

PDS_FTPS_data

January 2, 2026

```
[1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.signal import find_peaks
from scipy import ndimage
import scipy.constants as sci

sn_nk_colours = np.array([[0.399249, 0.368802, 0.539323, 1.    ],
                           [0.522175, 0.390367, 0.55673 , 1.    ],
                           [0.659176, 0.4005  , 0.56346 , 1.    ],
                           [0.793152, 0.443565, 0.598539, 1.    ],
                           [0.848157, 0.570831, 0.706555, 1.    ],
                           [0.884708, 0.699078, 0.812861, 1.    ]])

Pb_nk_colours = np.array([[0.491568, 0.232214, 0.207943, 1.    ],
                           [0.746299, 0.298576, 0.286641, 1.    ],
                           [0.876507, 0.438212, 0.310225, 1.    ],
                           [0.909375, 0.588853, 0.321654, 1.    ],
                           [0.940913, 0.740544, 0.341539, 1.    ]])
```

1 PDS data

- We truncated the data by hand to remove non-linear / noisy parts of the data near lower values
- This is stored in a csv where values past our determined threshold are set to 0
- We take the derivative of the PDS data and use scipy find peaks
- We'll stitch at a point to the left of the derivative peak the length of the FWHM

```
[2]: def convert_to_float(val):
    try:
        return float(val)
    except ValueError:
        return np.nan
Br_content = [0, 25, 50, 75, 100]
PDS_data_array = np.loadtxt("PDS/all_raw.csv", delimiter=',')
PDS_data_array_zeroed = np.loadtxt("PDS/all_zeroed.csv", delimiter=',')
```

```

for i, br in enumerate(Br_content):
    wavel = 1e9 * sci.h * sci.c / (PDS_data_array[:, 2 * i] * sci.e)
    plt.plot(wavel, PDS_data_array[:, 2 * i + 1], color=Pb_nk_colours[i],
    ↪linestyle='--')
    plt.plot(wavel[PDS_data_array_zeroed[:, 2 * i + 1] != 0],
    ↪PDS_data_array_zeroed[:, 2 * i + 1][PDS_data_array_zeroed[:, 2 * i
    ↪+ 1] != 0],
    ↪color=Pb_nk_colours[i], label=f"Br = {br}%")

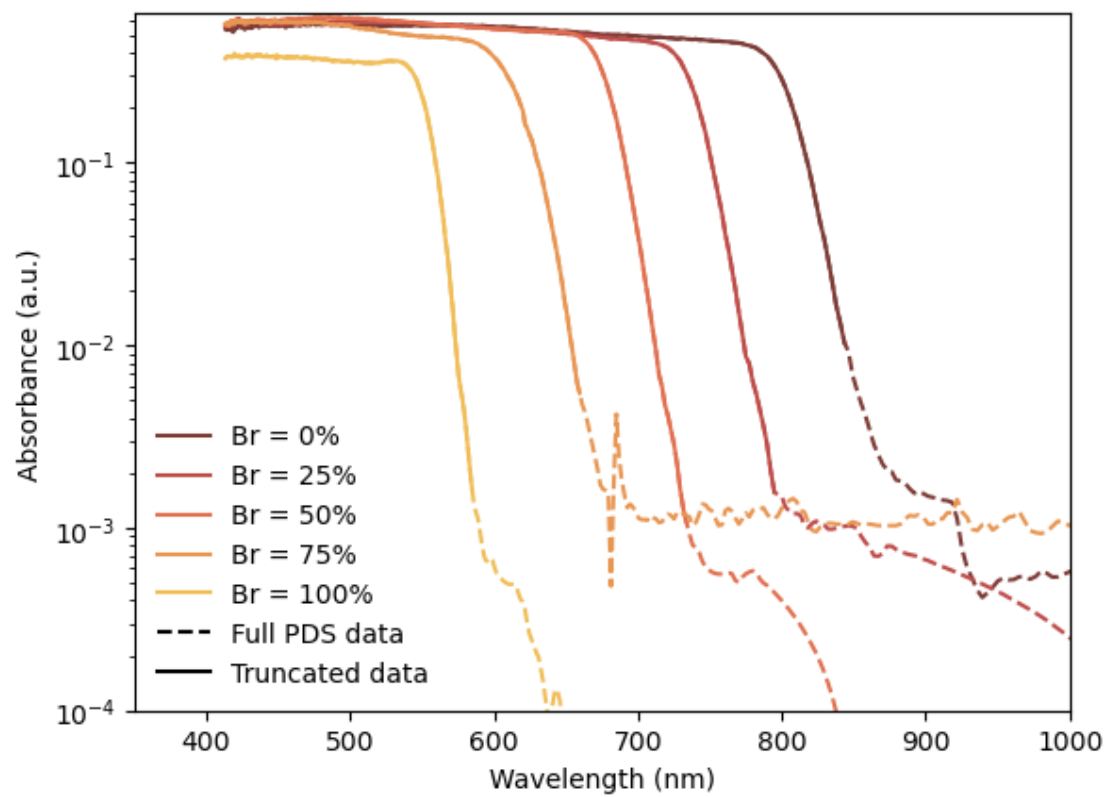
# Finalize plot
plt.xlim((350, 1000))
plt.ylim(1e-4)
plt.plot([], [], 'k--', label='Full PDS data')
plt.plot([], [], 'k', label='Truncated data')
plt.xlabel("Wavelength (nm)")
plt.ylabel("Absorbance (a.u.)")
plt.legend(frameon=False)
plt.yscale('log')
plt.show()

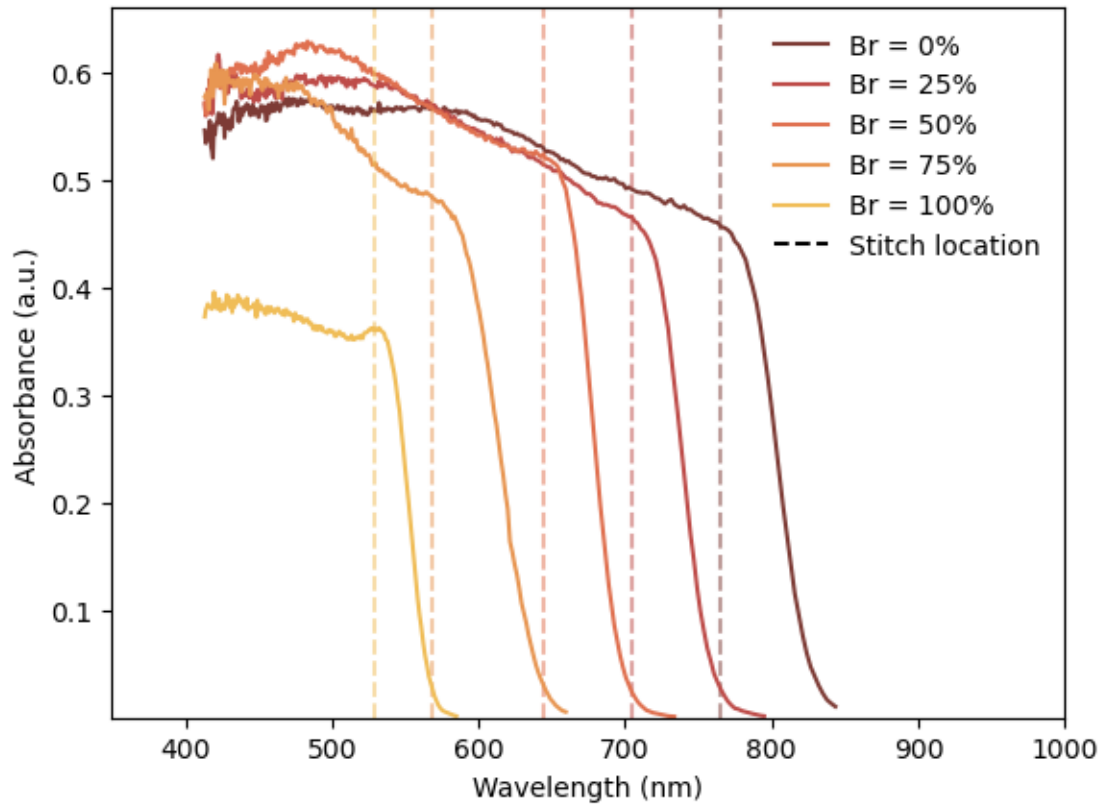
for i, br in enumerate(Br_content):
    wavel = 1e9 * sci.h * sci.c / (PDS_data_array[:, 2 * i] * sci.e)
    alpha = PDS_data_array_zeroed[:, 2 * i + 1]

    # Identify inflection point of the tail
    peaks, properties = find_peaks(-ndimage.gaussian_filter(np.diff(alpha), 3),
    ↪width=10, prominence=0.01)
    if peaks.size > 0:
        left_ips = properties["left_ips"][0]
        adjusted_index = int(peaks[0] - abs(left_ips - peaks[0]) * 2)
        stitch_location = wavel[adjusted_index]

        valid_indices = alpha != 0
        plt.plot(wavel[valid_indices], alpha[valid_indices],
    ↪color=Pb_nk_colours[i], label=f"Br = {br}%")
        plt.axvline(stitch_location, color=Pb_nk_colours[i], linestyle='--',
    ↪alpha=0.5)
plt.xlim((350, 1000))
plt.ylim(1e-4)
plt.plot([], [], 'k--', label='Stitch location')
plt.xlabel("Wavelength (nm)")
plt.ylabel("Absorbance (a.u.)")
plt.legend(frameon=False)
plt.show()

```





2 FTPS Data

- Multiple samples were measured to avoid device-specific issues in one sample
- One outlier was discarded
- Otherwise, final EQE traces created by averaging the data for all the measurements

```
[3]: xls_FTPS = pd.ExcelFile('FTPS/Pb-Sn series Oct 24 EQE data.xlsx')
outlier = '94-P6'

FTPS_df = {'Wavel': [], 'Sn%': [], 'Data': []}
for sheet_name in xls_FTPS.sheet_names:
    Sn_percent = float(sheet_name.split(' ')[-1].split('-')[-1])*10
    # We found 70% and 60% was basically the same
    if Sn_percent > 65:
        continue
    df = pd.read_excel(xls_FTPS, sheet_name=sheet_name)
    if outlier in df.columns:
        df = df.drop(columns=[outlier])

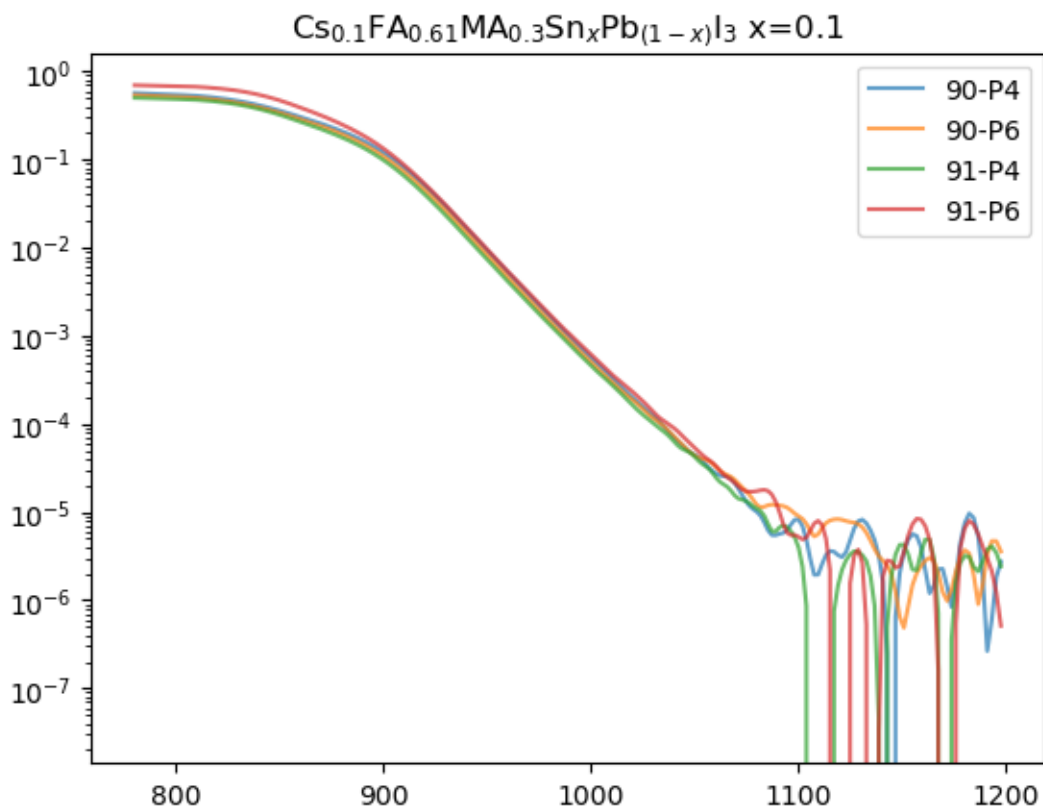
holded = []
```

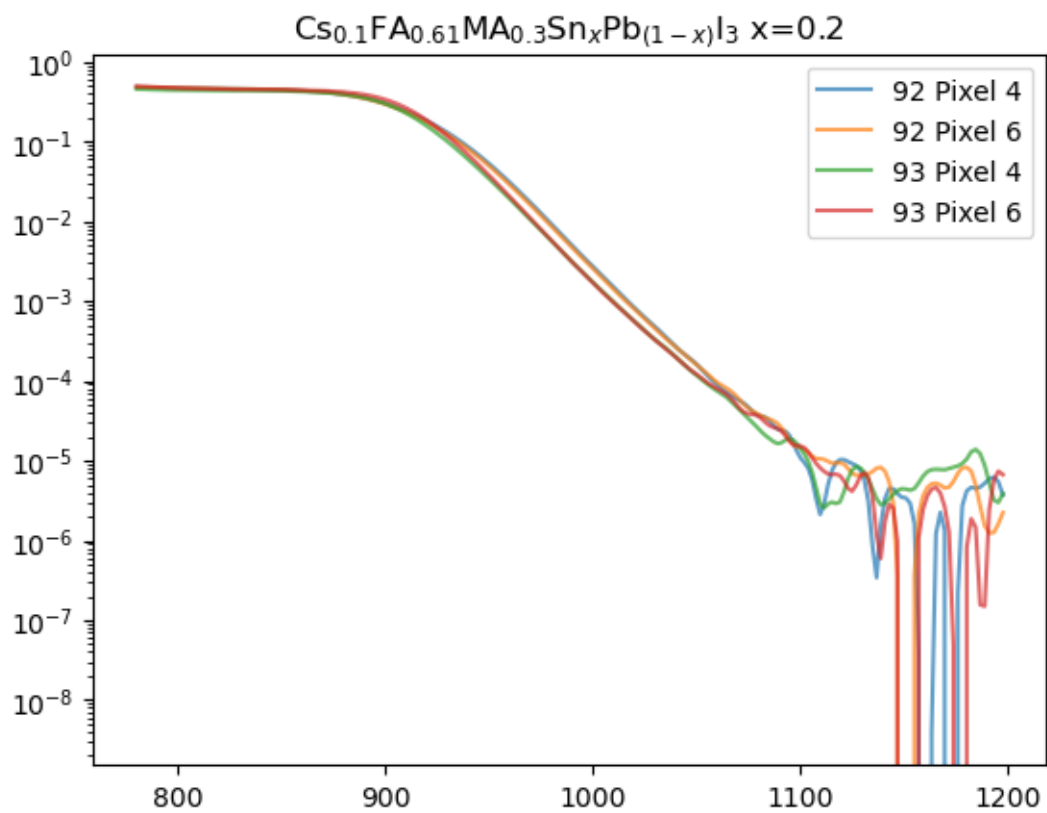
```

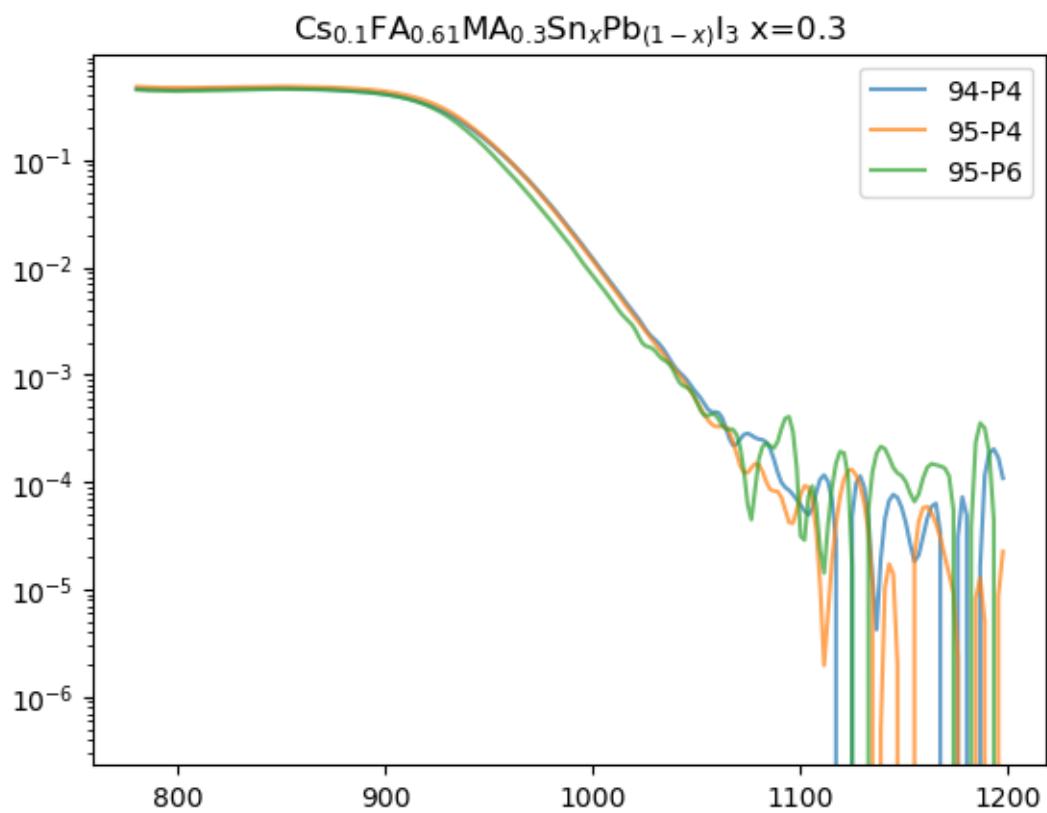
for col in df.columns[1:]:
    plt.plot(df['Wavelength'], df[col], label=f'{col}', alpha=0.7)
    holded.append(df[col])
mean_vals = np.mean(holded, axis=0)
title = title = "Cs$_{0.1}$FA$_{0.61}$MA$_{0.3}$Sn$_{x}$Pb$_{(1-x)}$I$_3$"+f" x={int(Sn_percent)/100}"
plt.title(title)
plt.yscale('log')
plt.legend()
plt.show()
FTPS_df['Wavel'].append(df['Wavelength'])
FTPS_df['Sn%'].append(Sn_percent)
FTPS_df['Data'].append(mean_vals)

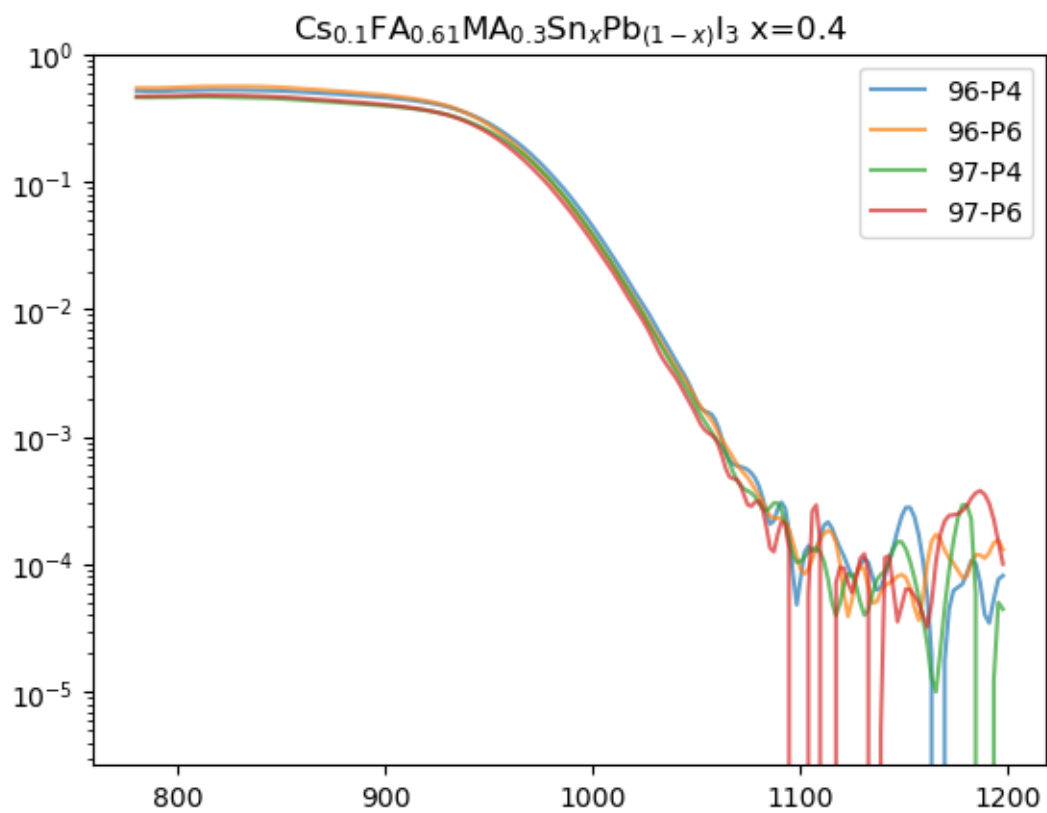
df_FTPS = pd.DataFrame(FTPS_df)

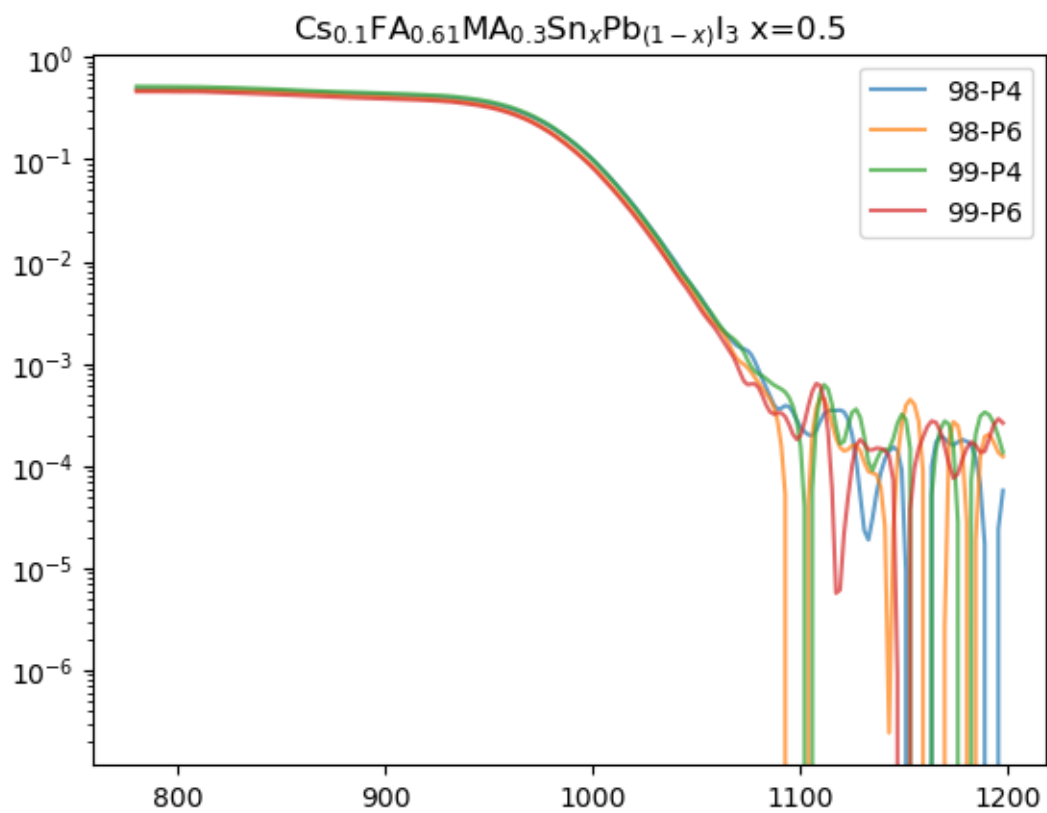
```

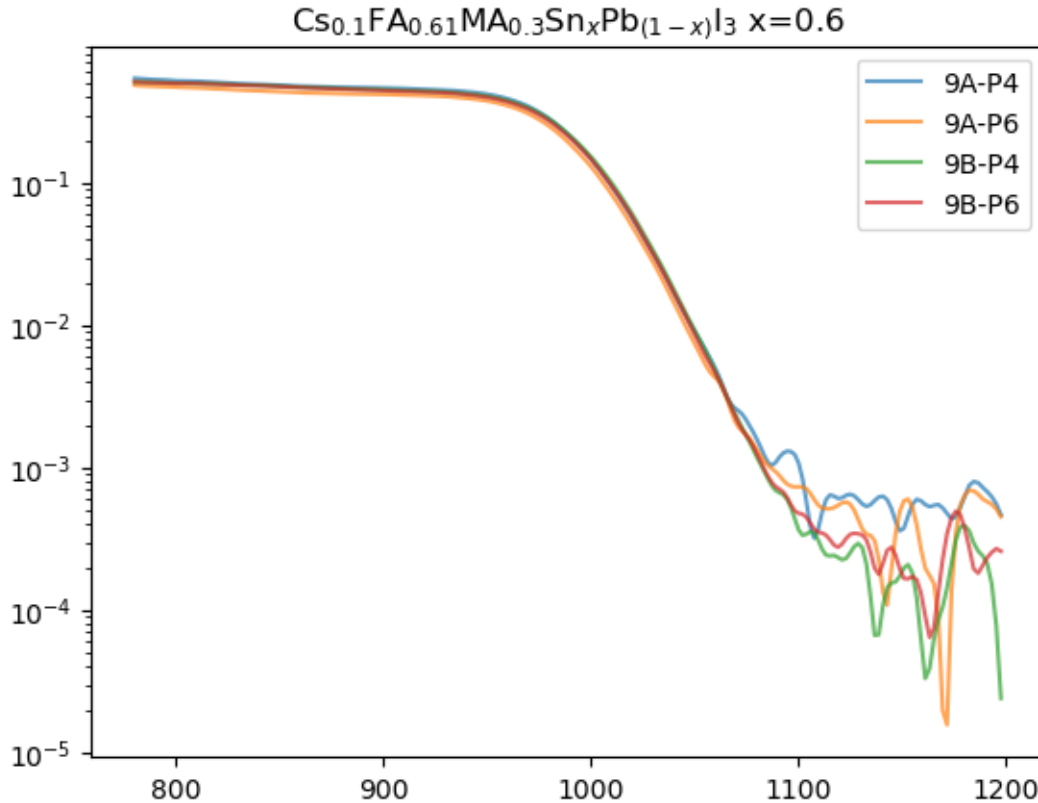












- We truncated the data by hand to remove non-linear / noisy parts of the data near lower values
- This is stored in a csv where values past our determined threshold are set to 0
- In this case we still did the derivative analysis as above but we chose to take only the slope bit, as the shape of the edge here is likely to be affected by other device-specific absorptions
 - Practically this means stitching at the left location of the FWHM rather than peak - FWHM length

```
[4]: FTPS_data_array = np.loadtxt("FTPS/PbSn_zeroed.csv", delimiter=',')
for i, row in df_FTPS.iterrows():
    plt.plot(row['Wavel'], row['Data'], linestyle = '--',
    color=sn_nk_colours[i])

    wavel_zeroed = FTPS_data_array[:,2*i]
    alpha_zeroed = FTPS_data_array[:,2*i+1]
    plt.plot(wavel_zeroed[alpha_zeroed!=0], alpha_zeroed[alpha_zeroed!=0],
    color=sn_nk_colours[i],
    label=f"Sn = {row['Sn%']}%")
plt.plot([], [], 'k--', label='Full averaged FTPS data')
```

```

plt.plot([], [], 'k', label='Truncated data')
plt.xlabel('Wavelength (nm)')
plt.ylabel('EQE (%)')
plt.legend(frameon = False)
plt.yscale('log')
plt.show()

for i, row in df_FTPS.iterrows():
    wavel_zeroed = FTPS_data_array[:,2*i]
    alpha_zeroed = FTPS_data_array[:,2*i+1]
    plt.plot(wavel_zeroed[alpha_zeroed!=0], alpha_zeroed[alpha_zeroed!=0],
    color=sn_nk_colours[i],
            label=f"Sn = {row['Sn%']}%")
    # Interpolate to uniform base to avoid issues later
    wavel = np.linspace(min(wavel_zeroed[alpha_zeroed!=0]),
    max(wavel_zeroed[alpha_zeroed!=0]), 1000)
    alpha = np.interp(wavel, wavel_zeroed[alpha_zeroed!=0],
    alpha_zeroed[alpha_zeroed!=0])

    peaks, properties = find_peaks(-1*(ndimage.gaussian_filter(np.
    diff(alpha),3)), width=10, prominence=00.001)
    stitch_location = wavel[int(properties["left_ips"][0]*0.9)]

    plt.axvline(stitch_location, color=sn_nk_colours[i], linestyle='--',
    alpha=0.5)

plt.plot([], [], 'k--', label='Stitch location')
plt.xlabel('Wavelength (nm)')
plt.ylabel('EQE (%)')
plt.xlim(800, 1100)
plt.legend(frameon = False)
plt.show()

```

