Time-series electron diffraction data analysis

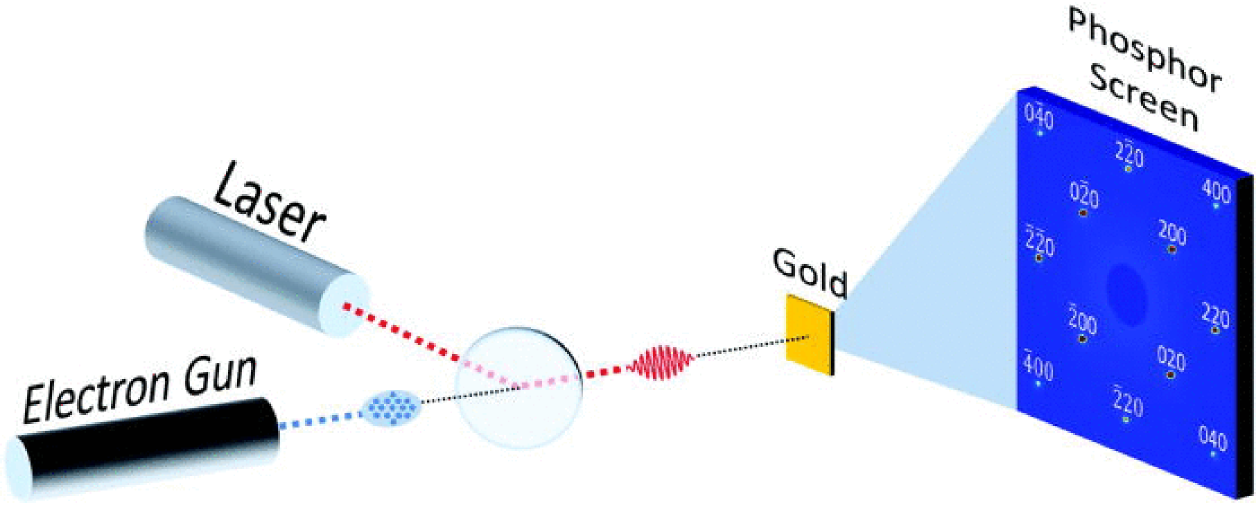
By John Feng

Demonstration of data analysis skills

What is Ultrafast Electron Diffraction?

Ultrafast electron diffraction is a technique used to probe the time-series structural dynamics of molecular samples. In essence, an ultrafast electron pulse illuminates a diffractive sample, then we can interpret its structure changes by analyzing the resultant diffraction pattern. The reaction of the sample is initiated by an ultrafast laser pulse. By varying the delay time between the electron pulse and laser pulse, we can capture enough diffraction images to make a “movie” of the ensuing structure changes.

This presentation will focus on the data analysis aspect of this project.

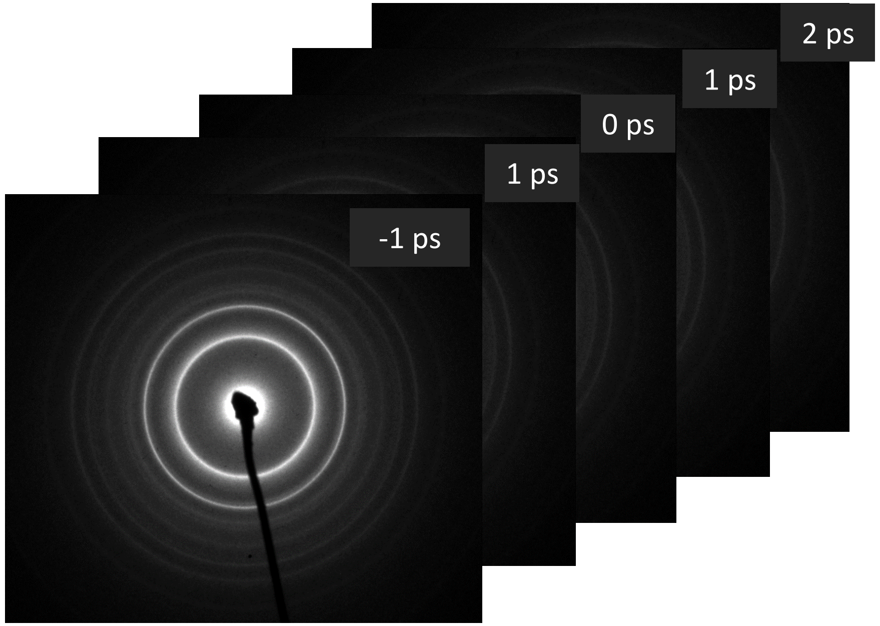


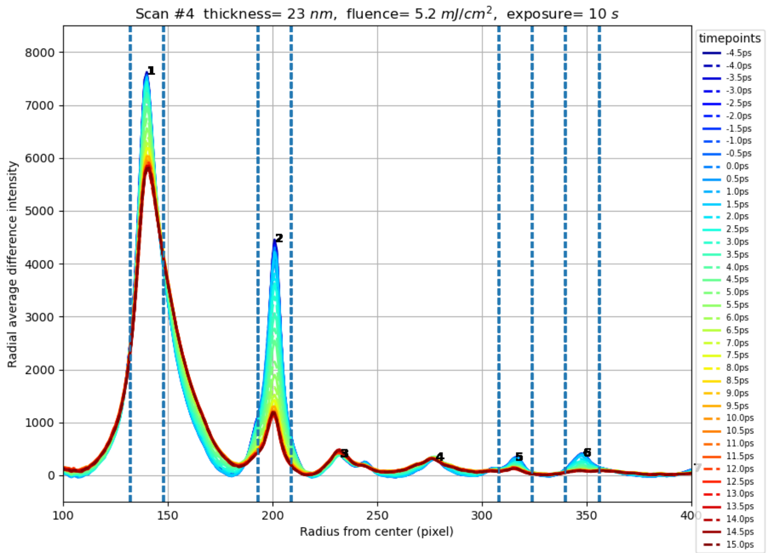
Diffraction image processing

In this project, we are specifically investigating the ultrafast melting of bismuth metal. This project involves processing and analyzing a series of electron diffraction images of polycrystalline bismuth. The goal is to find the speed and response time of its melting process.

First, we must process the 2D diffraction images into usable data. We do that by performing a radial average of the diffraction pattern, which is stored in TIFF image files. This is done with two main steps:

1. Find the center of the diffraction pattern
2. Integrate the intensity of the pixels in the radial axis



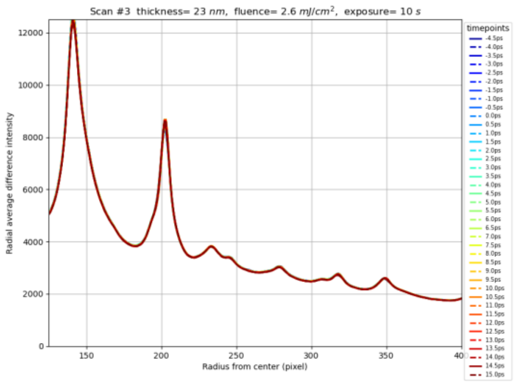


Processing background

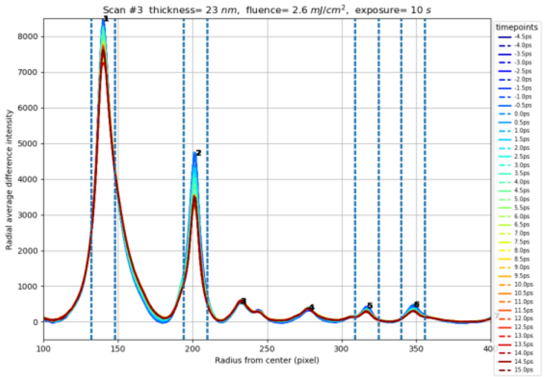
We must also deal with background noises. In capturing the diffraction pattern, there is unwanted background light (noise) that distorts the signal data.

The first step is to subtract the background image captured by the camera from the raw diffraction image. Then from the radial average, we computationally flattened the background using signal processing tools from Scipy.

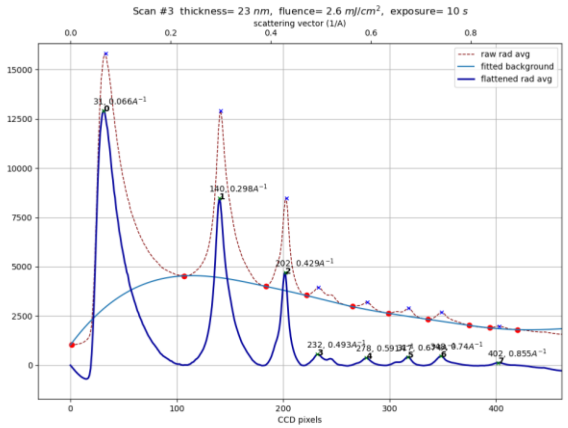
Original



Background subtracted



Flattening process



https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

Organization of data

- Sample
  - Fluence
    - Diffraction Peak
      - Timepoints

Sample is the specific sample we studied. The samples are different by the thickness of the Bi layer.

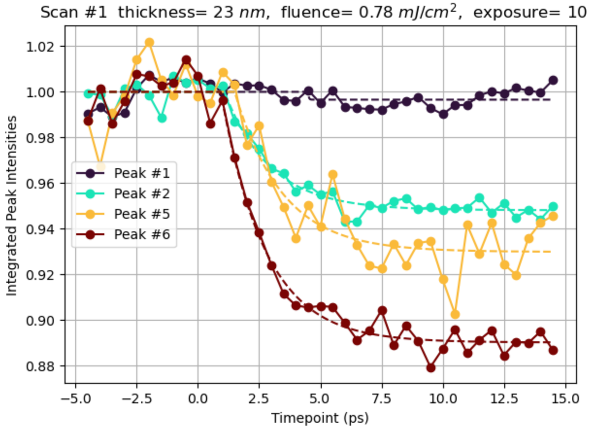
Fluence is the laser power used to trigger the sample in the experiment.

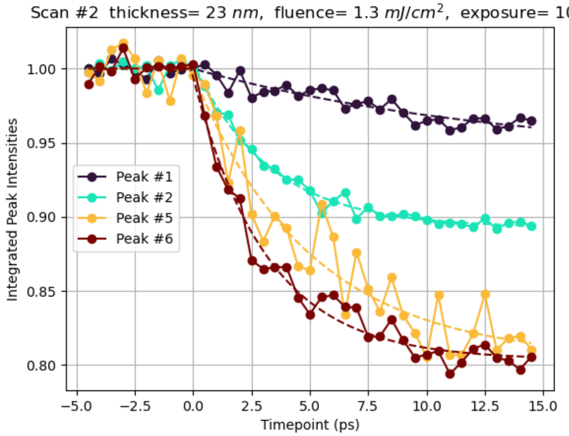
Diffraction peak refers to the specific ring of the diffraction pattern

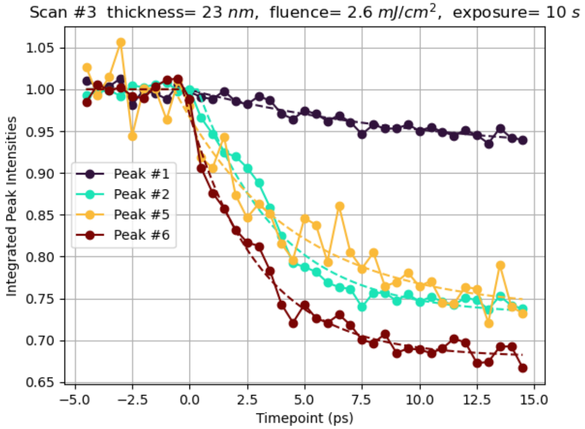
Timepoints are the time delay between pump and probe, usually plotted on the x-axis.

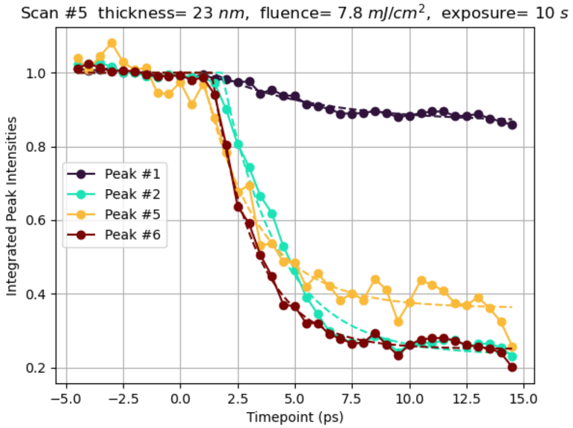
Data Visualization

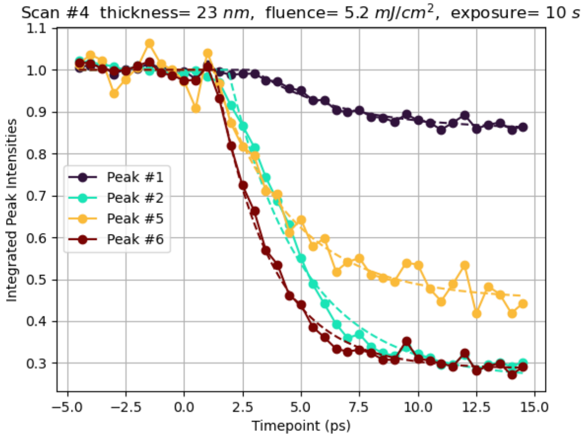
We look at the decay of peak intensities for each fluence

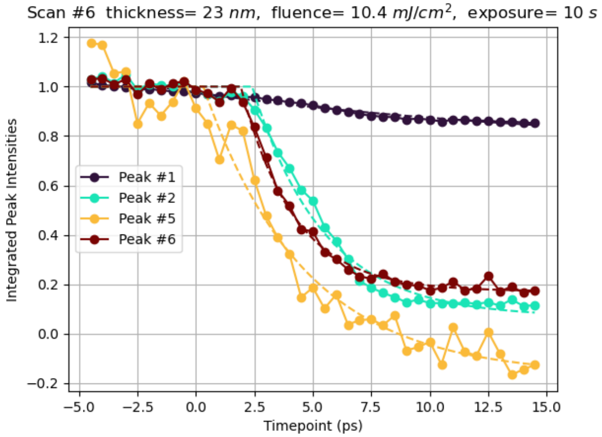






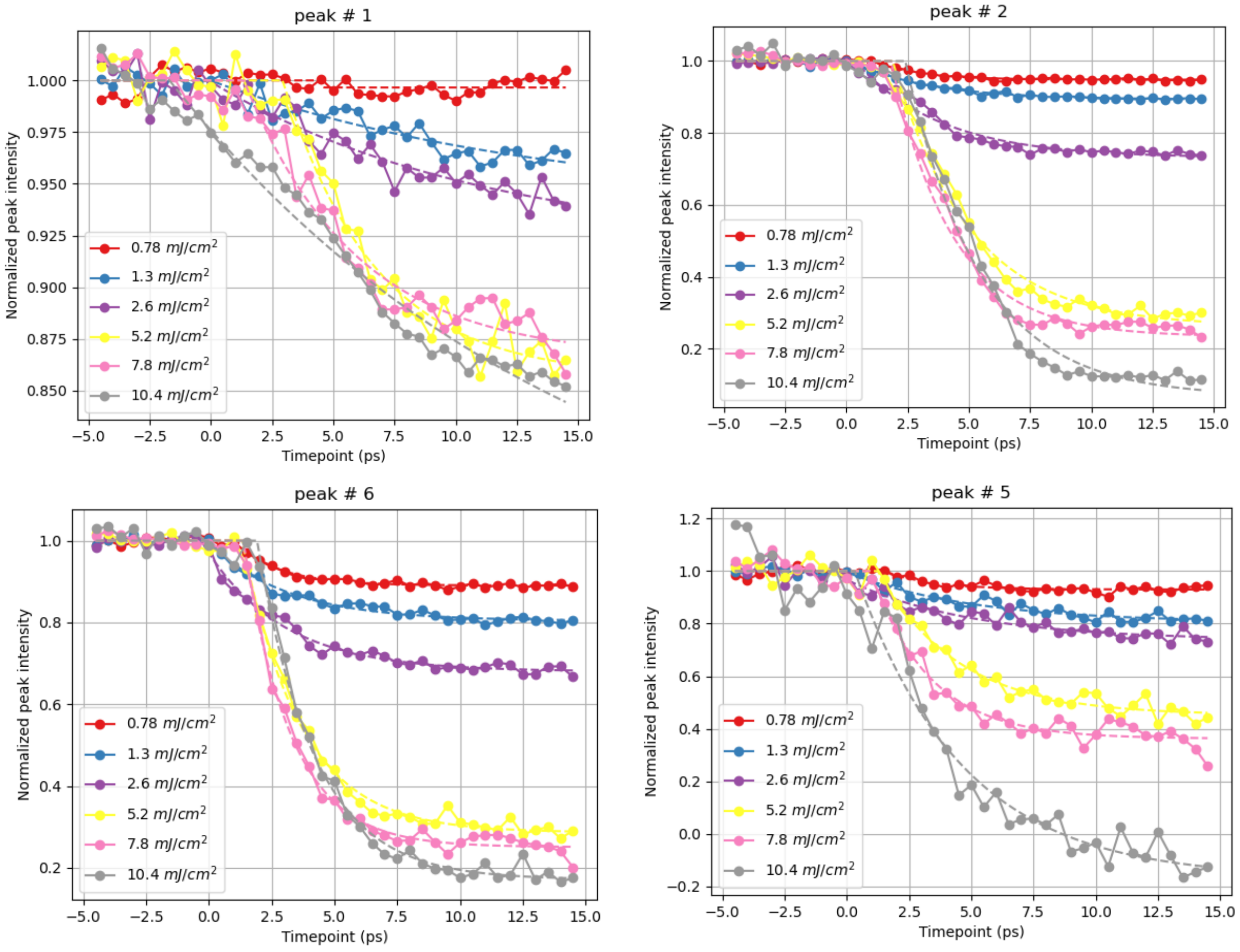






Same data, plotted differently.

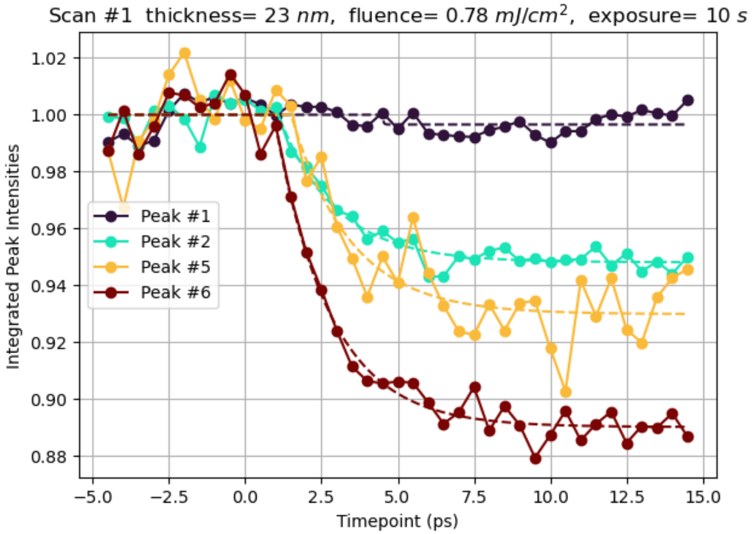
Looking at the effect of fluence on each peak



Model fitting

The decay of the peaks is modeled by a known equation. From the model, we extract the relevant parameters that give us insight into dynamic changes in the bismuth structure.

For example, the parameter $\boldsymbol{\tau}$ is the time constant of the exponential, i.e. melting time



Connected lines represent raw data

Dashed line is the model fit

Exact equation

$$I(t) = G_\sigma(t) * \left[c_i\left(1 - e^{-\frac{t}{\tau}}\right)\right] = \frac{1}{2}c_i\left[\left(1 + \mathrm{erf}\left(\frac{t}{\sqrt{2}\sigma}\right)\right) - e^{\frac{-t}{\tau}}e^{\frac{1}{2}\left(\frac{\sigma}{\tau}\right)^2}\left(1 + \mathrm{erf}\left(\frac{t - \frac{\sigma^2}{t}}{\sqrt{2}\sigma}\right)\right)\right]$$

```python
import numpy as np
from scipy.special import erf
from scipy.special import erfc
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
from matplotlib import cm

def Exp_Fit(timedata, peakdata, a_fit, tau_fit, t0_fit, c_fit, sigma = 1.0, plotlabel=''):
    """..."""

    times = np.array(timedata)   # convert whatever into array
    peaks = np.array(peakdata)

    def ExponentialIntensityDecay(x, A1, tau1, t0, c):
        #sigma = 0.3  # instrument response function ?

        erf1 = 1 + erf((x - t0) / (np.sqrt(2) * sigma))
        exp1 = np.exp((sigma ** 2 - 2 * (x - t0) * tau1) / (2 * tau1 ** 2))
        erfc1 = erfc((-sigma + ((x - t0) * tau1 / sigma)) / (np.sqrt(2) * tau1))

        return (1 / 2) * ((A1 * erf1) + (A1 * exp1) * (-2 + erfc1)) + c

    popt, pcov = curve_fit(ExponentialIntensityDecay, times, peaks, p0 =(a_fit, tau_fit, t0_fit, c_fit))

    x_out = np.arange(min(times), max(times), 0.01)
    y_out = ExponentialIntensityDecay(x_out, popt[0],popt[1],popt[2],popt[3])

    return x_out, y_out, popt, pcov
```

Parameters for model fit

Tools used

- Standard Python data manipulation libraries
  - Numpy - for certain math operations and dealing with arrays
  - Pandas - DataFrames are essential for this analysis
  - Scipy - curve fit and certain image processing tools
  - Sqlalchemy - exporting and query the data from a SQL server into Pandas dataframes
- PostgresSQL server for storing and organizing data
- Tableau for quick visualization of data

Thanks for listening ☺

Here is the GitHub of all the code used in this project

https://github.com/johnnykfeng/Bismuth-Data-Analysis-github/tree/working-branch

P.S. it's not very organized or well documented