sweepY [base_start2:base_end2]
30        y_base_mean3 = np.mean(y_base3)
31        y_base_max3 = max(y_base3)
32        y_base_min3 = min(y_base3)
33
34        # Get values for Region 1 cell 1 hyperpolarizing region
35        x_hyp1 = abf.sweepX [hyp_start1:hyp_end1]
36        y_hyp1 = abf.sweepY [hyp_start1:hyp_end1]
37        y_hyp_mean1 = np.mean(y_hyp1)
38        y_hyp_max1 = max(y_hyp1)
39        y_hyp_min1 = min(y_hyp1)
40
41        # Get values for Region 1 cell 2 hyperpolarizing region
42        x_hyp3 = abf.sweepX [hyp_start2:hyp_end2]
43        y_hyp3 = abf.sweepY [hyp_start2:hyp_end2]
44        y_hyp_mean3 = np.mean(y_hyp3)
45        y_hyp_max3 = max(y_hyp3)
46        y_hyp_min3 = min(y_hyp3)
47
48        # Get X and Y data for this sweep on Ch1
49        abf.setSweep (sweepNumber = sweep, channel = 2)
50
51        # Get values for Region 1 cell 2 baseline region
52        x2 = abf.sweepX [base_start1:trace_end1]
53        y2 = abf.sweepY [base_start1:trace_end1]
54        x_base2 = abf.sweepX [base_start1:base_end1]
55        y_base2 = abf.sweepY [base_start1:base_end1]
56        y_base_mean2 = np.mean(y_base2)
57        y_base_max2 = max(y_base2)
58        y_base_min2 = min(y_base2)
59
60        # Get values for Region 2 cell 2 baseline region
61        x4 = abf.sweepX [base_start2:trace_end2]
62        y4 = abf.sweepY [base_start2:trace_end2]
63        x_base4 = abf.sweepX [base_start2:base_end2]
64        y_base4 = abf.sweepY [base_start2:base_end2]
65        y_base_mean4 = np.mean(y_base4)
66        y_base_max4 = max(y_base4)
67        y_base_min4 = min(y_base4)
68
69        # Get values for cell 2 hyperpolarizing region
70        x_hyp2 = abf.sweepX [hyp_start1:hyp_end1]
71        y_hyp2 = abf.sweepY [hyp_start1:hyp_end1]
72        y_hyp_mean2 = np.mean(y_hyp2)
73        y_hyp_max2 = max(y_hyp2)
74        y_hyp_min2 = min(y_hyp2)
75
76        # Get values for cell 2 hyperpolarizing region
77        x_hyp4 = abf.sweepX [hyp_start2:hyp_end2]
78        y_hyp4 = abf.sweepY [hyp_start2:hyp_end2]
79        y_hyp_mean4 = np.mean(y_hyp4)
80        y_hyp_max4 = max(y_hyp4)
81        y_hyp_min4 = min(y_hyp4)
82
83
84        # Is cell 1 healthy?
85        if y_base_mean1 > -30:
86            ignore.append (sweep)
87            print ('Sweep #' + str(sweep) + ' removed: Vm exceeded - 30 mV on Cell 1(' + str(base_mean1) + ')')
88
89        # Is cell 2 healthy?
90        elif y_base_mean2 > -30:
91            ignore.append (sweep)
92            print ('Sweep #' + str(sweep) + ' removed: Vm exceeded - 30 mV on Cell 2')
93
94        # Filter 1 --> 2 connections
95        if sweep % 2 == 0 and sweep not in ignore:
96
97            # Is there a spike in the baseline of Region 1 cell 1?
98            if y_base_min1 < mean1 + (mean1 * sensitivity) or y_base_max1 > mean1 - (mean1 * sensitivity) :
99                ignore.append(sweep)
100                display(HTML('Sweep #' + str(sweep) + ' removed: Vm changed more than 25% on Region 1 Cell 1'))
101
102            # Is there a spike in the baseline of Region 1 cell 2
103            elif y_base_min2 < mean2 + (mean2 * sensitivity) or y_base_max2 > mean2 - (mean2 * sensitivity) :
104                ignore.append(sweep)
105                display(HTML('Sweep #' + str(sweep) + ' removed: Vm changed more than 25% on Region 1 Cell 2'))
106
107            # Is there a spike in the hyperpolarizing of Region 1 cell 2
108            elif y_hyp_min2 < y_base_mean2 + (y_base_mean2 * (sensitivity)) or y_hyp_max2 > y_base_mean2 - (y_base_mean2 * (sensitivity)) :
109                ignore.append(sweep)
110                display(HTML('Sweep #' + str(sweep) + ' removed: Spike in hyperpolarization in Region 1 Cell 2'))
111
112        else:
113            included1.append (sweep)
114            count1.append(i)
```

```
115
116 # Filter 2 -> 1 connection
117 elif sweep % 2 != 0 and sweep not in ignore:
118     #included2.append(sweep)
119     # Is there a spike in the baseline of Region 2 cell 1?
120     if y_base_min3 < mean1 + (mean1 * sensitivity) or y_base_max3 > mean1 - (mean1 * sensitivity) :
121         ignore.append(sweep)
122         display(HTML('Sweep #' + str(sweep) + ' removed: Vm changed more than 25% on Region 2 Cell 1') )
123
124     # Is there a spike in the baseline of Region 2 cell 2
125     #elif y_base_min4 < mean2 + (mean2 * sensitivity) or y_base_max4 > mean2 - (mean2 * sensitivity) :
126         #ignore.append(sweep)
127         #display(HTML('Sweep #' + str(sweep) + ' removed: Vm changed more than 25% on Region 2 Cell 2') )
128         # print(y_base_min4, mean2 + (mean2 * sensitivity), y_base_max4, mean2 - (mean2 * sensitivity))
129
130     # Is there a spike in the hyperpolarizing of Region 2 cell 1
131     elif y_hyp_min3 < mean1 + (mean1 * (sensitivity)) or y_hyp_max3 > mean1 - (mean1 * (sensitivity)) :
132         ignore.append(sweep)
133         display(HTML('Sweep #' + str(sweep) + ' removed: Spike in hyperpolarization in Region 2 Cell 1') )
134
135     else:
136         included2.append(sweep)
137         count2.append(i)
138     i = i + 1
139
140 df_filter = pd.DataFrame({'Filter': ['# Analyzed'],
141                          'Odd Sweep Numbers': [len(included1)],
142                          'Even Sweep Numbers': [len(included2)]})
143
144 display(HTML(df_filter.to_html()))
```

Sweep #59 removed: Vm changed more than 25% on Region 2 Cell 1

Sweep #92 removed: Vm changed more than 25% on Region 1 Cell 1

Sweep #95 removed: Vm changed more than 25% on Region 2 Cell 1

Sweep #102 removed: Vm changed more than 25% on Region 1 Cell 1

Sweep #104 removed: Vm changed more than 25% on Region 1 Cell 1

Sweep #105 removed: Vm changed more than 25% on Region 2 Cell 1

Sweep #108 removed: Spike in hyperpolarization in Region 1 Cell 2

Sweep #116 removed: Vm changed more than 25% on Region 1 Cell 1

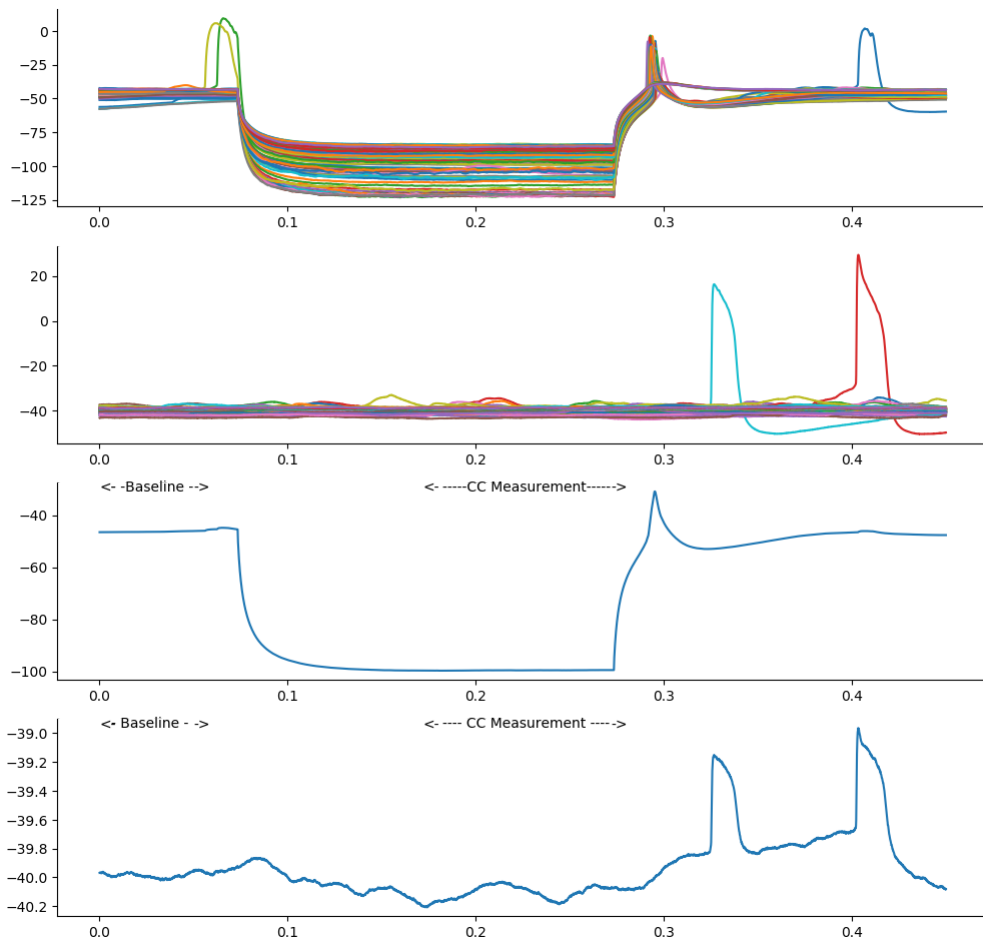
Sweep #158 removed: Vm changed more than 25% on Region 1 Cell 2

Filter	Odd Sweep Numbers	Even Sweep Numbers
0 # Analyzed	85	74

Connections 1--> 2

In [16]:

```
1 df1x = pd.DataFrame()
2 df1y = pd.DataFrame()
3 df2x = pd.DataFrame()
4 df2y = pd.DataFrame()
5 dfcmdx = pd.DataFrame()
6 dfcmdy = pd.DataFrame()
7 df_out = pd.DataFrame()
8 # print (included1)
9
10 file_out = file_in.replace('.abf','') + '_1-2_.csv'
11
12 i = 0
13 fig, ax = plt.subplots(4,1, figsize=(12, 12), dpi = 100)
14 for sweep in sweeps:
15     if sweep in included1:
16
17         abf.setSweep (sweepNumber = sweep, channel = 0)
18         x1 = abf.sweepX [base_start1:trace_end1]
19         y1 = abf.sweepY [base_start1:trace_end1]
20         df1x[sweep] = x1
21         df1y[sweep] = y1
22         ax[0].plot (x1, y1, linestyle = 'solid')
23
24         abf.setSweep (sweepNumber = sweep, channel = 1)
25         cmd_x1 = abf.sweepX [base_start1:trace_end1]
26         cmd_y1 = abf.sweepY [base_start1:trace_end1]
27         dfcmdx[sweep] = cmd_x1
28         dfcmdy[sweep] = cmd_y1
29
30         abf.setSweep (sweepNumber = sweep, channel = 2)
31         x2 = abf.sweepX [base_start1:trace_end1]
32         y2 = abf.sweepY [base_start1:trace_end1]
33         df2x[sweep] = x2
34         df2y[sweep] = y2
35         ax[1].plot (x2, y2, linestyle = 'solid')
36
37         i = i + 1
38
39 df_out['x_stim'] = df1x.mean(axis=1)
40 df_out['y_stim'] = df1y.mean(axis=1)
41 df_out['x_rec'] = df2x.mean(axis=1)
42 df_out['y_rec'] = df2y.mean(axis=1)
43 df_out['x_cmd'] = dfcmdx.mean(axis=1)
44 df_out['y_cmd'] = dfcmdy.mean(axis=1)
45
46 df_out['vm_stim'] = df_out.loc[base_start1:base_end1]['y_stim'].mean()
47 df_out['vm_rec'] = df_out.loc[base_start1:base_end1]['y_rec'].mean()
48 df_out['vrec'] = df_out.loc[hyp_start1:hyp_end1]['y_rec'].mean() - df_out['vm_rec']
49 df_out['vstim'] = df_out.loc[hyp_start1:hyp_end1]['y_stim'].mean() - df_out['vm_stim']
50 df_out['istim'] = df_out.loc[hyp_start1:hyp_end1]['y_cmd'].mean() - df_out.loc[base_start1:base_end1]['y_cmd'].mean()
51 df_out['p_cc'] = df_out['vrec'] / df_out['vstim'] * 100
52
53 ax[0].spines['right'].set_visible(False)
54 ax[0].spines['top'].set_visible(False)
55 ax[1].spines['right'].set_visible(False)
56 ax[1].spines['top'].set_visible(False)
57
58 ax[2].plot (df_out['x_stim'],df_out['y_stim'], linestyle = 'solid')
59 ax[2].annotate('<->', xy=((base_start1/sf_s), max(df_out['y_stim'])), color = 'black')
60 ax[2].annotate('<->', xy=((base_end1/sf_s), max(df_out['y_stim'])), color = 'black')
61 ax[2].annotate('<- Baseline ->', xy=((base_end1/sf_s - base_start1/sf)/8, max(df_out['y_stim'])), color = 'black')
62 ax[2].annotate('<->', xy=((hyp_start1/sf_s), max(df_out['y_stim'])), color = 'black')
63 ax[2].annotate('<->', xy=((hyp_end1/sf_s), max(df_out['y_stim'])), color = 'black')
64 ax[2].annotate('<-----CC Measurement----->', xy=((hyp_start1/sf_s + (hyp_end1/sf_s - hyp_start1/sf_s)/10), max(df_out['y_stim'])), color = 'black')
65 ax[2].spines['right'].set_visible(False)
66 ax[2].spines['top'].set_visible(False)
67
68 ax[3].plot (df_out['x_rec'],df_out['y_rec'], linestyle = 'solid')
69 ax[3].annotate('<->', xy=((base_start1/sf_s), max(df_out['y_rec'])), color = 'black')
70 ax[3].annotate('<->', xy=((base_end1/sf_s), max(df_out['y_rec'])), color = 'black')
71 ax[3].annotate('<- Baseline ->', xy=((base_end1/sf_s - base_start1/sf)/8, max(df_out['y_rec'])), color = 'black')
72 ax[3].annotate('<->', xy=((hyp_start1/sf_s), max(df_out['y_rec'])), color = 'black')
73 ax[3].annotate('<->', xy=((hyp_end1/sf_s), max(df_out['y_rec'])), color = 'black')
74 ax[3].annotate('<-----CC Measurement----->', xy=((hyp_start1/sf_s + (hyp_end1/sf_s - hyp_start1/sf_s)/10), max(df_out['y_rec'])), color = 'black')
75 ax[3].spines['right'].set_visible(False)
76 ax[3].spines['top'].set_visible(False)
77
78 try:
79     os.stat(dir_out)
80 except:
81     os.makedirs(dir_out)
82
83 # Write data to file
84 os.chdir(dir_out)
85 df_out.to_csv(file_out, index = False)
```



```
In [25]: 1 file_out = file_in.replace('.abf','') + '_1-2_summary.csv'
2
3 df_summary = pd.DataFrame(df_out.iloc[0][6:12])
4 df_summary = df_summary.reset_index()
5 df_summary.columns = ['measurement', 'mean']
6 df_summary = df_summary.transpose()
7 df_summary.columns = df_summary.iloc[0][:]
8 df_summary = df_summary[df_summary.index != 'measurement']
9
10
11 display(HTML(df_summary.to_html()))
12
13 try:
14     os.stat(dir_out)
15 except:
16     os.makedirs(dir_out)
17
18 # Write data to file
19 os.chdir(dir_out)
20 df_out.to_csv(file_out, index = False)
21
```

measurement	vm_stim	vm_rec	vrec	vstim	istim	p_cc
mean	-46.4109	-39.9894	-0.119301	-53.1536	-201.608	0.224446

**Connections 2 --> 1**

```

In [9]: 1 df1x = pd.DataFrame()
2 df1y = pd.DataFrame()
3 df2x = pd.DataFrame()
4 df2y = pd.DataFrame()
5 df_out2 = pd.DataFrame()
6
7
8 file_out = file_in.replace('.abf','') + '_2-1_.csv'
9
10 i = 0
11 fig, ax = plt.subplots(4,1, figsize=(12, 12), dpi = 100)
12 for sweep in sweeps:
13     if sweep in included2:
14
15         abf.setSweep (sweepNumber = sweep, channel = 0)
16         x1 = abf.sweepX [base_start2:trace_end2]
17         y1 = abf.sweepY [base_start2:trace_end2]
18         df1x[sweep] = x1
19         df1y[sweep] = y1
20
21         ax[0].plot (x1, y1, linestyle = 'solid')
22         abf.setSweep (sweepNumber = sweep, channel = 2)
23         x2 = abf.sweepX [base_start2:trace_end2]
24         y2 = abf.sweepY [base_start2:trace_end2]
25         df2x[sweep] = x2
26         df2y[sweep] = y2
27         ax[1].plot (x2, y2, linestyle = 'solid')
28
29         i = i + 1
30
31 df_out2['x_rec'] = df1x.mean(axis=1)
32 df_out2['y_rec'] = df1y.mean(axis=1)
33 df_out2['x_stim'] = df2x.mean(axis=1)
34 df_out2['y_stim'] = df2y.mean(axis=1)
35
36 df_out2['vm_stim'] = df_out2.loc[base_start1:base_end1]['y_stim'].mean()
37 df_out2['vm_rec'] = df_out2.loc[base_start1:base_end1]['y_rec'].mean()
38 df_out2['vrec'] = df_out2.loc[hyp_start1:hyp_end1]['y_rec'].mean() - df_out2['vm_rec']
39 df_out2['vstim'] = df_out2.loc[hyp_start1:hyp_end1]['y_stim'].mean() - df_out2['vm_stim']
40 df_out2['istim'] = df_out2['istim']
41 df_out2['p_cc'] = df_out2['vrec'] / df_out2['vstim'] * 100
42
43 ax[0].spines['right'].set_visible(False)
44 ax[0].spines['top'].set_visible(False)
45 ax[1].spines['right'].set_visible(False)
46 ax[1].spines['top'].set_visible(False)
47
48 ax[2].plot (df_out2['x_stim'],df_out2['y_rec'], linestyle = 'solid')
49 ax[2].annotate('<->', xy=((base_start2/sf_s), max(df_out2['y_rec']))), color = 'black')
50 ax[2].annotate('<->', xy=((base_end2/sf_s), max(df_out2['y_rec']))), color = 'black')
51 ax[2].annotate('<->', xy=((hyp_start2/sf_s), max(df_out2['y_rec']))), color = 'black')
52 ax[2].annotate('<->', xy=((hyp_end2/sf_s), max(df_out2['y_rec']))), color = 'black')
53 ax[2].spines['right'].set_visible(False)
54 ax[2].spines['top'].set_visible(False)
55
56 ax[3].plot (df_out2['x_rec'],df_out2['y_stim'], linestyle = 'solid')
57 ax[3].annotate('<->', xy=((base_start2/sf_s), max(df_out2['y_stim']))), color = 'black')
58 ax[3].annotate('<->', xy=((base_end2/sf_s), max(df_out2['y_stim']))), color = 'black')
59 ax[3].annotate('<->', xy=((hyp_start2/sf_s), max(df_out2['y_stim']))), color = 'black')
60 ax[3].annotate('<->', xy=((hyp_end2/sf_s), max(df_out2['y_stim']))), color = 'black')
61 ax[3].spines['right'].set_visible(False)
62 ax[3].spines['top'].set_visible(False)
63
64
65 try:
66     os.stat(dir_out)
67 except:
68     os.makedirs(dir_out)
69
70 # Write data to file
71 os.chdir(dir_out)
72 df_out2.to_csv(file_out, index = False)

```

```
In [26]: 1 file_out = file_in.replace('.abf','') + '_2-1_summary.csv'
2
3 df_summary = pd.DataFrame(df_out2.iloc[0][4:12])
4 df_summary = df_summary.reset_index()
5 df_summary.columns = ['measurement', 'mean']
6 df_summary = df_summary.transpose()
7 df_summary.columns = df_summary.iloc[0][:]
8 df_summary = df_summary[df_summary.index != 'measurement']
9
10
11
12 display(HTML(df_summary.to_html()))
13
14 try:
15     os.stat(dir_out)
16 except:
17     os.makedirs(dir_out)
18
19 # Write data to file
20 os.chdir(dir_out)
21 df_out2.to_csv(file_out, index = False)
```

measurement	vm_stim	vm_rec	vrec	vstim	istim	p_cc
mean	-39.8397	-46.2078	0.45512	-26.2637	-201.608	-1.73289

```
In [ ]: 1
```