|24.01| $MoS2+$ SC in DMSO absorbance measurements

In [ ]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import argrelextrema
```

In [ ]:

```python
df = pd.read_csv('Mos2 abs 24.csv',  skiprows=0)
df = df.iloc[1: 501, :10]
# df.iloc[]
df
```

Out[ ]:

	reference in signal	Unnamed: 1	bath_sc01_1500_45m+emp	Unnamed: 3	bath_sc005_1500_45m	Unnamed: 5	mix_sc01_1500+4000_45m
1	800	0.0001511226874	800	0.6473237276	800	0.5856873393	80
2	799	0.0001632158092	799	0.6487264037	799	0.5873479843	79
3	798	9.464941104E-005	798	0.6502585411	798	0.5891718864	79
4	797	0.0001716319966	797	0.652056396	797	0.5908094645	79
5	796	0.0001828710956	796	0.653573215	796	0.5926903486	79
...	...	...	...	...	...	...	..
496	305	-7.42346092E-005	305	3.046042442	305	4.169921398	30
497	304	1.69556406E-005	304	3.093570471	304	4.395051956	30
498	303	-0.0001188521055	303	3.137406826	303	4.585757732	30
499	302	4.745157275E-005	302	3.193946838	302	5.800307274	30
500	301	-7.164644921E-005	301	3.247264147	301	10	30

500 rows × 10 columns

In [ ]:

```python
col = [x for x in df.columns]

new_col = {}
Samp_name = []
n = 0
for i in range(len(col)):
    if i % 2 == 1:
        new_col[col[i]] = 'mes #' + str(n)
        n = n + 1
    else:
        new_col[col[i]] = col[i]
        Samp_name.append(col[i])

df2 = df.rename(new_col, axis=1)
df2.tail()
```

Out[ ]:

	reference in signal	mes #0	bath_sc01_1500_45m+emp	mes #1	bath_sc005_1500_45m	mes #2	mix_sc01_1500+4000_45m
496	305	-7.42346092E-005	305	3.046042442	305	4.169921398	305
497	304	1.69556406E-005	304	3.093570471	304	4.395051956	304
498	303	-0.0001188521055	303	3.137406826	303	4.585757732	303
499	302	4.745157275E-005	302	3.193946838	302	5.800307274	302
500	301	-7.164644921E-005	301	3.247264147	301	10	301

In [ ]:

```python
### read data with column #
init_clm = 2
# -------------
data = df2.to_numpy(dtype=float)

Samp_name = Samp_name[init_clm - 1:]

Abs = data[:, init_clm + 1 ::2]
Wave_l = data[:, init_clm ::2]

# for i in range(np.shape(Abs)[1]):
```

```
#     print(i,len(Abs[:, i]), len(Wave_l[:, i]))
# data[:, ::2]
```

In [ ]:

```
print(Samp_name)
Samp_name2 = ['bath_1h 50/20/200 1500_45m', 'bath_1h 50/10/200 1500_45m',
              'mix_2h 50/10/200 (1500+4000)_45m', 'mix_2h 50/10/200 1500_45m']
```

```
['bath_sc01_1500_45m+emp', 'bath_sc005_1500_45m', 'mix_sc01_1500+4000_45m', 'mix_sc01_1500_45m']
```

Size and conc of liquid-exfoliated nanosheets

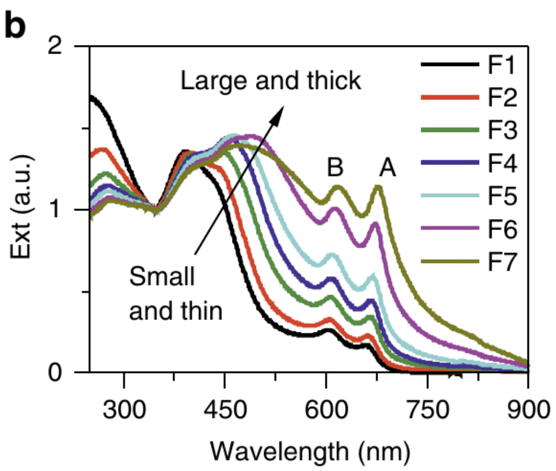
- The objective is to be able to understand the basic algorithm of finding Size and conc for MoS2 using absorbance spectra and realise the very motivation behind procedure steps "We measured extinction spectra for each fraction. To distinguish it from the true absorbance, Abs, we will refer to the extinction as Ext, where $T = 10^{-Ext}$ (T is the optical transmittance)." For clarification: Absorbance by definition:

$$A = -log_{10}T$$

According to article:

$$T = 10^{-Ext} \rightarrow Ext == A$$

Extinction spectra of the fractions normalized to the local minimum at 345 nm. The positions of the A- and B-excitons are marked.



> Extremum was found not via second derivative of Extinction spectra.
>
> The point is consdered maximum (minimum) if N nearest points are greater_equal then (less_eaual then) than the maximum

In [ ]:

```
def getExt(Abs_min, Wav_min, c, tolerance=10, ):
    # c = np.greater_equal  or  np.less_equal
    Abs_minS = np.array([])
    Wav_minS = np.array([])

    for i in range(np.shape(Abs_min)[1]):
        ind = argrelextrema(
            Abs_min[:, i], comparator = c, order=tolerance)
        # print(Abs_min[:, i][ind])
        Abs_minS = np.append(Abs_minS, Abs_min[:, i][ind])
        Wav_minS = np.append(Wav_minS, Wav_min[:, i][ind])
    return Abs_minS, Wav_minS


Wav_1min = Wave_l[np.all(Wave_l < 400, axis=1), :]
Abs_1min = Abs[np.all(Wave_l < 400, axis=1), :]

Abs_1minS, Wav_1minS = getExt(Abs_1min, Wav_1min, c = np.less_equal)
```

In [ ]:

```
# ---------- get B max ----------
Wav_to650 = Wave_l[np.all(Wave_l < 650, axis=1), :]
Abs_to650 = Abs[np.all(Wave_l < 650, axis=1), :]

Wav_550_650 = Wav_to650[np.all(Wav_to650 > 580, axis=1), :]
Abs_550_650 = Abs_to650[np.all(Wav_to650 > 580, axis=1), :]

Abs_Bmax, Wav_Bmax = getExt(Abs_550_650, Wav_550_650, c=np.greater_equal, tolerance=80)
```

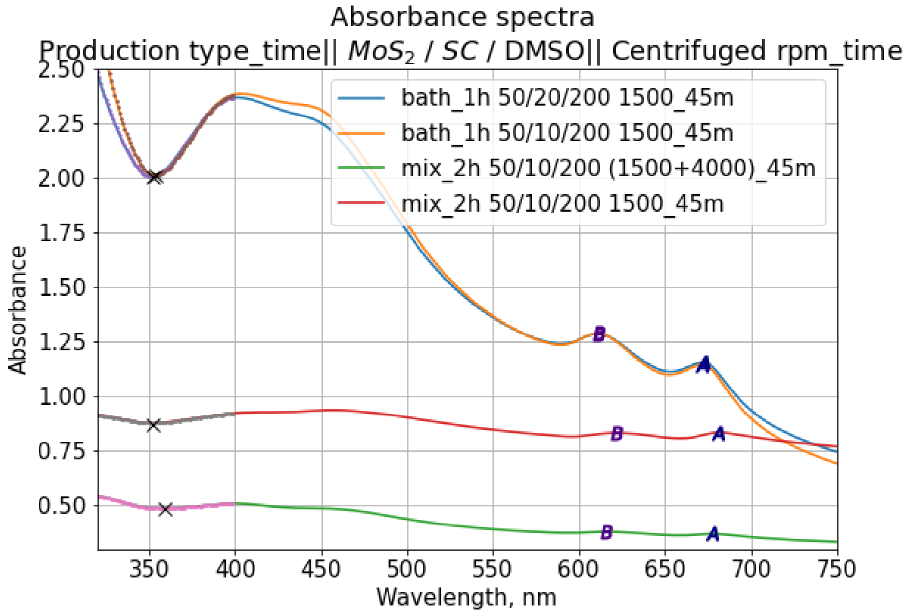
In [ ]:

```
# ---------- get A max ----------
Wav_to700 = Wave_l[np.all(Wave_l < 700, axis=1), :]
Abs_to700 = Abs[np.all(Wave_l < 700, axis=1), :]

Wav_650_700 = Wav_to700[np.all(Wav_to700 > 650, axis=1), :]
Abs_650_700 = Abs_to700[np.all(Wav_to700 > 650, axis=1), :]

Abs_Amax, Wav_Amax = getExt(Abs_650_700, Wav_650_700, c=np.greater_equal, tolerance=80)
```

In [ ]:

```
plt.rcParams.update(
    {'font.size': 15, 'lines.markersize': 0, 'lines.linewidth': 1.6, 'axes.grid': True, 'lines.marker': '.'})
figsize = (9, 6)
# ------plot spectra
fig, ax = plt.subplots(figsize=figsize)
ax.plot(Wave_l, Abs)


ax.plot(Wav_1min, Abs_1min, marker='.', markersize=3, linewidth=0)
ax.plot(Wav_1minS, Abs_1minS, marker='x', c='black', markersize=9, linewidth=0)
ax.plot(Wav_Bmax, Abs_Bmax, marker='$B$',
        c='indigo', markersize=9, linewidth=0)
ax.plot(Wav_Amax, Abs_Amax, marker='$A$', c='navy', markersize=9, linewidth=0)


ax.set_xlim(320, 750)
ax.set_ylim(0.3, 2.5)

# ax.set_title('$MoS_2\ /\ SC $ in DMSO Absorbance spectra')
ax.set_title('Absorbance spectra \n Production type_time|| $MoS_2\ /\ SC$ / DMSO|| Centrifuged rpm_time')
ax.set_xlabel('Wavelength, nm')
ax.set_ylabel('Absorbance')
ax.legend(Samp_name2)
```

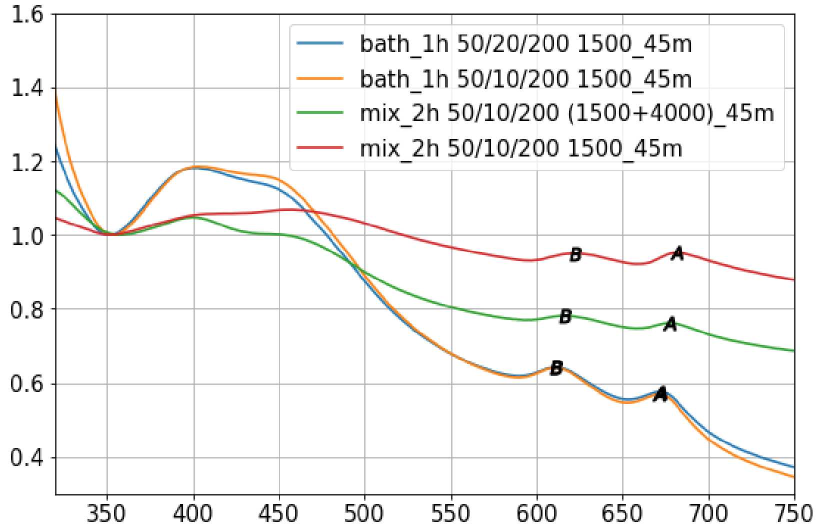
Out[ ]: <matplotlib.legend.Legend at 0x7f934fded5e0>



In [ ]:

```
fig, ax = plt.subplots(figsize=figsize)
# Normilizing at local minima around 350
Abs_norm = Abs / Abs_1minS
B_to_350 = Abs_Bmax / Abs_1minS
A_to_350 = Abs_Amax / Abs_1minS

ax.plot(Wave_l, Abs_norm)
ax.plot(Wav_Bmax, B_to_350, marker='$B$',
        c='black', markersize=9, linewidth=0)
ax.plot(Wav_Amax, A_to_350, marker='$A$',
        c='black', markersize=9, linewidth=0)

ax.set_xlim(320, 750)
ax.set_ylim(0.3, 1.6)
ax.legend(Samp_name2)
```

Out[ ]: <matplotlib.legend.Legend at 0x7f934fec4220>



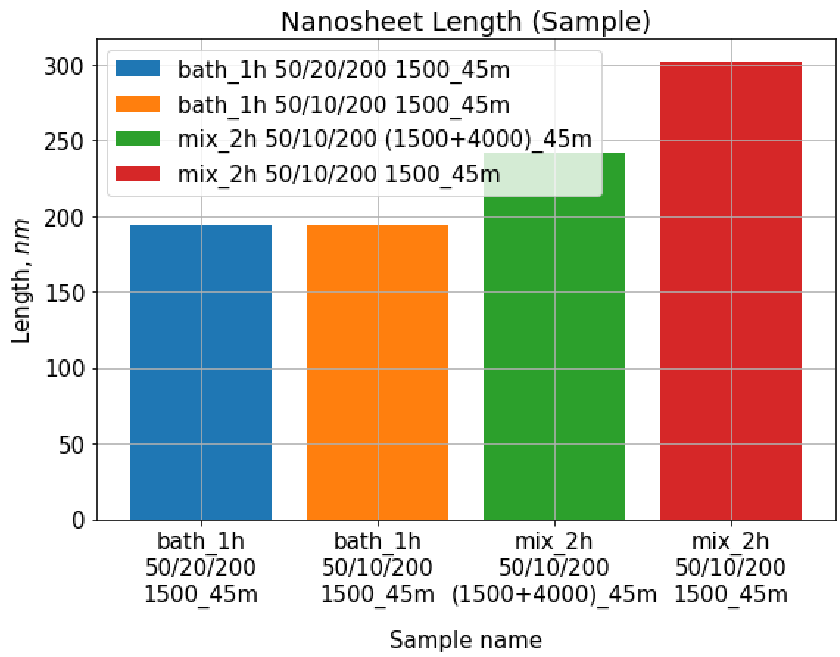
Determine nonsheets length

$$L(\mathrm{nm}) = \frac{3.5\mathrm{Ext_B}/\mathrm{Ext_{345}} - 0.14}{11.5 - \mathrm{Ext_B}\,/\,\mathrm{Ext_{345}}} \cdot 1000$$

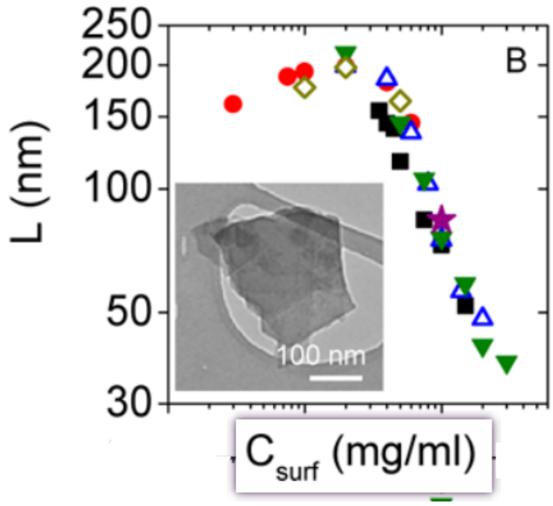
In [ ]:

```
def L(rat):
    return (3.5 * rat - 0.14) * 1000 / (11.5 - rat)
    # returns value in nanomiters


plt.rcParams.update(
    {'font.size': 15, 'lines.markersize': 10, 'lines.linewidth': 0, 'axes.grid': True, 'lines.marker': 'o'})

fig, ax = plt.subplots(figsize=figsize)
oxName = [x.replace(' ', '\n') for x in Samp_name2]
# oxName = ['\n'.join(x.split(' ',)[:-1]) for x in Samp_name2]

for i in range(len(B_to_350)):
    # ax.plot(np.arange(len(B_to_350))[i], L(B_to_350)[i])
    ax.bar(oxName[i], L(B_to_350)[i])

ax.set_title(
    'Nanosheet Length (Sample)')
ax.set_xlabel('Sample name', labelpad = 15)
ax.set_ylabel('Length, $ nm$')
ax.legend(Samp_name2)
```

Out[ ]: <matplotlib.legend.Legend at 0x7f934fcb7220>



Coleman for the reference

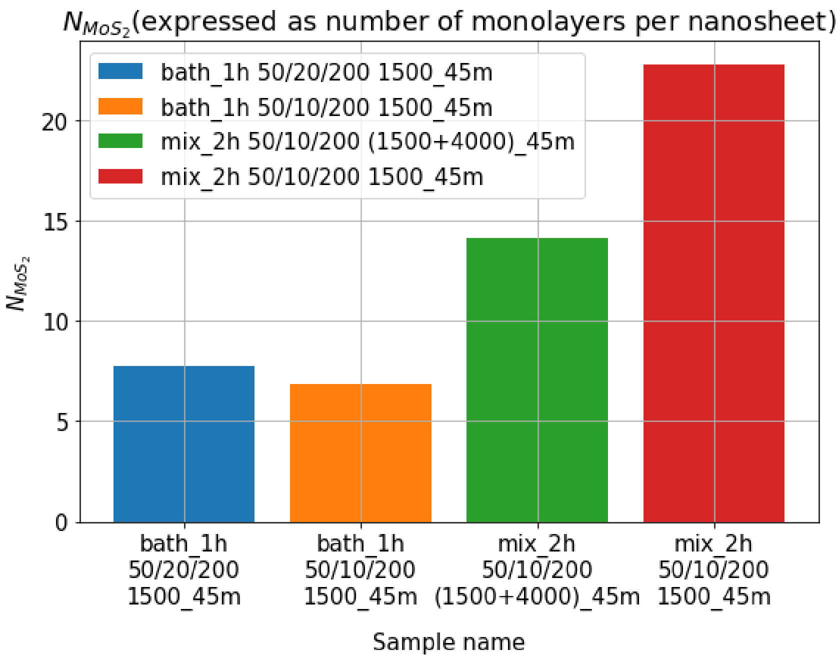


Determine nanosheet thickness:

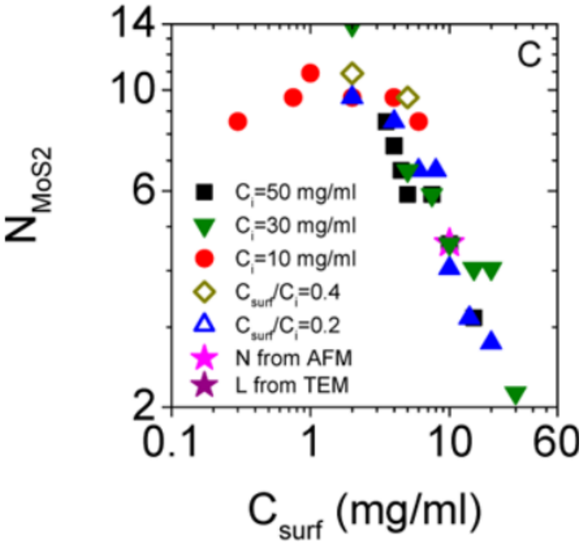
$$N_{\mathrm{MoS_2}} = 2.3 \times 10^{36} e^{-54888/\lambda_A} = \frac{\text{number of monolayers}}{\text{number of nanosheets}}$$

In [ ]:
```
def Thick(WL_A):
    return 2.3 * 10**36 * np.exp(-54888 / WL_A)


plt.rcParams.update(
    {'font.size': 15, 'lines.markersize': 10, 'lines.linewidth': 0, 'axes.grid': True, 'lines.marker': 'o'})

fig, ax = plt.subplots(figsize=figsize)

for i in range(len(Wav_Amax)):
    ax.bar(oxName[i], Thick(Wav_Amax)[i])

ax.set_title('$N_{MoS_2}$(expressed as number of monolayers per nanosheet)')
ax.set_xlabel('Sample name', labelpad=15)
ax.set_ylabel('$N_{MoS_2}$')
ax.legend(Samp_name2)
```

Out[ ]: <matplotlib.legend.Legend at 0x7f934fbc6400>



Coleman reference



C
N_{MoS2}
14
10
6
2
0.1
1
10
60
C_{surf} (mg/ml)
C_i=50 mg/ml
C_i=30 mg/ml
C_i=10 mg/ml
C_{surf}/C_i=0.4
C_{surf}/C_i=0.2
N from AFM
L from TEM

In [ ]: