In [1]:
```
from pyIClab import (
    DSM_CEConstrutor, IonChromatograph, Eluent, SwitchingValve,
    Column, IonExchanger, SampleLoop, Detector,
    )
from pyIClab import ContaminatedPhreeqcSuppressorBeta as Suppressor
from pyIClab.engines.equilibriums import find_x_LSSM
from pyIClab.engines.models import _total_mix
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
import warnings

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

```

In [2]:
```
def local_post_distrubute(model, /, *,
    mix_n: int,
    ):
    '''
    Just for prettier water dips.
    '''

    for _ in range(mix_n):
        _total_mix(model)

```

In [3]:
```
class LocalConstructor(DSM_CEConstrutor):

    def set_x(self):

        kmap = self.set_kmap()

        return find_x_LSSM(kmap, -1)

    def set_post_distribute(self):

        return local_post_distrubute

    def set_post_distribute_params(self):

        N = self.set_N()
        length = self.host.length.to('cm').magnitude
        target_N = round(length / 0.04)
        mix_n = round(np.log2(2*N / target_N)) + 1

        return {'mix_n': mix_n}

```

In [4]:
```
solutions = [{'Cl-1': f'{c} mM'} for c in [.1, .5, 2.5, 10, 100]]

```

In [5]:
```
ic_collection = []
for i in range(len(solutions)):
    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='25 uL')

    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[5]:
```
[<IC System "IC0">,
 <IC System "IC1">,
 <IC System "IC2">,
 <IC System "IC3">,
 <IC System "IC4">]
```

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

    solution = solutions[i]
    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()
```

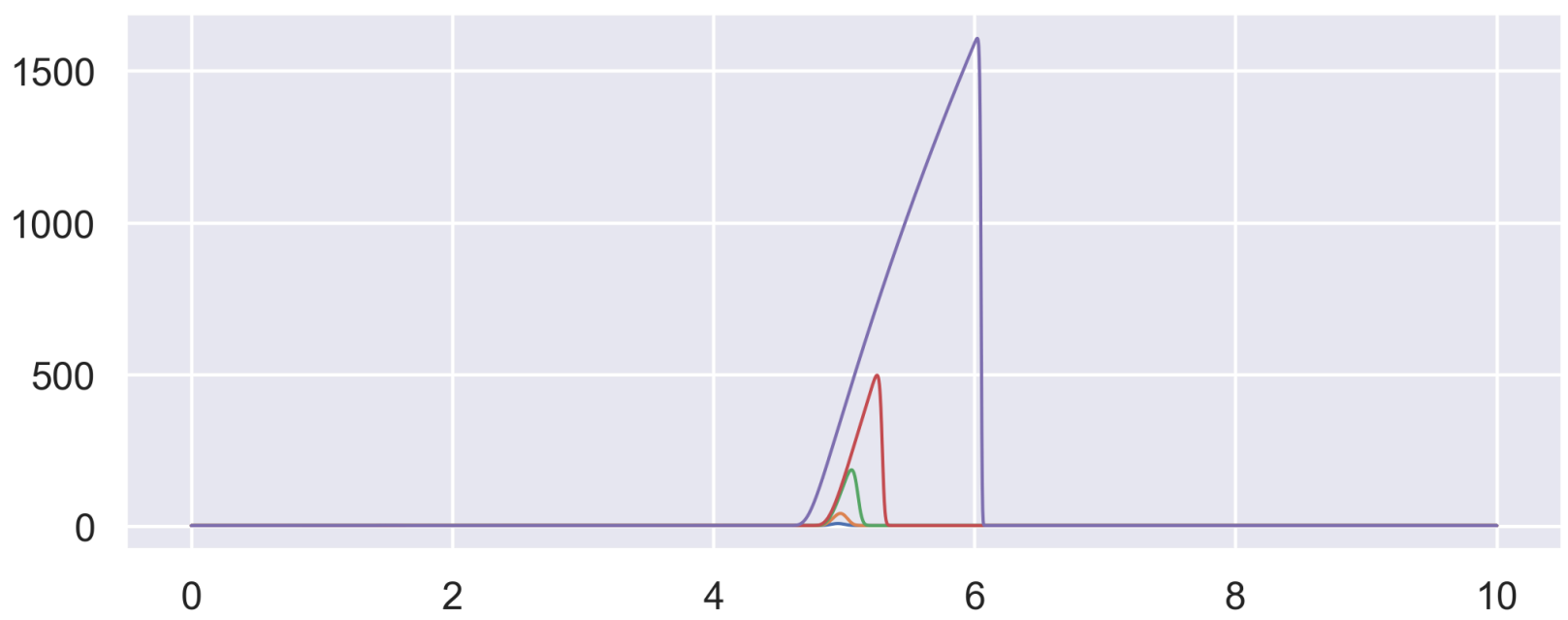
In [7]:
```
df_collection = []
for i, ic in enumerate(ic_collection):

    (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 [8]:
```
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 [9]:
```
from scipy.signal import find_peaks
tR = []
for df in df_collection:
    (pk, *_), _ = find_peaks(df['signal']-df['signal'][0], height=2)
    tR.append(df['time'][pk])

tR
```

Out[9]:
```
[4.952822440813605,
 4.9711600533511175,
 5.056179893297767,
 5.25122540846949,
 6.0197380793597866]
```

In [10]:
```
# for i, (s, df)  in enumerate(zip(solutions, df_collection)):

#     df.to_csv(f'effective charge ratio {s.copy().popitem()}-18.75NaOH.csv', index=False)
```

In [ ]:
```

```