

```
In [1]: 1 import warnings
2 import numpy as np
3 import pandas as pd
4 from pyIClab import (
5     IonExchanger, Column, Eluent, SwitchingValve, PEEKTubing,
6     SampleLoop, DSM_SimpleEquilibriums, ContaminatedPhreeqcSuppressorBeta,
7     Detector, IonChromatograph, DSM_SEConstrutor
8 )
9 import seaborn as sns
10 import matplotlib.pyplot as plt
11 from pyIClab.beadedbag import mpl_custom_rcconfig
12 from IPython.display import clear_output
13 from scipy.integrate import quad
14 from scipy.interpolate import interp1d
```

```
In [2]: 1 test_params = dict(
2     test_name = 'Gradient I',
3     fname = 'homemade_EG_20240801.dat',
4     directory = 'db',
5     profile = {
6         'OH-':(
7             (0, 18),
8             (0.01, 18),
9             (23, 29),
10            (23, 18),
11            (25, 18),
12            ),
13     },
14     length = '15 cm',
15     inner_diameter = '4.6 mm',
16     CO2_level = .01,
17     sample = {
18         'F-': '0.1 mM',
19         'Cl-': '0.1 mM',
20         'NO2-': '0.2 mM',
21         'Br-': '0.2 mM',
22         'NO3-': '0.2 mM',
23         'SO4-2': '0.1 mM',
24         'PO4-3': '0.2 mM',
25     },
26 )
27
```

```
In [3]: 1 from pyIClab import DSM_CEConstrutor
2 from pyIClab.engines.equilibriums import find_x_LSSM
3
4 class LocalConstructor(DSM_CEConstrutor):
5
6     def set_x(self):
7
8         kmap = self.set_kmap()
9
10        return find_x_LSSM(kmap, -1)
11
```

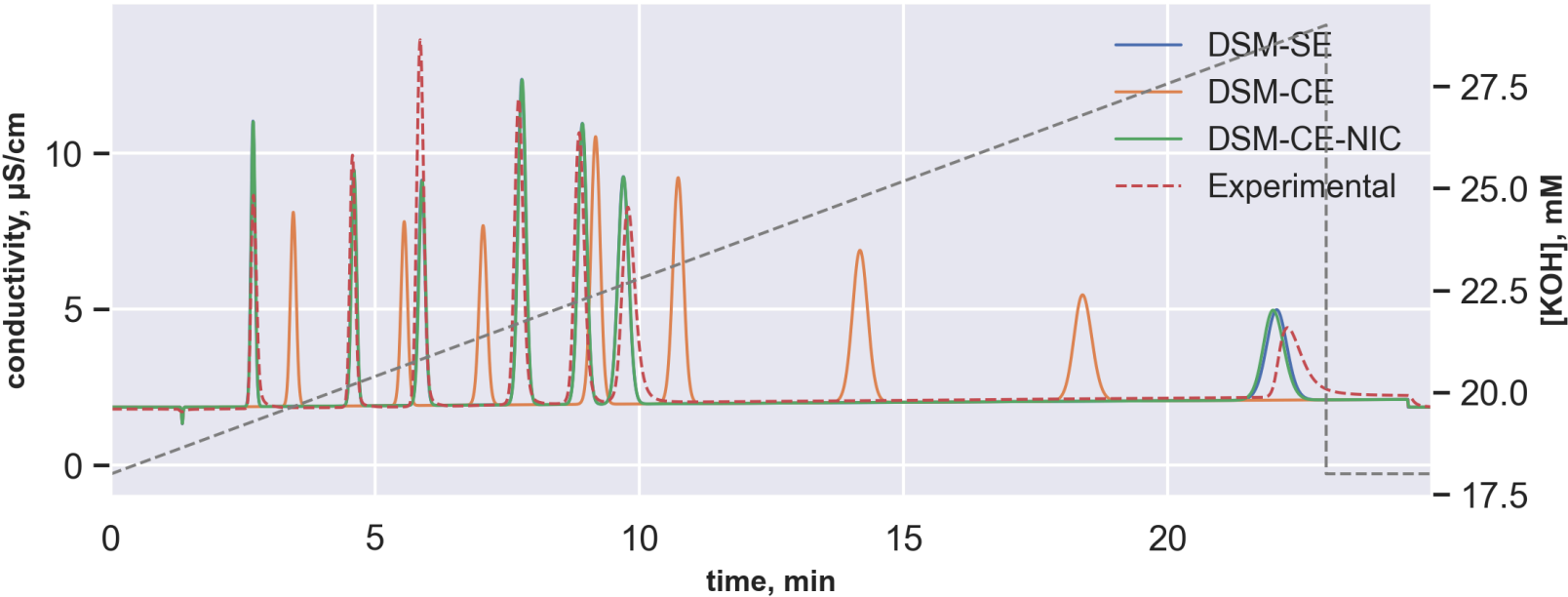
```
In [4]: 1 def unit_test(
2     *,
3     test_name,
4     fname,
5     directory,
6     length,
7     inner_diameter,
8     profile,
9     CO2_level,
10    sample,
11    model_constructor_prompt,
12    ):
13
14    # -----
15    sp = IonExchanger.load(fname, directory=directory)
16    column = Column('Column', length=length, ID=inner_diameter)
17    column.pack(sp)
18
19    # -----
20    eluent = Eluent(name='EG', profile=profile)
21    tb = SampleLoop('PEEK', V='0.21 mL') # hold-up volume
22    sixport = SwitchingValve.SixPort()
23    loop = SampleLoop('Loop', '25 uL')
24    suppressor = ContaminatedPhreeqcSuppressorBeta('Suppressor', 'anion', _CO2_level=CO2_level)
25    detector = Detector('Detector')
26
27    # -----
28    eluent.assemble(tb)
29    sixport.assemble(0, tb)
30    sixport.assemble(1, column)
31    sixport.assemble([2, 5], loop)
32    column.assemble(suppressor)
33    suppressor.assemble(detector)
34    ic = IonChromatograph('Gradient-Test', ('OH-',), lockon=sixport)
35
36    commands = '''
37    0.0 min, sixport, inject
38    0.5 min, sixport, load
39    '''
40    ic.reset_commands(commands)
41    prompt = model_constructor_prompt if model_constructor_prompt != 'DSM_CE_NIC' else LocalConstructo
42    ic.set_ModelConstructor(prompt , column)
43
44    # -----
45    water = {'Cl-': '1e-9 mM'}
46    ic.inject(water, loop)
47    ic.go(tmax=eluent._tmax)
48    df1 = detector.get_signals(signal_type='conductivity')
49    df1.to_csv(
50        f'''{test_name}-{str(model_constructor_prompt).replace('_', '-')}.txt''',
51        '-Background.txt',
52        index=False,
53    )
54
55    # -----
56    ic.inject(sample, loop)
57    ic.go(tmax=eluent._tmax)
58    df2 = detector.get_signals(signal_type='conductivity')
59    df2.to_csv(
60        f'''{test_name}-{str(model_constructor_prompt).replace('_', '-')}.txt''',
61        '-Total.txt',
62        index=False,
63    )
64
65    return df1, df2
```

```
In [5]: 1 for model_constructor_prompt in ['DSM_SE', 'DSM_CE', 'DSM_CE_NIC']:
2     with warnings.catch_warnings(action='ignore'):
3         df1, df2 = unit_test(
4             model_constructor_prompt=model_constructor_prompt,
5             **test_params,
6         )
7         clear_output()
```

```
In [6]: 1 backgrounds = {}
2 chroms = {}
3 for model in ['DSM-SE', 'DSM-CE', 'DSM-CE-NIC']:
4     test_name = test_params.get('test_name')
5     fname_bg = f'{test_name}-{model}-Background.txt'
6     backgrounds[model] = pd.read_csv(fname_bg)
7     fname_tt = f'{test_name}-{model}-Total.txt'
8     chroms[model] = pd.read_csv(fname_tt)
9
10 df_exp = pd.read_csv(f'{test_name}-exp-Total.txt',
11     sep='\s+',
12     skiprows=43,
13     names=['time', 'step', 'signal'],
14     )[['time', 'signal']]
15
```

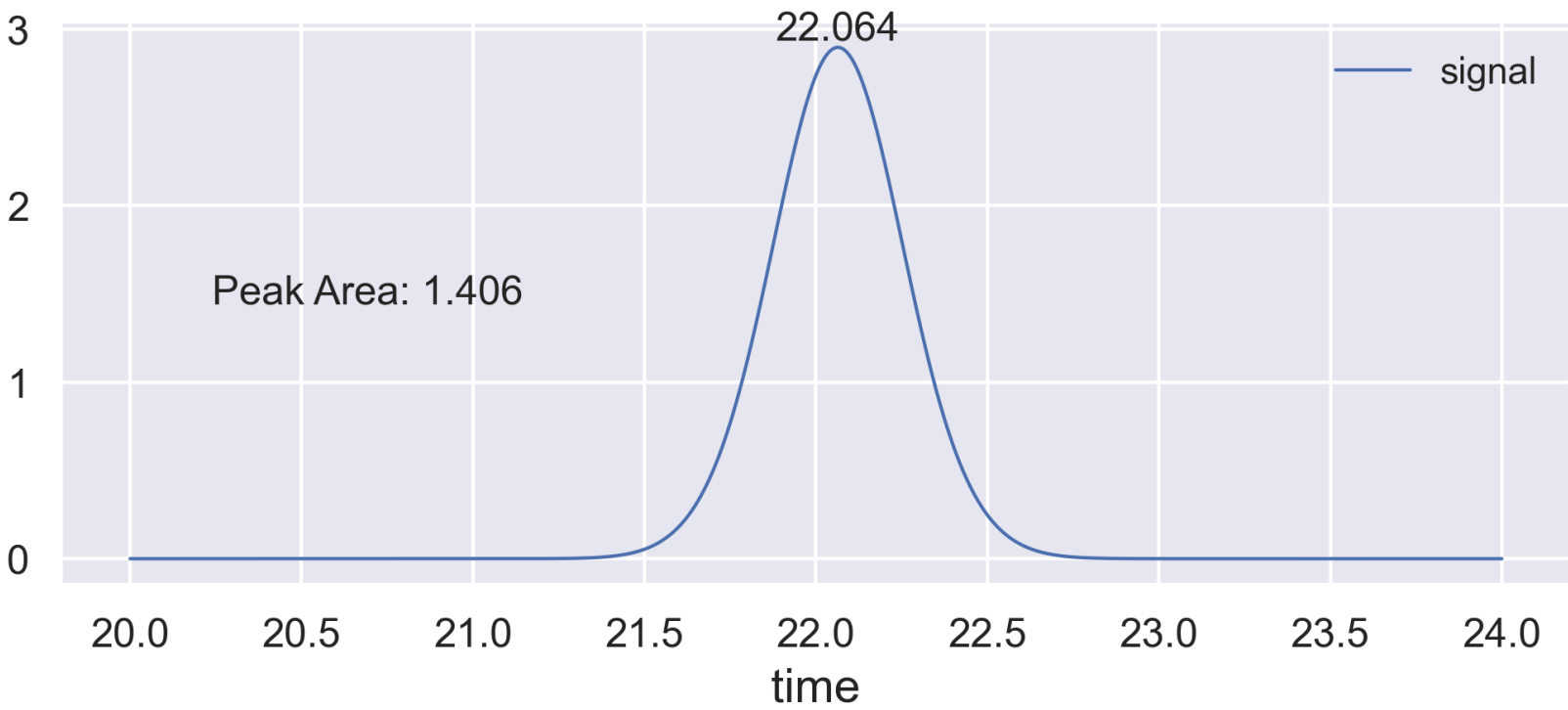
```
In [7]: 1 sns.set()
2 plt.rcParams.update(mpl_custom_rcconfig)
3 fig, ax = plt.subplots()
4
5 for i, model in enumerate(['DSM-SE', 'DSM-CE', 'DSM-CE-NIC']):
6
7     df = chroms[model]
8     x, y = df['time'], df['signal']
9     ax.plot(x, y, label=model)
10
11 ax.plot('time', 'signal', data=df_exp, label='Experimental', linestyle='--')
12 ax.set_xlabel('time, min', fontsize=10, fontweight='bold')
13 ax.set_ylabel('conductivity, μS/cm', fontsize=10, fontweight='bold')
14 ax.set(xlim=(0, max(x)), ylim=(-1, max(y)*1.2))
15 ax.legend()
16
17 # for i in peaks:
18 #     tR = x[i]
19 #     signal = y[i]
20 #     ax.text(tR, signal + .1, f'{tR:.2f}', ha='center', zorder=2)
21
22 profile = test_params.get('profile')
23 ax2 = ax.twinx()
24 ax2.plot(*zip(*profile['OH-']), color='grey', linestyle='--', zorder=1)
25 ax2.grid(visible=False)
26 ax2.set_ylabel('[KOH], mM', fontsize=10, fontweight='bold')
```

Out[7]: Text(0, 0.5, '[KOH], mM')



In [18]:

```
1 df0 = backgrounds['DSM-SE']
2 df1 = chroms['DSM-SE']
3 f0 = interp1d(df0['time'], df0['signal'])
4 f1 = interp1d(df1['time'], df1['signal'])
5 f = lambda t: f1(t) - f0(t)
6
7 window = (20, 24)
8 t = np.linspace(*window, 10000, endpoint=True)
9 df = pd.DataFrame(data=dict(time=t, signal=f(t)))
10 ax = df.plot(x='time', y='signal')
11 ax.text(df['time'][df['signal'].argmax()], df['signal'].max(),
12         s=round(df['time'][df['signal'].argmax()], 3),
13         ha='center',
14         va='bottom',)
15 with warnings.catch_warnings(action='ignore'):
16     ax.text(0.1, .5, f'Peak Area: {round(quad(f, *window)[0], 4)}',
17            transform=ax.transAxes,)
```



In []:

1