In [1]:
```
import warnings
from pyIClab import (
    DSM_CEConstrutor, IonChromatograph, Eluent, SwitchingValve,
    Column, IonExchanger, SampleLoop, Detector, Dummy,
    )
from pyIClab import ContaminatedPhreeqcSuppressorBeta as Suppressor
from pyIClab.engines.equilibriums import find_x_LSSM
from pyIClab.beadedbag import mpl_custom_rcconfig
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from IPython.display import clear_output

sns.set()
plt.rcParams.update(mpl_custom_rcconfig)
```

In [2]:
```
class LocalConstructor(DSM_CEConstrutor):

    def set_x(self):

        kmap = self.set_kmap()

        return find_x_LSSM(kmap, -1)

```

In [3]:
```
Vinj = ['25 uL', '100 uL', '500 uL', '1000 uL', '2000 uL']

```

In [4]:
```
ic_collection = []
for i, V in enumerate(Vinj):
    eluent = Eluent.HydroxideIsocratic('18.75 mM', name=f'EG{i}')
    sp = IonExchanger.load('home_made.dat', directory='db')
    column = Column(f'Home_made{i}', length='15 cm', ID='4.6 mm')
    column.pack(sp)
    sixport = SwitchingValve.SixPort(name=f'Sixport{i}')
    suppressor = Suppressor(name=f'Suppressor{i}', kind='anion', _CO2_level=.018)
    detector = Detector(name=f'Detector{i}')
    loop = SampleLoop(name=f'Loop{i}', V=V)

    sixport.assemble(0, eluent)
    sixport.assemble([2, 5], loop)
    sixport.assemble(1, column)
    column.assemble(suppressor)
    suppressor.assemble(detector)

    ic = IonChromatograph(name=f'IC{i}', competing_ions=('OH[-1]',), lockon=sixport)
    ic_collection.append(ic)

ic_collection
```

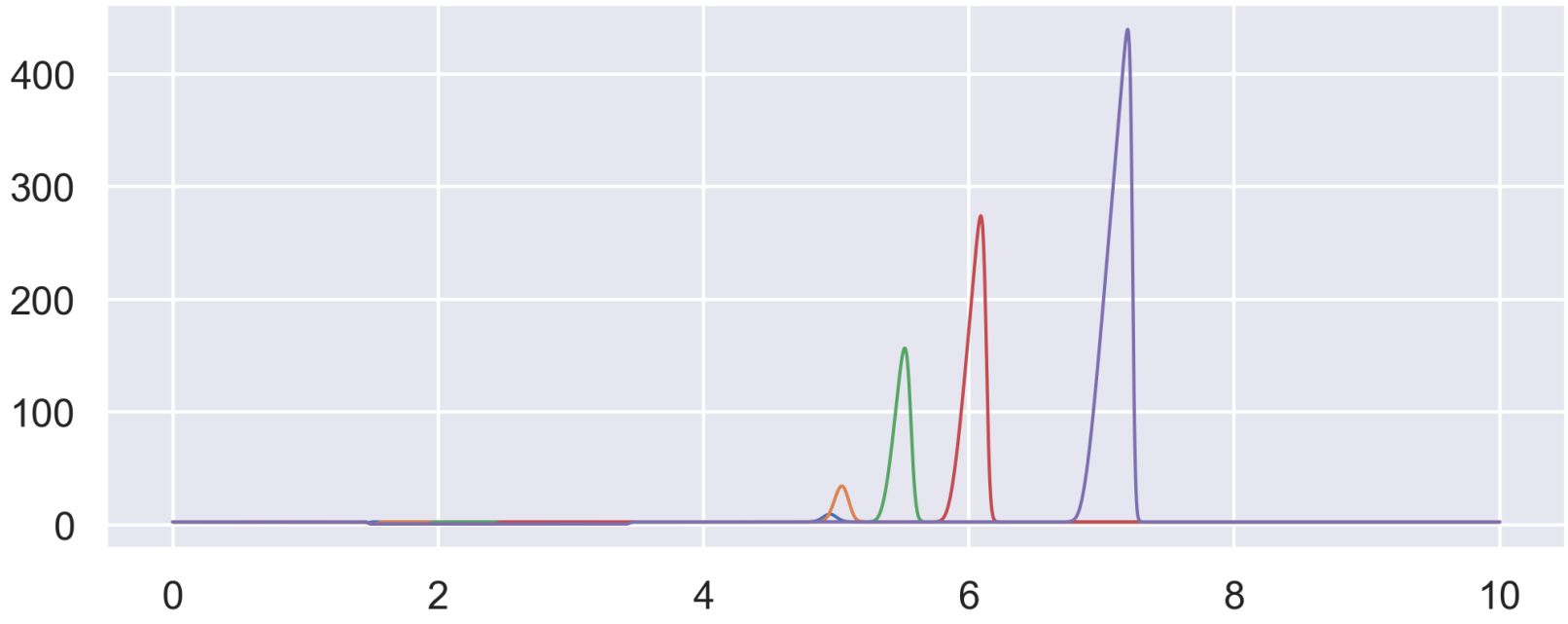
Out[4]:
```
[<IC System "IC0">,
 <IC System "IC1">,
 <IC System "IC2">,
 <IC System "IC3">,
 <IC System "IC4">]
```

In [5]:
```
for i, ic in enumerate(ic_collection):

    solution = {'Cl-': '0.1 mM'}
    ic.inject(solution, f'loop{i}')
    commands = f'''
    0.0 min, sixport{i}, inject
    '''
    ic.reset_commands(commands)
    ic.set_ModelConstructor(LocalConstructor, f'Home_made{i}')
    with warnings.catch_warnings(action='ignore'):
        ic.start(tmax='10 min')
        clear_output()
```

In [6]:
```
df_collection = []
for i, ic in enumerate(ic_collection):

    (detector,) = tuple(ic.detectors)
    df_collection.append(detector.get_signals(signal_type='conductivity'))

```

```
Calculating eluent conductivity on <Detector "Detector0">...:   0%|          | 0/5999 [00:00<?, ?it/s]
Calculating eluent conductivity on <Detector "Detector1">...:   0%|          | 0/5999 [00:00<?, ?it/s]
Calculating eluent conductivity on <Detector "Detector2">...:   0%|          | 0/5999 [00:00<?, ?it/s]
Calculating eluent conductivity on <Detector "Detector3">...:   0%|          | 0/5999 [00:00<?, ?it/s]
Calculating eluent conductivity on <Detector "Detector4">...:   0%|          | 0/5999 [00:00<?, ?it/s]
```

In [7]:
```
fig, ax = plt.subplots()
offsets = [0, 0, 0, 0, 0, 0]
for i, df in enumerate(df_collection):
    ax.plot(df['time'], df['signal']+offsets[i])
#     ax.set(ylim=(0, 1))

```



In [8]:
```
from scipy.signal import find_peaks
```

In [9]:
```
tR = []
for df in df_collection:
    (pk, *_), _ = find_peaks(df['signal']-df['signal'][0], height=2)
    tR.append(df['time'][pk])

tR
```

Out[9]:
```
[4.951155385128376,
 5.0428434478159385,
 5.516287262420807,
 6.08975441813938,
 7.196679393131044]
```

In [10]:
```
for i, df  in enumerate(df_collection):

    df.to_csv(f'effective charge ratio {Vinj[i]}-18.75NaOH.csv', index=False)
```

In [ ]:
```

```