Lorentzian_Final

November 17, 2022

1 Objective: Given a Gold nanocube, obtain its wavelength (energy) scans 10 times consecutively via AOTF-VIS. Following this:

1.0.1 1. Plot the 10 consecutive scans on a single graph, taking wavelength (nm) in the x-axis and corrected extinction (a.u) in the y-axis.

1.0.2 2. Extract the regions in all the scans where a local peak is observed. Plot them together in the same manner as above, using energy (eV) in the x-axis instead.

1.0.3 3. Obtain the best-fit Lorentzian curve for all the scans. Extract the respective Full Width at Half Maximum (FWHM) and peak Resonant Energy (E_{res}) for all of the scans. Plot all the best-fits together.

1.0.4 4. Plot all the best-fits together but ensure that the curves are normalised w.r.t. peak corrected extinction.

1.0.5 5. Obtain ΔFWHM i.e. the FWHM of all scans relative to that of the 1^{st} scan. Plot these values w.r.t. E_{res}.

1.0.6 6. Obtain ΔE_{res} i.e. the peak Resonant Energy of all the scans relative to that of the 1^{st} scan. Plot these values w.r.t. E_{res}.

1.1 Relevant packages and modules required

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.constants as sc
from pylab import cm
from lmfit.models import LorentzianModel
from tabulate import tabulate
```

1.2 Function to extract our parameters of interest as well as the number of repeats for the respective scans from the data files

[2]:
```python
def parameter_formatting_WL(filename, r_skip=84, n_rows=3, loc=2):
    user_col = ["Wavelength(nm)", "FocusCorrectionY[µm]", "Repeat(x)", "M(V)", "Phase(')", "IS(A)", "Phase(').1", "R(V)", "Phase(').2"]
    readfile = pd.read_csv(filename, delim_whitespace=1, skiprows=r_skip, names=user_col)
    stepsfile = pd.read_csv(filename, sep='\t', skiprows=9, nrows=n_rows)
    repeats_WL = int(stepsfile.iloc[loc]['Loops:'][-8:-6])
    repeats_M = 1
    if n_rows == 4:
        repeats_M=int(stepsfile.iloc[0]['Loops:'][-9:-6])
    repeats=[repeats_WL, repeats_M]
    return readfile, repeats
```

1.3 Function to generate extinction arrays for the given csv file

[3]:
```python
def extinction(readfile, repeats):
    extinctions = readfile.loc[:,"R(V)"]/readfile.loc[:,"M(V)"]
    extinctions = extinctions.groupby(np.arange(len(extinctions))//repeats[0]).mean()
    return extinctions
```

1.4 Function to store extinction data for the consecutive scans as a 2D array i.e. Scan 1 data will be in column 1, Scan 2 in column 2, and so on

[4]:
```python
def ext_matrix(extinctions, repeats):
    n_rows = len(extinctions)//repeats[1]
    n_cols = repeats[1]
    extin_matrix=np.zeros((n_rows, n_cols))
    for i in range(n_cols):
        loc = n_rows*i
        extin_matrix[:,i] = extinctions[loc:(loc+n_rows)]
    return extin_matrix
```

1.5 Function that finds the corrected extinctions for all the scans by utilising the reference data

[5]:
```python
def corr_ext(extin_matrix, extin_ref):
    n_rows, n_cols = extin_matrix.shape
    corr_extin_matrix = np.zeros((n_rows, n_cols))
    for i in range(n_cols):
        corr_extin_matrix[:,i] = 1.00 - (extin_matrix[:,i]/extin_ref)
    return corr_extin_matrix
```

1.6 Function that prepares the extinction 2D array as well as the wavelengths for subsequent plotting as per Objective 1

[6]:
```
def data_prep_ob1(wavelengths, corr_extin_matrix):
    n_rows, n_cols = corr_extin_matrix.shape
    xData = np.zeros((n_cols, n_rows))
    yData = np.zeros((n_cols, n_rows))
    for i in range(n_cols):
        xData[i,:] = wavelengths
        yData[i,:] = corr_extin_matrix[:,i]
    return xData, yData
```

1.7 Function that yields the plot as per Objective 1 and 2

[7]:
```
def plot_ob1_2(xData = [], yData = [], xLabel = '', yLabel = 'Extinction (a.u.
↪)', Title = '', f = ''):
    n_rows, n_cols = yData.shape
    colors = cm.get_cmap('viridis', n_rows)
    plt.figure(num = None, figsize = (10, 7), facecolor = 'w', edgecolor = 'k')
    for i in range(n_rows):
        plt.plot(xData[i,:], yData[i,:], label = "Scan "+str(i+1), color =␣
↪colors(i))
        plt.xlabel(xLabel)
        plt.ylabel(yLabel)
        plt.legend(loc='upper right')
        plt.title(Title)
        plt.legend(bbox_to_anchor=(0.99, 0.99),  borderaxespad=0)
    plt.show
    plt.savefig(f)
```

1.8 Function that converts the wavelength (in nm) to energy (in eV)

[8]:
```
def lambda_to_E(wavelengths):
    return (sc.h*sc.c*10**9/sc.e)/(wavelengths)
```

1.9 Function that yields the wavelength slice of the expected peak as well as the corresponding extinction data slice

[9]:
```
def data_slice(xData, yData, left_thres = 650.0, right_thres = 755.0):
    n_rows, n_cols = yData.shape
    xSlice = np.zeros((n_rows, n_cols))
    ySlice = np.zeros((n_rows, n_cols))
    for i in range(n_rows):
        begin_index = np.nonzero(xData[i,:] == left_thres)[0][0]
        end_index = np.nonzero(xData[i,:] == right_thres)[0][0]
        fit_range = slice(begin_index, end_index+1)
```

```
        xSlice[i,fit_range] = xData[i,fit_range]
        ySlice[i,fit_range] = yData[i,fit_range]
    return xSlice[:, ~np.all(xSlice == 0, axis = 0)], ySlice[:, ~np.all(ySlice␣
↪== 0, axis = 0)]
```

1.10 Function that obtains the best-fit Lorentzian curves for the consecutive data slices provided

[10]:
```
def lorentzian_fit(xSlice, ySlice):
    n_rows, n_cols = ySlice.shape
    model = LorentzianModel()
    result_params = []
    result_bestfit = np.zeros((n_rows, n_cols))
    for i in range(n_rows):
        params = model.guess(ySlice[i,:], xSlice[i,:])
        results = model.fit(ySlice[i,:], params, x=xSlice[i,:])
        result_params.append(list(results.params.items()))
        result_bestfit[i,:] = results.best_fit
    return result_params, result_bestfit
```

1.11 Function that extracts the model best-fit parameters held in a dictionary and stores it in a 2D array

[11]:
```
def param_dict_array(params):
    n_rows = len(params)
    no_of_params = len(params[0])
    param_set = np.zeros((n_rows, no_of_params))
    for i in range(n_rows):
        for j in range(no_of_params):
            param_set[i,j] = params[i][j][1].value
    return param_set
```

1.12 Function that extracts the FWHM, E_{res}, and height of peak maxima for the 10 consecutive scans

[12]:
```
def param_extract(param_set):
    E_res = param_set[:,1]
    fwhm = param_set[:,3]
    height_max_peak = param_set[:,4]
    return fwhm, E_res, height_max_peak
```

1.13 Function that tabulates the FWHM, ΔFWHM, E_{res}, ΔE_{res}, and Height of peak maxima for the 10 consecutive scans

[13]:
```
def param_table(fwhm, delta_fwhm, E_res, delta_E_res, h_max):
    num_el = len(fwhm)
    table = [['Scan no.', 'FWHM (eV)', 'ΔFWHM (meV)', 'E_res (eV)', 'ΔE_res␣
↪(meV)', 'Height of peak maxima (a.u.)']]
    for i in range(num_el):
        row = ['Scan '+str(i+1), fwhm[i], delta_fwhm[i]*1000, E_res[i],␣
↪delta_E_res[i]*1000, h_max[i]]
        table.append(row)
    print(tabulate(table, headers='firstrow', tablefmt='fancy_grid', floatfmt=".
↪4f"))
    table_df=pd.DataFrame(table)
    table_df.to_csv('Parameters_of_interest.csv', index=False, header=False)
```

1.14 Function that normalizes the extinction dataset as well as the best-fit extinction dataset w.r.t. height of peak maxima for the 10 consecutive scans as per Objective 4

[14]:
```
def extinction_normalize(yData, yFit, h_max):
    n_rows, n_cols = yData.shape
    yData_norm = np.zeros((n_rows, n_cols))
    yFit_norm = np.zeros((n_rows, n_cols))
    for i in range(n_rows):
        yData_norm[i,:] = yData[i,:]/h_max[i]
        yFit_norm[i,:] = yFit[i,:]/h_max[i]
    return yData_norm, yFit_norm
```

1.15 Function that plots the best-fit Lorentzian curves along with the corresponding datasets (as dots) for the 10 consecutive scans as per Objective 3 and 4

[15]:
```
def plot_ob3_4(xData = [], yData = [], yFit = [], xLabel = '', yLabel =␣
↪'Extinction (a.u.)', Title = '', f = ''):
    n_rows, n_cols = yData.shape
    colors = cm.get_cmap('viridis', n_rows)
    plt.figure(num = None, figsize = (10, 7), facecolor = 'w', edgecolor = 'k')
    for i in range(n_rows):
        plt.plot(xData[i,:], yFit[i,:], '-', label = "Scan "+str(i+1), color =␣
↪colors(i))
        plt.plot(xData[i,:], yData[i,:], '.', color = colors(i))
        plt.xlabel(xLabel)
        plt.ylabel(yLabel)
        plt.legend(loc='upper right')
        plt.title(Title)
```

```
        plt.legend(bbox_to_anchor=(0.99, 0.99),  borderaxespad=0)
    plt.show
    plt.savefig(f)
```

1.16 Function that finds ΔFWHM and ΔE_{res} as per Objective 5 and 6

[16]:
```
def delta_param(fwhm, E_res):
    n_rows = len(fwhm)
    delta_fwhm = []
    delta_E_res = []
    for i in range(n_rows):
        delta_fwhm.append((fwhm[i] - fwhm[0]))
        delta_E_res.append((E_res[i] - E_res[0]))
    return delta_fwhm, delta_E_res
```

1.17 Function that plots ΔFWHM (meV) and ΔE_{res} (meV) versus E_{res} (eV) for the 10 consecutive scans as per Objective 5 and 6

[17]:
```
def plot_ob5_6(xData = [], yData = [], xLabel = '', yLabel = '', Title = '', f␣
 ↪= '', loci = 'best'):
    n_rows = len(yData)
    colors = cm.get_cmap('viridis', n_rows)
    plt.figure(num = None, figsize = (10, 7), facecolor = 'w', edgecolor = 'k')
    for i in range(n_rows):
        plt.plot(xData[i], yData[i]*1000, '.', label = "Scan "+str(i+1), color␣
 ↪= colors(i))
        plt.xlabel(xLabel)
        plt.ylabel(yLabel)
        plt.legend(loc=loci)
        plt.title(Title)
        #plt.legend(bbox_to_anchor=(0.99, 0.99),  borderaxespad=0)
    plt.show
    plt.savefig(f)
```

1.18 Function that tabulates the ΔFWHM (meV) and ΔE_{res} (meV) for the 10th scan

[18]:
```
def scan10_table(delta_fwhm, delta_E_res):
    table = [['ΔFWHM_cube (meV)', 'ΔE_res_cube (meV)'], [delta_fwhm[-1]*1000,␣
 ↪delta_E_res[-1]*1000]]
    print(tabulate(table, headers='firstrow', tablefmt='fancy_grid', floatfmt=".
 ↪4f"))
    plt.savefig('array.png')
    table_df=pd.DataFrame(table)
    table_df.to_csv('Cube_parameters.csv', index=False, header=False)
```

1.19 Main function that executes everything required from start to finish

[19]:
```
def Nanocube_Lorentzian(f1 = "10ConsScan_cube9.csv", f2 = "10ConsScan_cube9ref.
↪csv"):

    hits, repeats1 = parameter_formatting_WL(f1, 86, 4, 3)
    miss, repeats2 = parameter_formatting_WL(f2, 84, 3, 2)

    hits_ext = extinction(hits, repeats1)
    miss_ext = extinction(miss, repeats2)

    hits_ext_matrix = ext_matrix(hits_ext, repeats1)

    corr_hits_ext_matrix = corr_ext(hits_ext_matrix, miss_ext)

    wavelengths = miss.loc[::repeats2[0],'Wavelength(nm)'].values

    xdata, ydata = data_prep_ob1(wavelengths, corr_hits_ext_matrix)

    plot_ob1_2(xdata, ydata, 'Wavelength (nm)', 'Extinction (a.u.)',␣
↪'Consecutive Wavelength Scans on Nanocube', 'Full_wavelength_scan.png')

    xslice1, yslice = data_slice(xdata, ydata)
    xslice = lambda_to_E(xslice1)

    plot_ob1_2(xslice, yslice, 'Energy (eV)', 'Extinction (a.u.)', 'Consecutive␣
↪Wavelength Scans on Nanocube', 'Sliced_energy_scan.png')

    fit_params, fit_ydata = lorentzian_fit(xslice, yslice)
    fit_param_set = param_dict_array(fit_params)

    FWHM, E_res, height_max_peak = param_extract(fit_param_set)

    plot_ob3_4(xslice, yslice, fit_ydata, 'Energy (eV)', 'Extinction (a.u.)',␣
↪'Best-fit Lorentzian Curves for Consecutive Scans',␣
↪'Best-fit_Lorentzian_with_dataset.png')
    plot_ob1_2(xslice, fit_ydata, 'Energy (eV)', 'Extinction (a.u.)', 'Best-fit␣
↪Lorentzian Curves for Consecutive Scans', 'Best-fit_Lorentzian.png')

    yslice_norm, fit_ydata_norm = extinction_normalize(yslice, fit_ydata,␣
↪height_max_peak)

    plot_ob3_4(xslice, yslice_norm, fit_ydata_norm, 'Energy (eV)', 'Extinction␣
↪(a.u.)', 'Normalized Best-fit Lorentzian Curves for Consecutive Scans',␣
↪'Best-fit_Lorentzian_with_dataset_normalized.png')
```

```
    plot_ob1_2(xslice, fit_ydata_norm, 'Energy (eV)', 'Extinction (a.u.)',␣
↪'Normalized Best-fit Lorentzian Curves for Consecutive Scans',␣
↪'Best-fit_Lorentzian_normalized.png')

    delta_FWHM, delta_E_res = delta_param(FWHM, E_res)

    plot_ob5_6(E_res, delta_FWHM, 'E_res (eV)', 'ΔFWHM (meV)', 'Change in FWHM␣
↪vs E_res', 'Delta_FWHM_vs_E-res.png')
    plot_ob5_6(E_res, delta_E_res, 'E_res (eV)', 'ΔE_res (meV)', 'Change in␣
↪E_res vs E_res', 'Delta_E-res_vs_E-res.png', 'lower right')

    param_table(FWHM, delta_FWHM, E_res, delta_E_res, height_max_peak)
    scan10_table(delta_FWHM, delta_E_res)
```

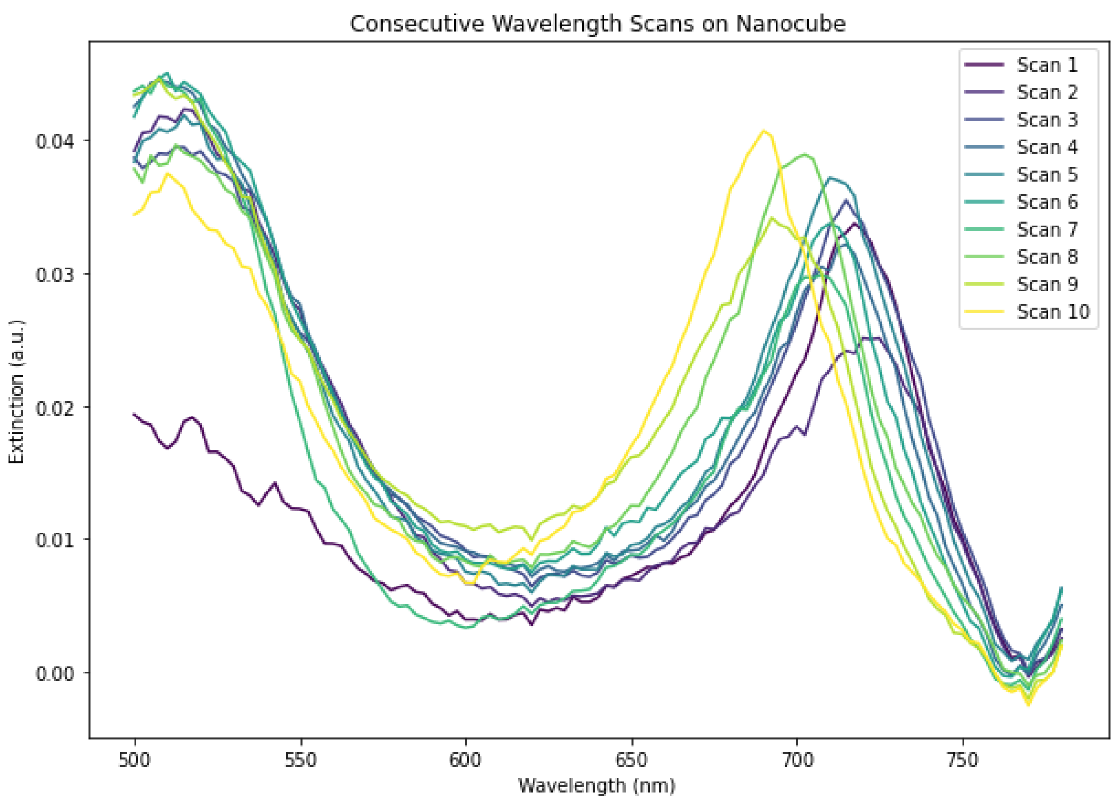
1.20 Just call the main function!

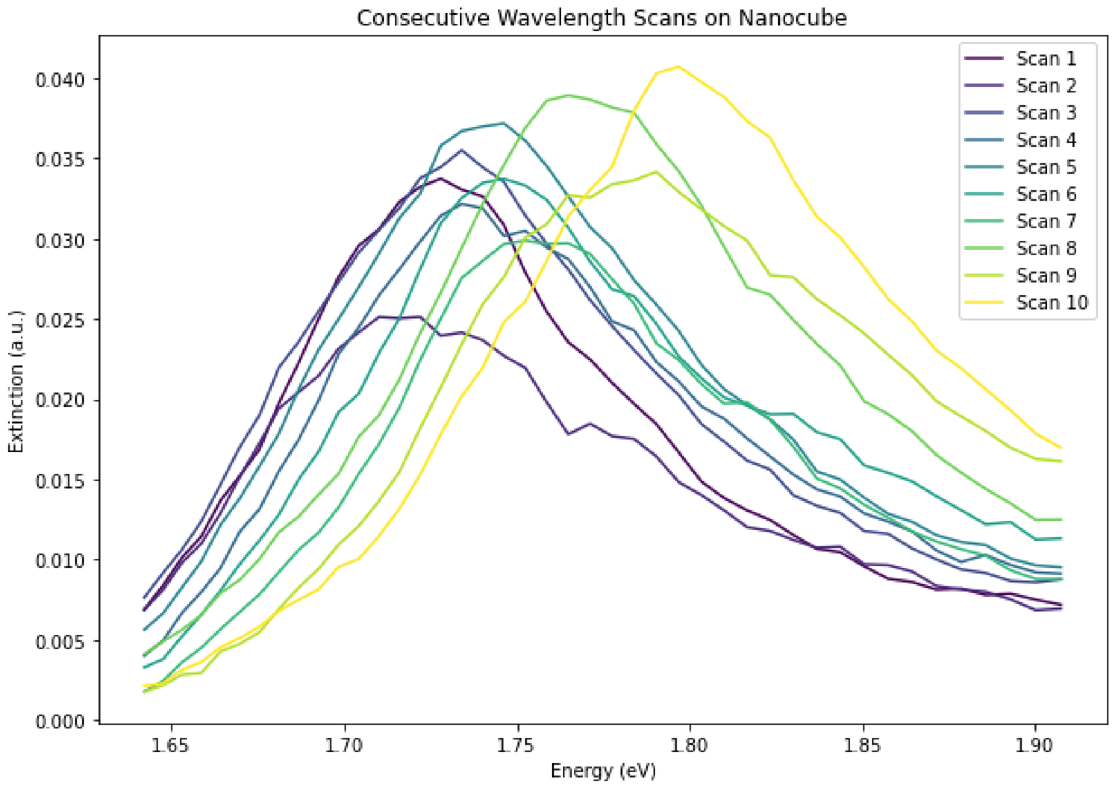
[20]:
```
Nanocube_Lorentzian()
```

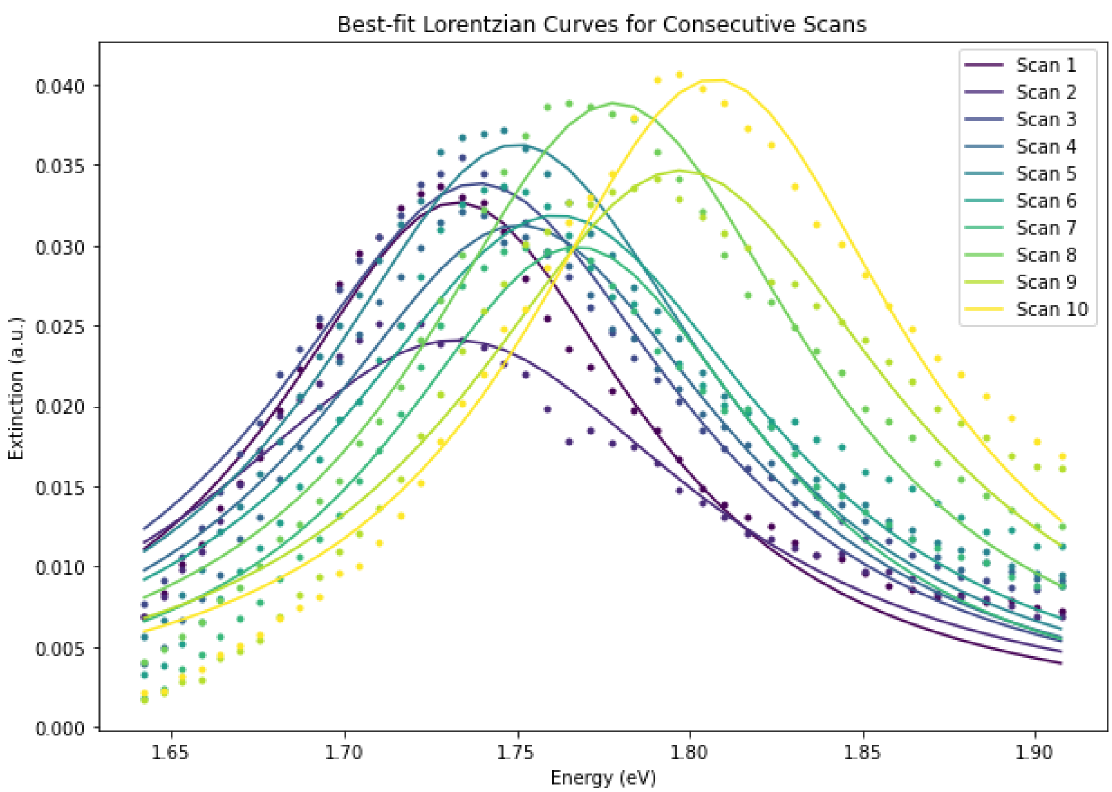
```
 Scan no.       FWHM (eV)     ΔFWHM (meV)     E_res (eV)     ΔE_res (meV)
Height of peak maxima (a.u.)


 Scan 1            0.1299          0.0000          1.7328           0.0000
0.0327


 Scan 2            0.1724         42.4778          1.7324          -0.3949
0.0241


 Scan 3            0.1463         16.3927          1.7387           5.9443
0.0338


 Scan 4            0.1463         16.3442          1.7507          17.9328
0.0313


 Scan 5            0.1417         11.7528          1.7500          17.2166
0.0363


 Scan 6            0.1517         21.7167          1.7614          28.6154
0.0319
```

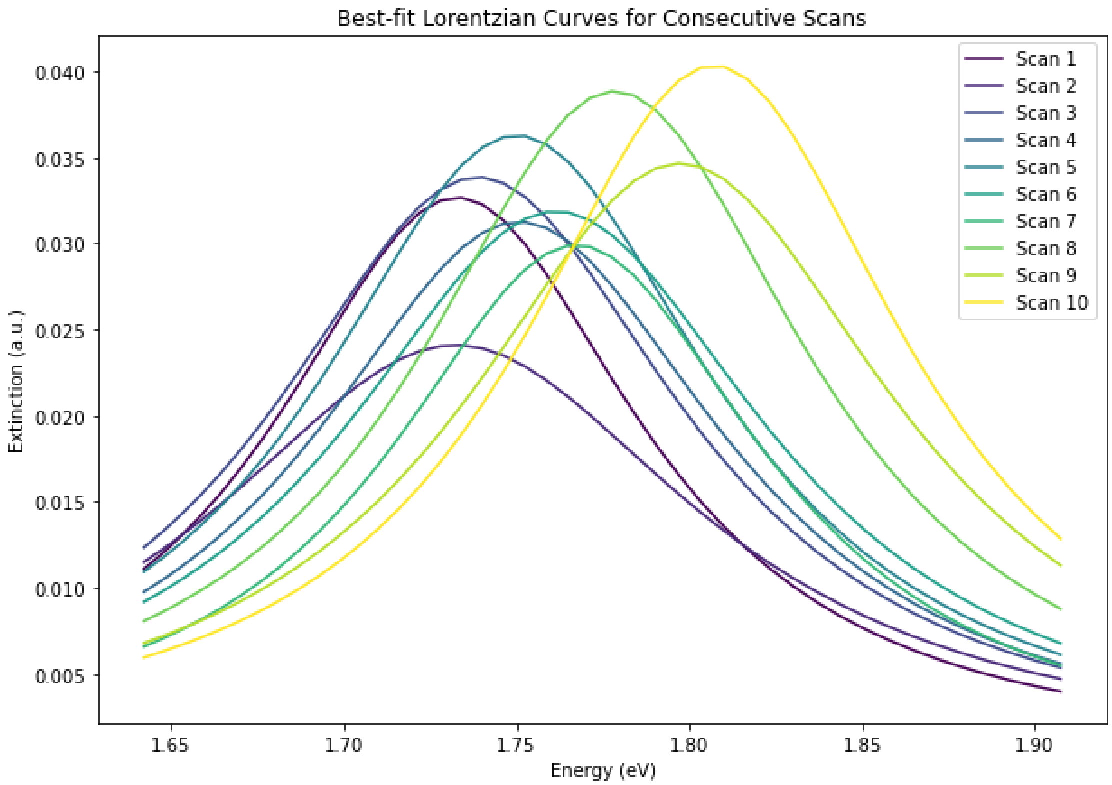
Scan 7	0.1329	2.9923	1.7673	34.4895	0.0299
Scan 8	0.1394	9.5031	1.7784	45.5542	0.0388
Scan 9	0.1531	23.2088	1.7974	64.5580	0.0347
Scan 10	0.1372	7.2526	1.8071	74.2891	0.0403

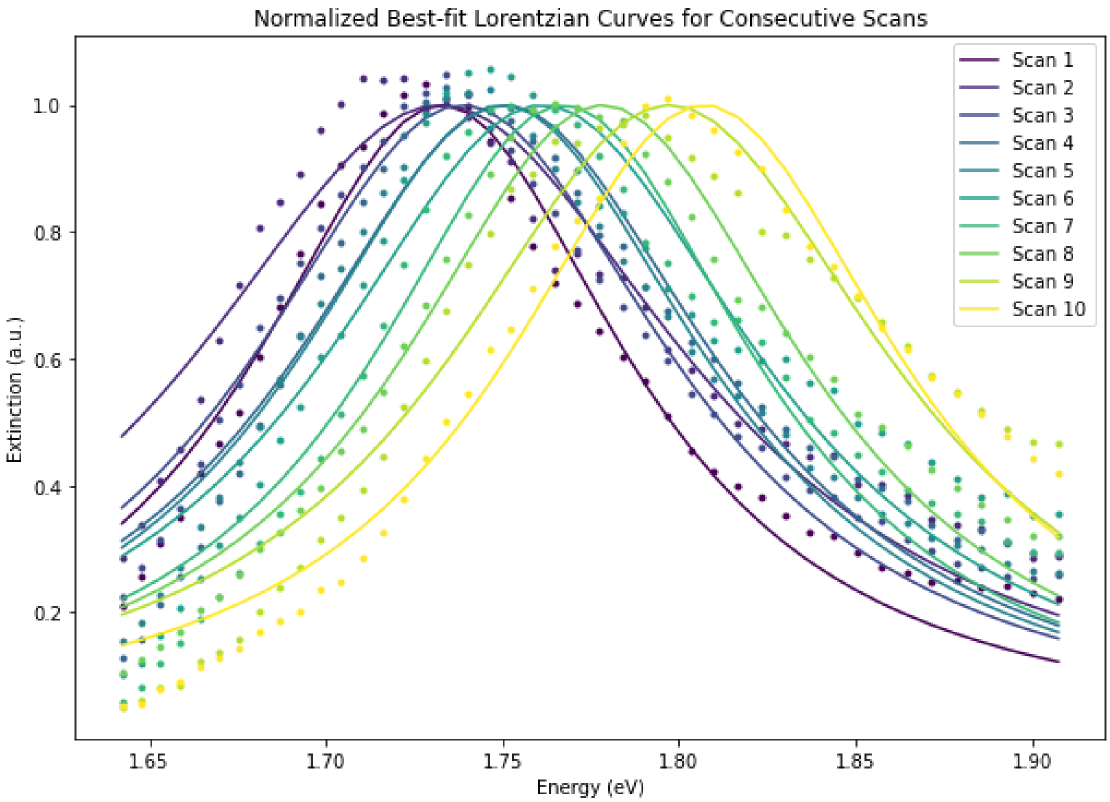
ΔFWHM_cube (meV)	ΔE_res_cube (meV)
7.2526	74.2891

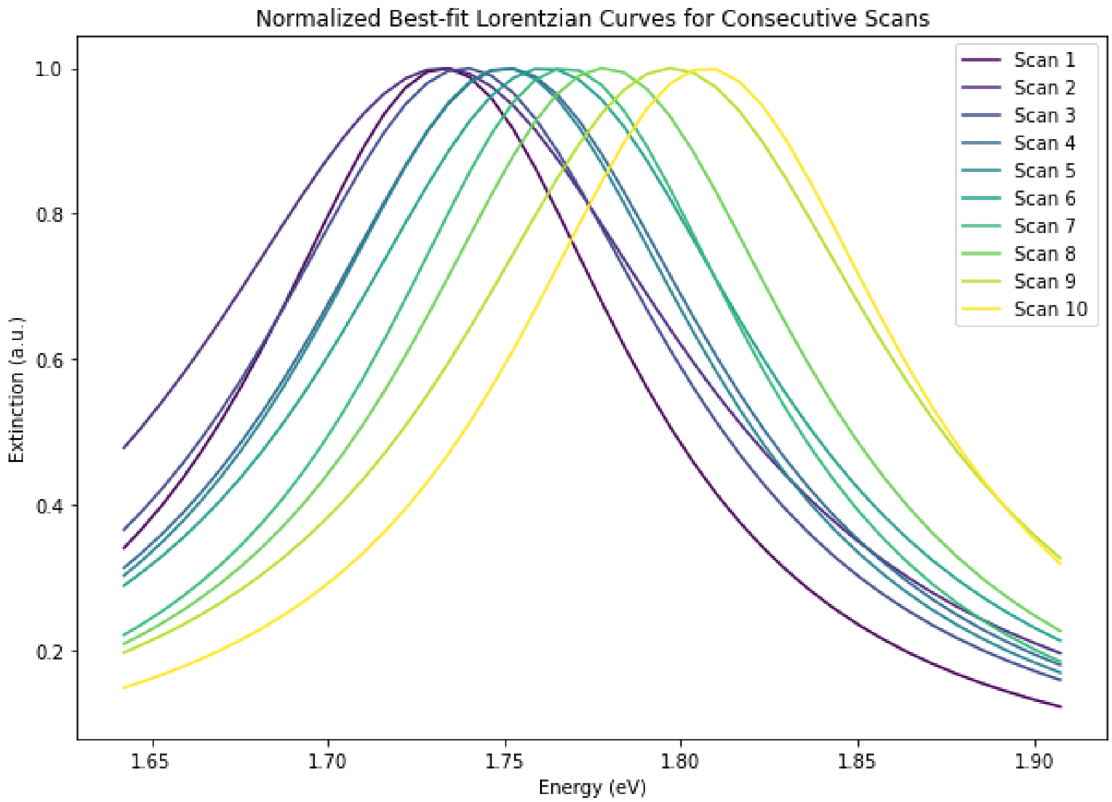


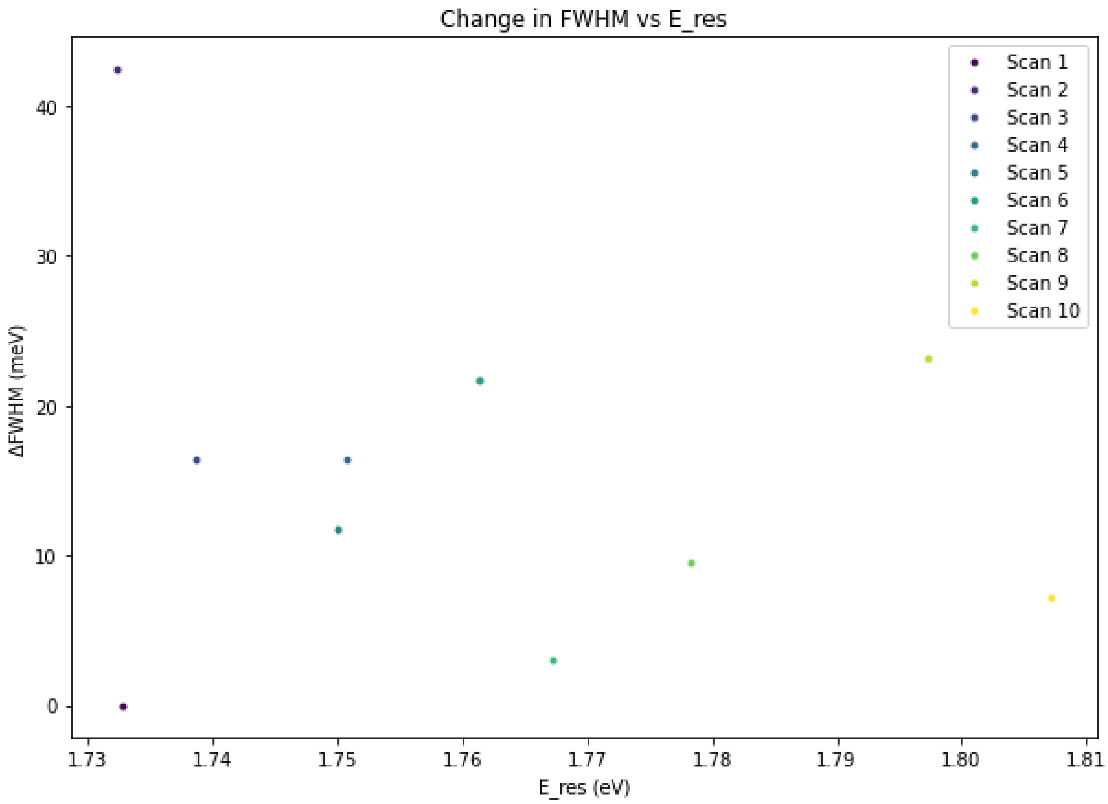

Consecutive Wavelength Scans on Nanocube
Extinction (a.u.)
Energy (eV)
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
1.65
1.70
1.75
1.80
1.85
1.90
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10

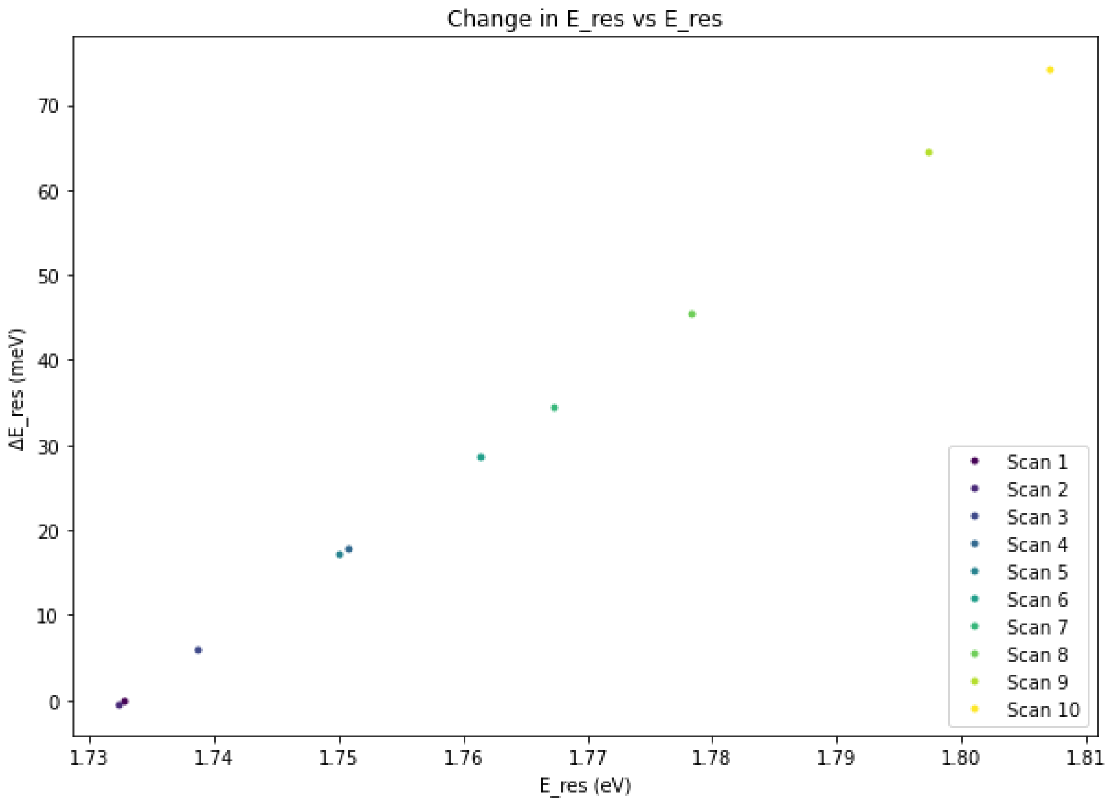

Best-fit Lorentzian Curves for Consecutive Scans
Extinction (a.u.)
Energy (eV)
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
1.65
1.70
1.75
1.80
1.85
1.90
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10


Best-fit Lorentzian Curves for Consecutive Scans
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
Extinction (a.u.)
1.65
1.70
1.75
1.80
1.85
1.90
Energy (eV)


Normalized Best-fit Lorentzian Curves for Consecutive Scans
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10
Extinction (a.u.)
Energy (eV)
1.0
0.8
0.6
0.4
0.2
1.65
1.70
1.75
1.80
1.85
1.90


Normalized Best-fit Lorentzian Curves for Consecutive Scans
Extinction (a.u.)
Energy (eV)
1.0
0.8
0.6
0.4
0.2
1.65
1.70
1.75
1.80
1.85
1.90
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10


Change in FWHM vs E_res
ΔFWHM (meV)
E_res (eV)
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10
0
10
20
30
40
1.73
1.74
1.75
1.76
1.77
1.78
1.79
1.80
1.81


Change in E_res vs E_res
ΔE_res (meV)
E_res (eV)
0
10
20
30
40
50
60
70
1.73
1.74
1.75
1.76
1.77
1.78
1.79
1.80
1.81
Scan 1
Scan 2
Scan 3
Scan 4
Scan 5
Scan 6
Scan 7
Scan 8
Scan 9
Scan 10