In [16]:

```
import warnings
import numpy as np
import pandas as pd
from pyIClab import (
    IonExchanger, Column, Eluent, SwitchingValve, PEEKTubing,
    SampleLoop, DSM_SimpleEquilibriums, ContaminatedPhreeqcSuppressorBeta,
    Detector, IonChromatograph, DSM_SEConstrutor
    )
import seaborn as sns
import matplotlib.pyplot as plt
from pyIClab.beadedbag import mpl_custom_rcconfig
from IPython.display import clear_output
from scipy.integrate import quad
from scipy.interpolate import interp1d
```

In [2]:

```
test_params = dict(
    test_name = 'Gradient A',
    fname = 'as18-20240731.dat',
    directory = 'db',
    profile = {
        'OH-':(
            (5, 10),
            (13, 25),
            (15, 25),
            (15, 10),
            (20, 10),
            ),
        },
    length = '25 cm',
    inner_diameter = '4 mm',
    CO2_level = .0101,
    sample = {
        'F-': '0.1 mM',
        'Cl-': '0.1 mM',
        'NO2-': '0.2 mM',
        'Br-': '0.2 mM',
        'NO3-': '0.2 mM',
        'SO4-2': '0.1 mM',
        'PO4-3': '0.2 mM',
        },
    )

```

In [3]:

```
from pyIClab import DSM_CEConstrutor
from pyIClab.engines.equilibriums import find_x_LSSM

class LocalConstructor(DSM_CEConstrutor):

    def set_x(self):

        kmap = self.set_kmap()

        return find_x_LSSM(kmap, -1)

```

In [4]:

```
def unit_test(
    *,
    test_name,
    fname,
    directory,
    length,
    inner_diameter,
    profile,
    CO2_level,
    sample,
    model_constructor_prompt,
    ):

    # -----------
    sp = IonExchanger.load(fname, directory=directory)
    column = Column('Column', length=length, ID=inner_diameter)
    column.pack(sp)

    # -----------
    eluent = Eluent(name='EG', profile=profile)
    tb = SampleLoop('PEEK', V='0.21 mL') # hold-up volume
    sixport = SwitchingValve.SixPort()
    loop = SampleLoop('Loop', '25 uL')
    suppressor = ContaminatedPhreeqcSuppressorBeta('Suppressor', 'anion', _CO2_level=CO2_level)
    detector = Detector('Detector')

    # -----------
    eluent.assemble(tb)
    sixport.assemble(0, tb)
    sixport.assemble(1, column)
    sixport.assemble([2, 5], loop)
    column.assemble(suppressor)
    suppressor.assemble(detector)
    ic = IonChromatograph('Gradient-Test', ('OH-',), lockon=sixport)

    commands = '''
    0.0 min, sixport, inject
    0.5 min, sixport, load
    '''
    ic.reset_commands(commands)
    prompt = model_constructor_prompt if model_constructor_prompt != 'DSM_CE_NIC' else LocalConstructo
    ic.set_ModelConstructor(prompt , column)

    # -----------
    water = {'Cl-': '1e-9 mM'}
    ic.inject(water, loop)
    ic.go(tmax=eluent._tmax)
    df1 = detector.get_signals(signal_type='conductivity')
    df1.to_csv(
        f'''{test_name}-{str(model_constructor_prompt).replace('_', '-')}'''
        '''-Background.txt''',
        index=False,
        )

    # -----------
    ic.inject(sample, loop)
    ic.go(tmax=eluent._tmax)
    df2 = detector.get_signals(signal_type='conductivity')
    df2.to_csv(
        f'''{test_name}-{str(model_constructor_prompt).replace('_', '-')}'''
        '''-Total.txt''',
        index=False,
        )

    return df1, df2
```

In [5]:

```
for model_constructor_prompt in ['DSM_SE', 'DSM_CE', 'DSM_CE_NIC']:
    with warnings.catch_warnings(action='ignore'):
        df1, df2 = unit_test(
            model_constructor_prompt=model_constructor_prompt,
            **test_params,
            )
        clear_output()
```

In [13]:

```
backgrounds = {}
chroms = {}
for model in ['DSM-SE', 'DSM-CE', 'DSM-CE-NIC']:
    test_name = test_params.get('test_name')
    fname_bg = f'{test_name}-{model}-Background.txt'
    backgrounds[model] = pd.read_csv(fname_bg)
    fname_tt = f'{test_name}-{model}-Total.txt'
    chroms[model] = pd.read_csv(fname_tt)

df_exp = pd.read_csv(f'{test_name}-exp-Total.txt',
    sep='\s+',
    skiprows=43,
    names=['time', 'step', 'signal'],
    )[['time', 'signal']]

```

Out[13]:

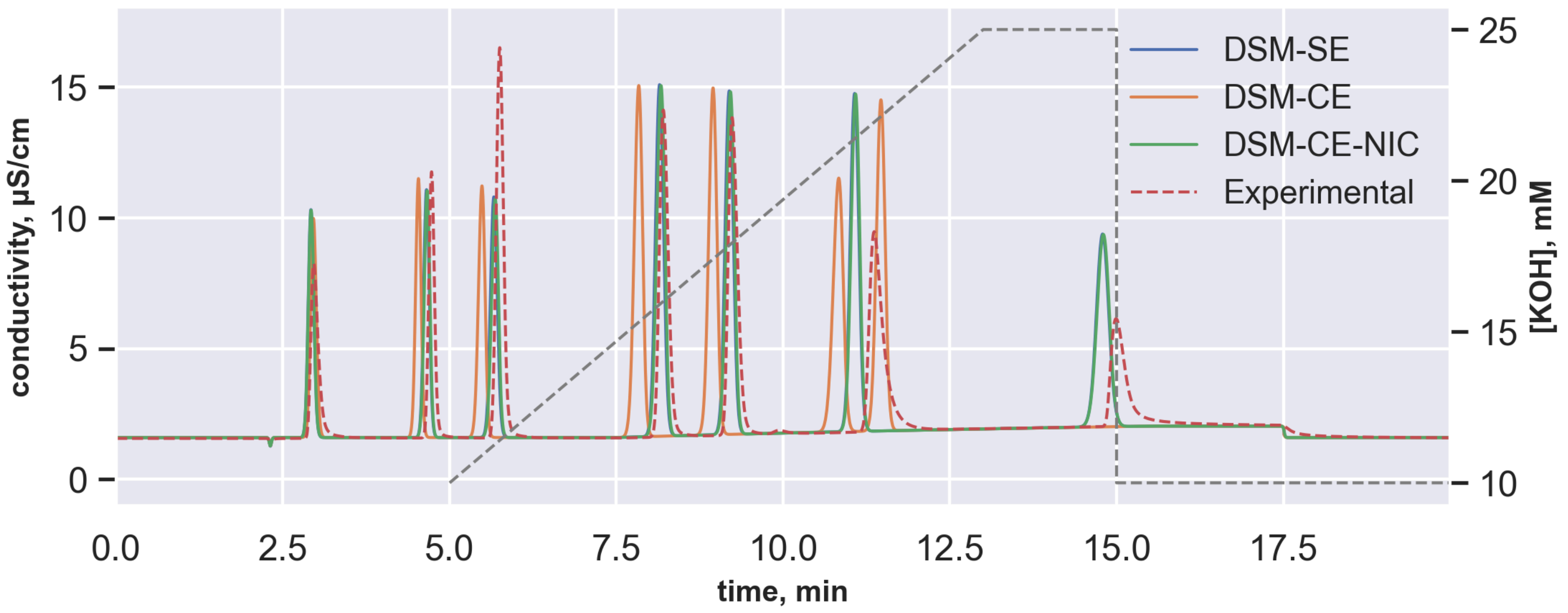
	time	signal
0	0.000000	1.56545
1	0.001667	1.56546
2	0.003333	1.56546
3	0.005000	1.56546
4	0.006667	1.56546
...	...	...
11996	19.993333	1.58953
11997	19.995000	1.58951
11998	19.996667	1.58949
11999	19.998333	1.58947
12000	20.000000	1.58945

12001 rows × 2 columns

In [25]:

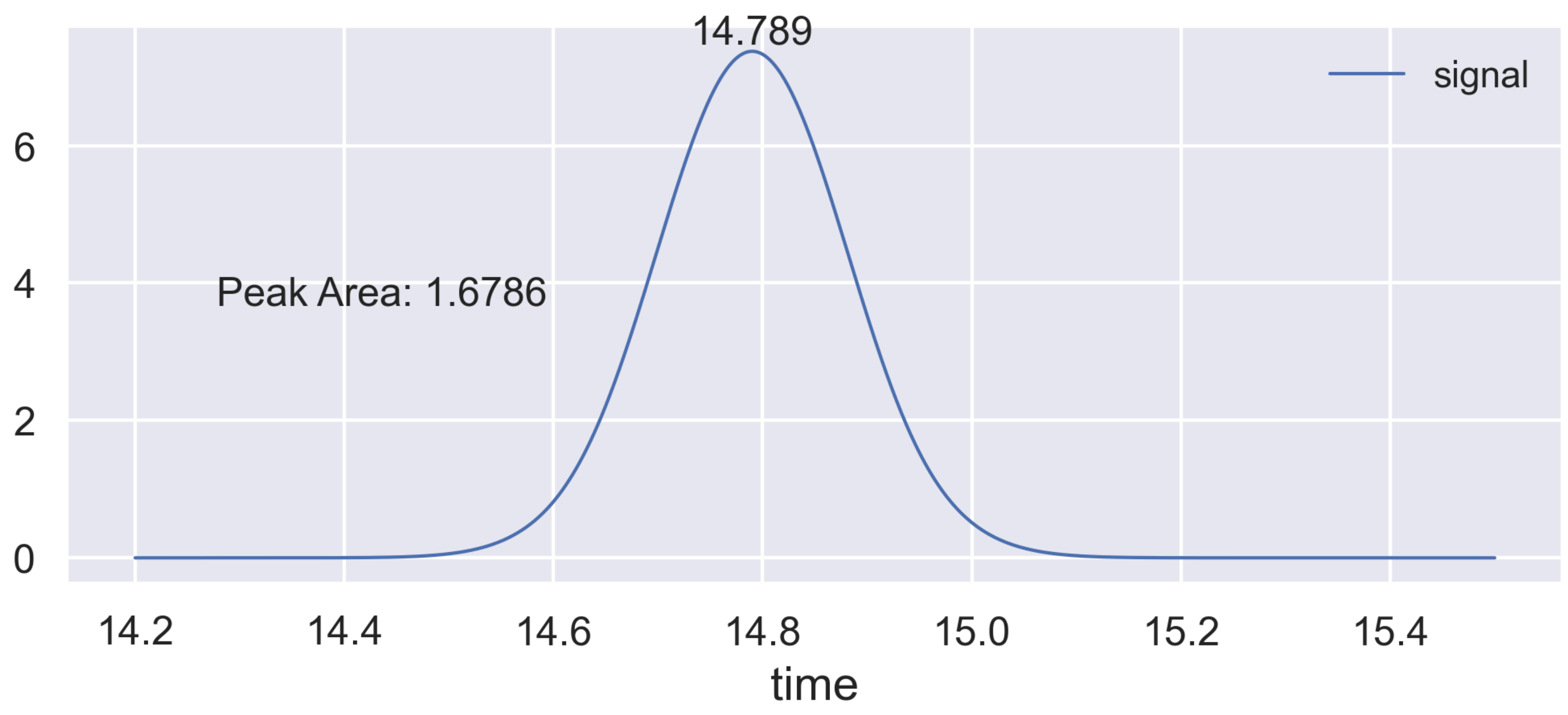
```
sns.set()
plt.rcParams.update(mpl_custom_rcconfig)
fig, ax = plt.subplots()

for i, model in enumerate(['DSM-SE', 'DSM-CE', 'DSM-CE-NIC']):

    df = chroms[model]
    x, y = df['time'], df['signal']
    ax.plot(x, y, label=model)

ax.plot('time', 'signal', data=df_exp, label='Experimental', linestyle='--')
ax.set_xlabel('time, min', fontsize=10, fontweight='bold')
ax.set_ylabel('conductivity, μS/cm', fontsize=10, fontweight='bold')
ax.set(xlim=(0, max(x)), ylim=(-1, max(y)*1.2))
ax.legend()

# for i in peaks:
#     tR = x[i]
#     signal = y[i]
#     ax.text(tR, signal + .1, f'{tR:.2f}', ha='center', zorder=2)

profile = test_params.get('profile')
ax2 = ax.twinx()
ax2.plot(*zip(*profile['OH-']), color='grey', linestyle='--', zorder=1)
ax2.grid(visible=False)
ax2.set_ylabel('[KOH], mM', fontsize=10, fontweight='bold')
```

Out[25]: Text(0, 0.5, '[KOH], mM')



In [43]:

```
df0 = backgrounds['DSM-SE']
df1 = chroms['DSM-SE']
f0 = interp1d(df0['time'], df0['signal'])
f1 = interp1d(df1['time'], df1['signal'])
f = lambda t: f1(t) - f0(t)

window = (14.2, 15.5)
t = np.linspace(*window, 10000, endpoint=True)
df = pd.DataFrame(data=dict(time=t, signal=f(t)))
ax = df.plot(x='time', y='signal')
ax.text(df['time'][df['signal'].argmax()], df['signal'].max(),
    s=round(df['time'][df['signal'].argmax()], 3),
    ha='center',
    va='bottom',)
with warnings.catch_warnings(action='ignore'):
    ax.text(0.1, .5, f'Peak Area: {round(quad(f, *window)[0], 4)}',
        transform=ax.transAxes,)
```



In [ ]:

```

```