



Development of modern and easy-to-use XRD analysis tools for NSUF users

Presenter: Mehmet Topsakal

Date 04/15/2025











































Outline of this talk

- Introduction to NSUF facility at NSLS-II
- What was the motivation of this development?
- How does the *easyXRD* work?

Who are we?

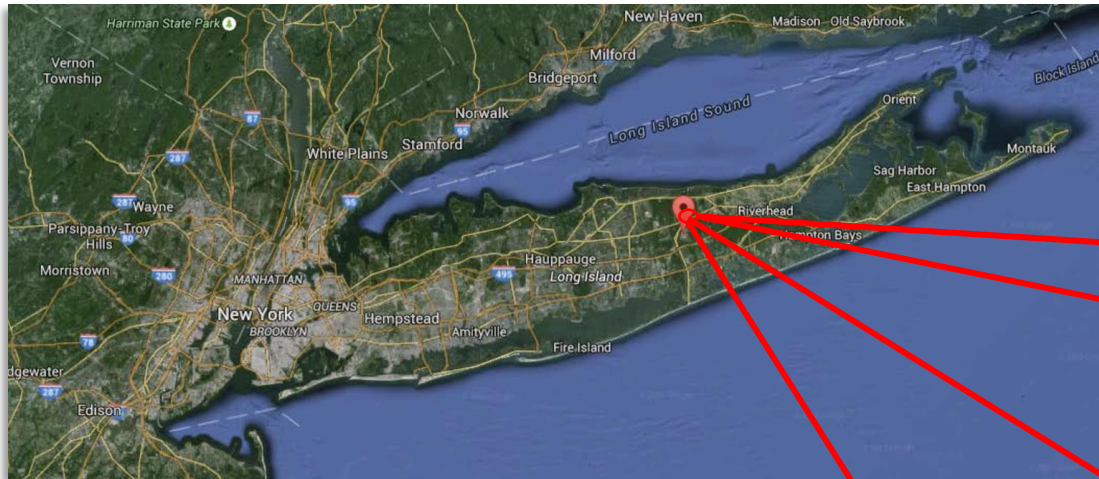
NSUF Capabilities Offer Research Opportunities

Neutron Irradiations	Ion Irradiations	Gamma Irradiations	Hot Cells & Shielded Cells	Low Activity Laboratories	Beamlines	High Performance Computing
 Idaho National Laboratory	 WISCONSIN UNIVERSITY OF WISCONSIN-MADISON	 Idaho National Laboratory	 Idaho National Laboratory	 Idaho National Laboratory	 BROOKHAVEN NATIONAL LABORATORY	 Idaho National Laboratory
 OAK RIDGE National Laboratory	 M UNIVERSITY OF MICHIGAN	 OAK RIDGE National Laboratory	 OAK RIDGE National Laboratory	 CAES Center for Advanced Energy Studies	 Argonne NATIONAL LABORATORY	
 MIT Massachusetts Institute of Technology	 Argonne NATIONAL LABORATORY	 Sandia National Laboratories	 PNL	 OAK RIDGE National Laboratory	 NC STATE UNIVERSITY	
 NC STATE UNIVERSITY	 Sandia National Laboratories		 Los Alamos NATIONAL LABORATORY EST. 1943	 Cal	 Los Alamos NATIONAL LABORATORY EST. 1943	
 OHIO STATE	 ATM		 Westinghouse	 PNL		
 Sandia National Laboratories	 TEXAS A&M UNIVERSITY.		 M UNIVERSITY OF MICHIGAN	 Los Alamos NATIONAL LABORATORY EST. 1943	 NC STATE UNIVERSITY	 M UNIVERSITY OF MICHIGAN
 SCK • CEN STUDECENTRUM VOOR KERNENERGIE CENTRE D'ETUDE DE L'ENERGIE NUCLEAIRE	 Lawrence Livermore National Laboratory		 SCK • CEN STUDECENTRUM VOOR KERNENERGIE CENTRE D'ETUDE DE L'ENERGIE NUCLEAIRE	 PURDUE UNIVERSITY		
				 WISCONSIN UNIVERSITY OF WISCONSIN-MADISON		

Visit nsuf.inl.gov for details of individual facilities

Where are we?

Brookhaven National Laboratory (BNL) is a United States Department of Energy national laboratory located in Upton, New York, on Long Island, and was formally established in 1947.











































- Located on Long Island
- Land: 5322 acres
- 319 buildings

BNL hosts National Synchrotron Light Source-II (NSLS-II)

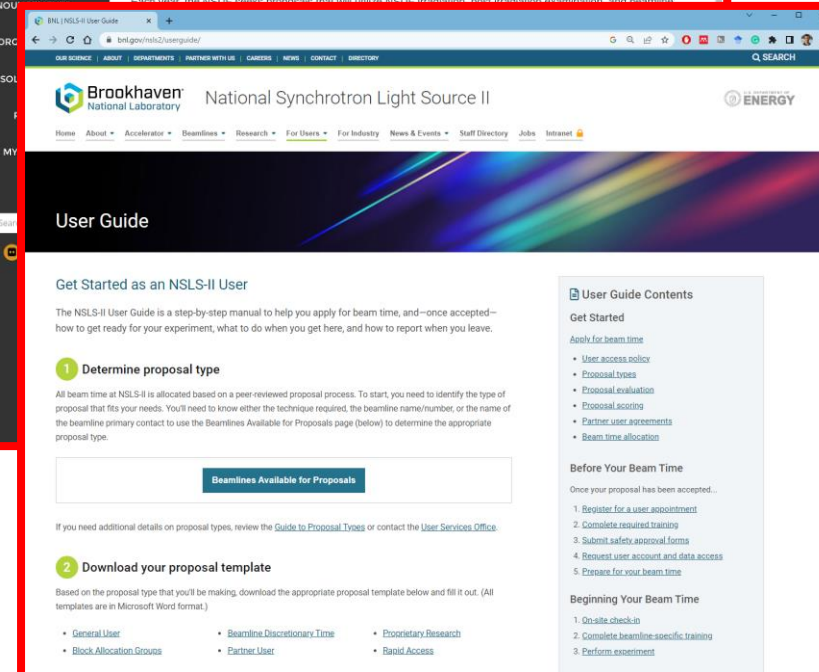
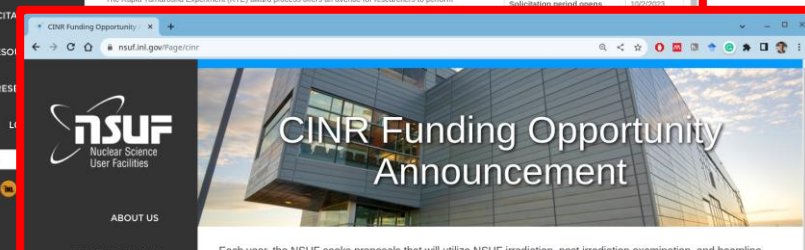
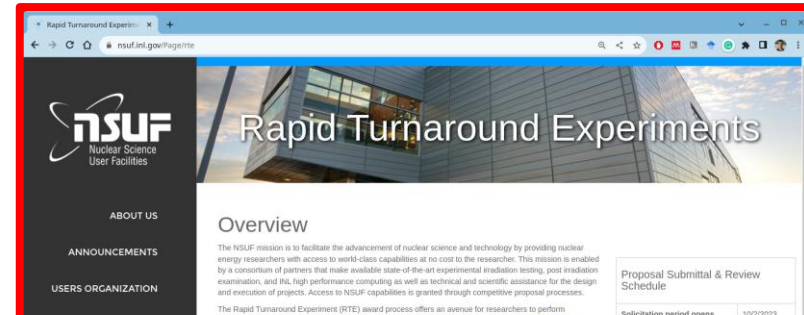
Who are we?

Since **2017**, we are supporting nuclear science users under the umbrella of Nuclear Science User Facilities (NSUF)

NSUF Capabilities Offer Research Opportunities

Neutron Irradiations	Ion Irradiations	Gamma Irradiations	Hot Cells & Shielded Cells	Low Activity Laboratories	Beamlines	High Performance Computing
 Idaho National Laboratory	 WISCONSIN UNIVERSITY OF ECONOMIC RESEARCH	 Idaho National Laboratory	 Idaho National Laboratory	 Idaho National Laboratory	 BROOKHAVEN NATIONAL LABORATORY	 Idaho National Laboratory
 National Laboratory	 UNIVERSITY OF MICHIGAN	 National Laboratory	 National Laboratory	 Center for Advanced Energy Studies	 ARGONNE NATIONAL LABORATORY	
 Massachusetts Institute of Technology	 Sandia National Laboratories	 Sandia National Laboratories	 PNNL	 National Laboratory	 NC STATE UNIVERSITY	
 NC STATE UNIVERSITY	 ARGONNE NATIONAL LABORATORY	 Los Alamos National Laboratory	 Cal	 PNNL	 Los Alamos National Laboratory	
 OHIO STATE UNIVERSITY	 Sandia National Laboratories	 Westinghouse	 UNIVERSITY OF MICHIGAN	 Los Alamos National Laboratory		
 Sandia National Laboratories	 TEXAS A&M UNIVERSITY	 UNIVERSITY OF MICHIGAN	 NC STATE UNIVERSITY	 UNIVERSITY OF MICHIGAN		
 LAWRENCE LIVERMORE NATIONAL LABORATORY	 STUDECENTRUM VOOR KERNEENERGIE CENTRE D'ETUDE DE L'ENERGIE NUCLEAIRE	 STUDECENTRUM VOOR KERNEENERGIE CENTRE D'ETUDE DE L'ENERGIE NUCLEAIRE	 PURDUE UNIVERSITY	 UNIVERSITY OF WISCONSIN		

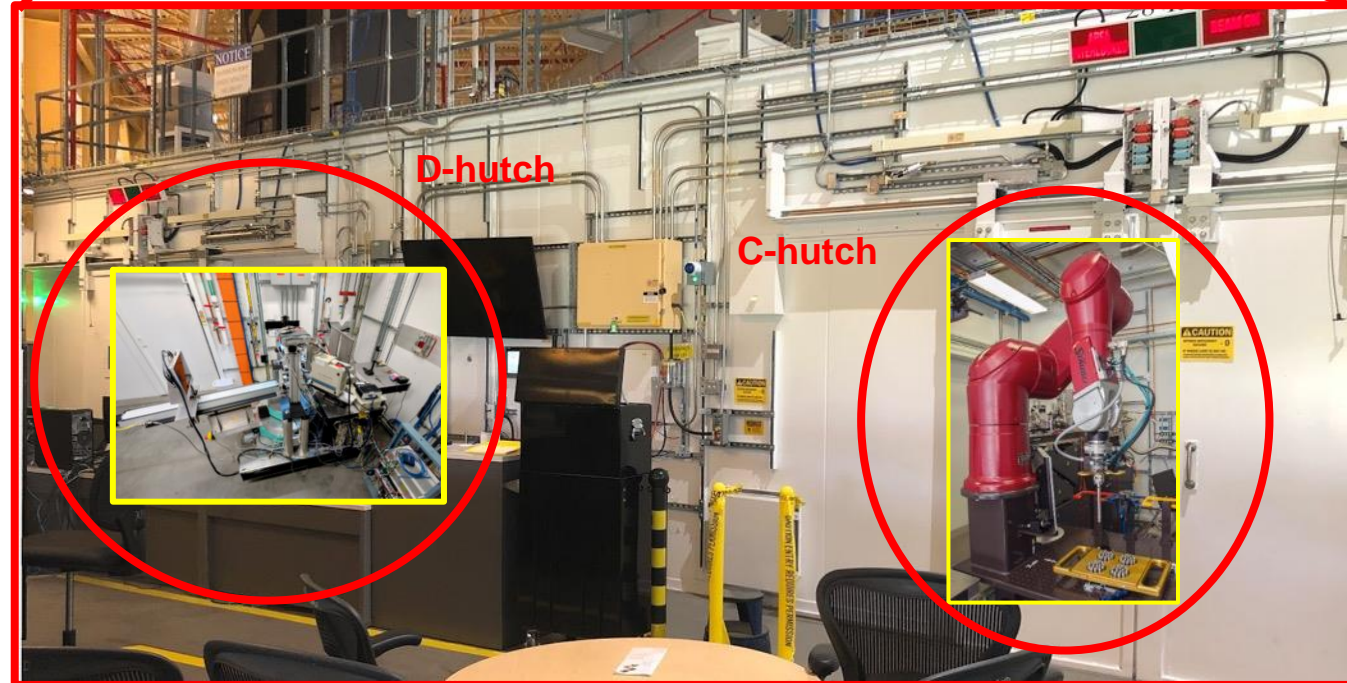
Visit nsuf.inl.gov for details of individual facilities





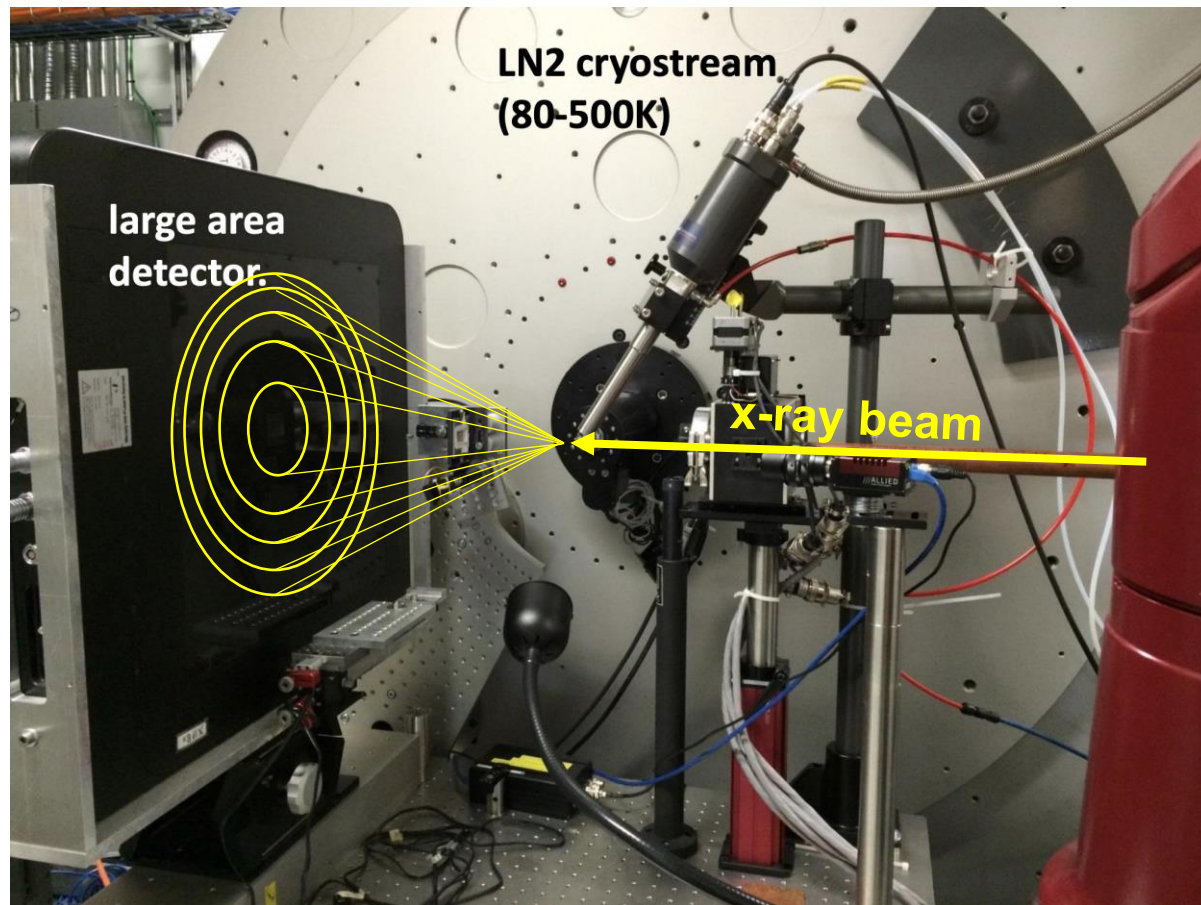
XPD beamline (28-ID) at NSLS-II

%6 of beamtime of XPD beamline
is allocated to NSUF users.
(3-days of beamtime, three times in a year)

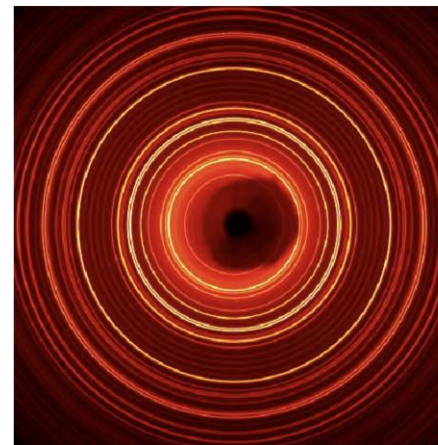


One-slide on X-ray diffraction method

A typical X-ray diffraction setup with area detector

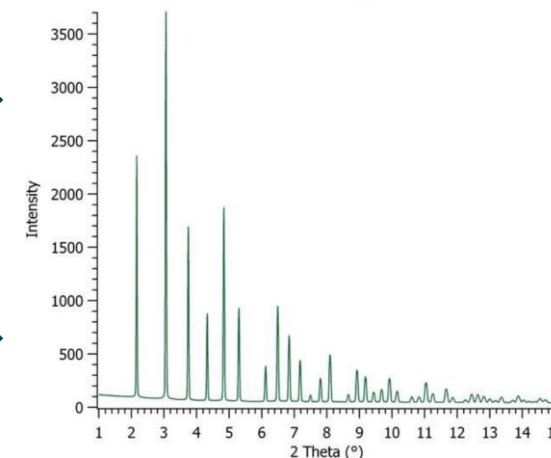


2D image



Azimuthal
integration

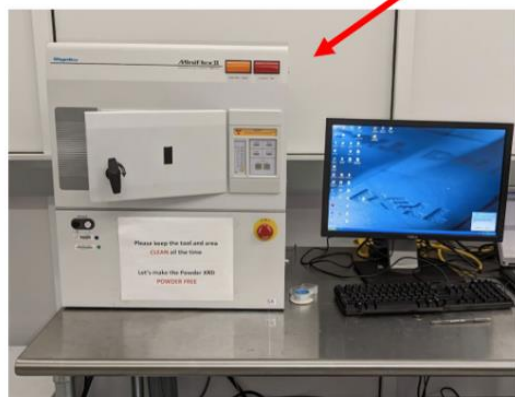
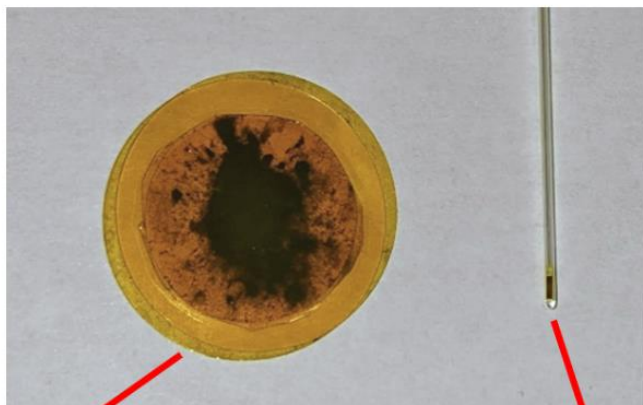
1D pattern



- Phase information
- Lattice parameters
- Phase mass fraction
- Micro strain
- Crystalline size
- Stacking faults

Why do we need synchrotron resources ?

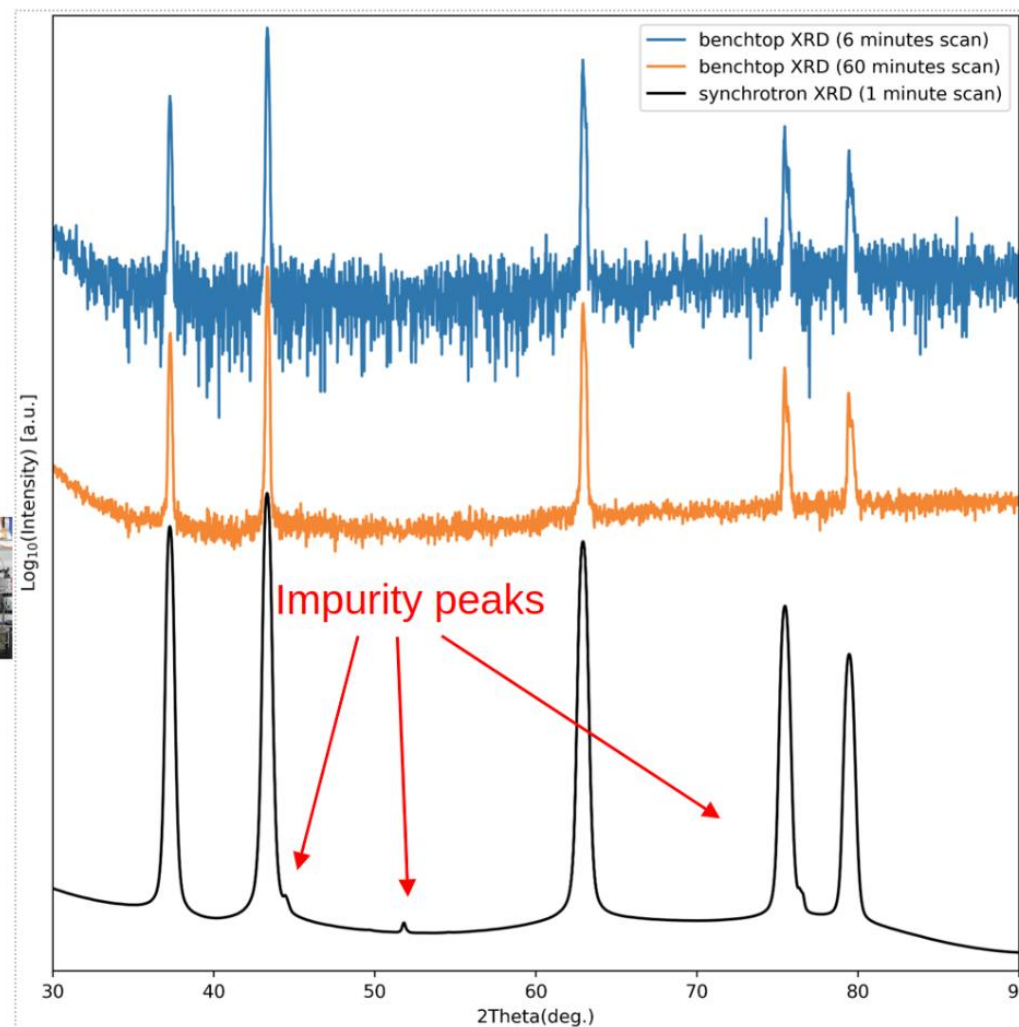
NiO powder test sample

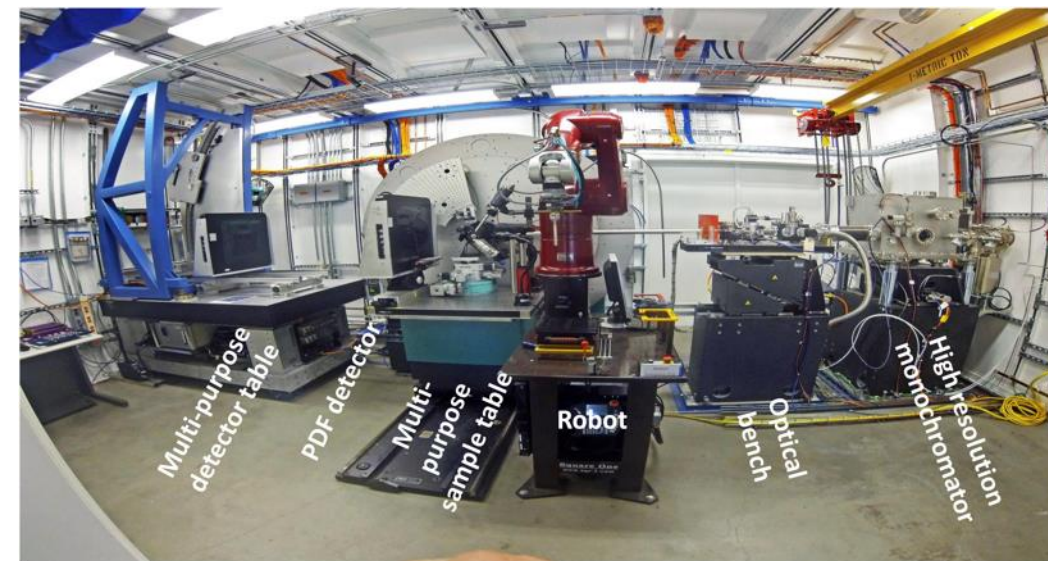


Benchtop XRD (Rigaku)

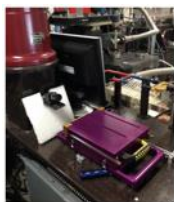


NSLS-II (XPD beamline)

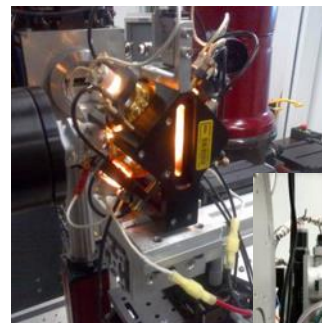




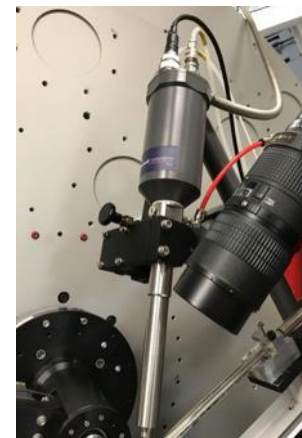
Sample Holder



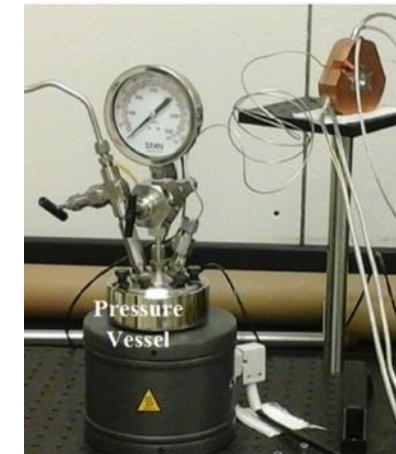
Robot for high-throughput sample changing



Heating up to 2000°C



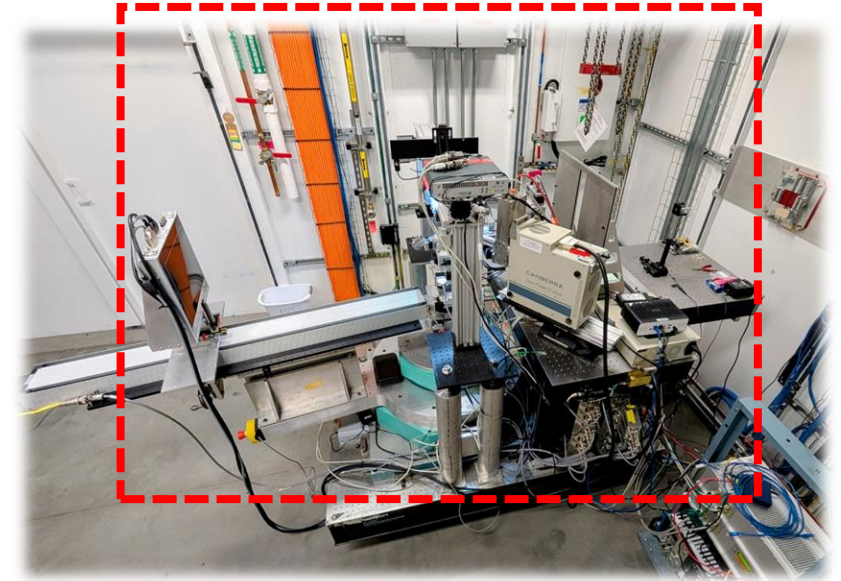
Cryostream (80K-500K)



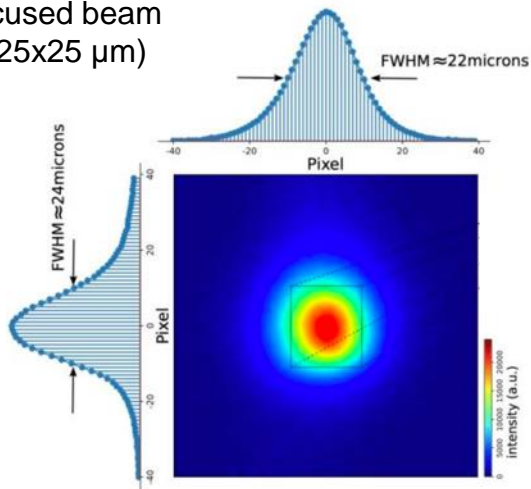
Corrosion cell



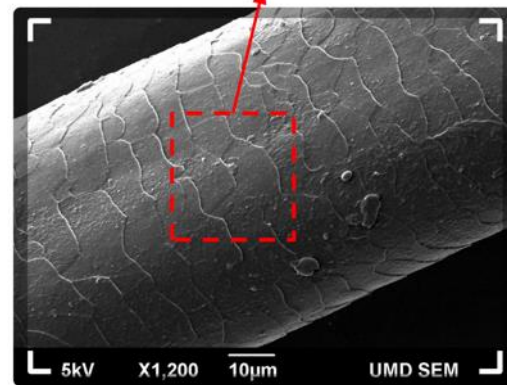
proudly supported by



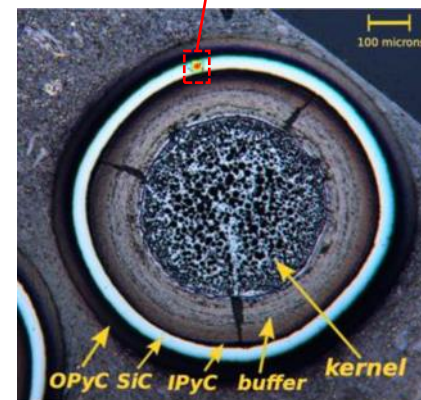
Focused beam
(~25x25 μm)



25 microns x 25 microns
spot on human hair



25 microns x 25 microns
spot on TRISO fuel



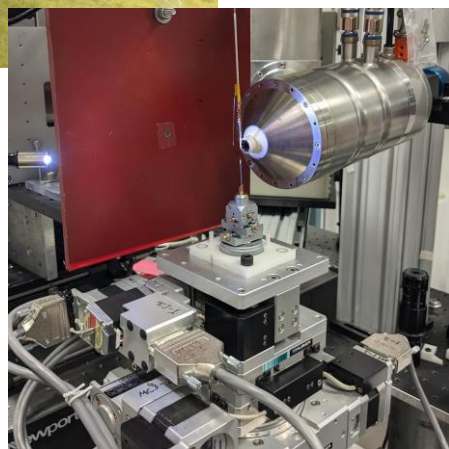
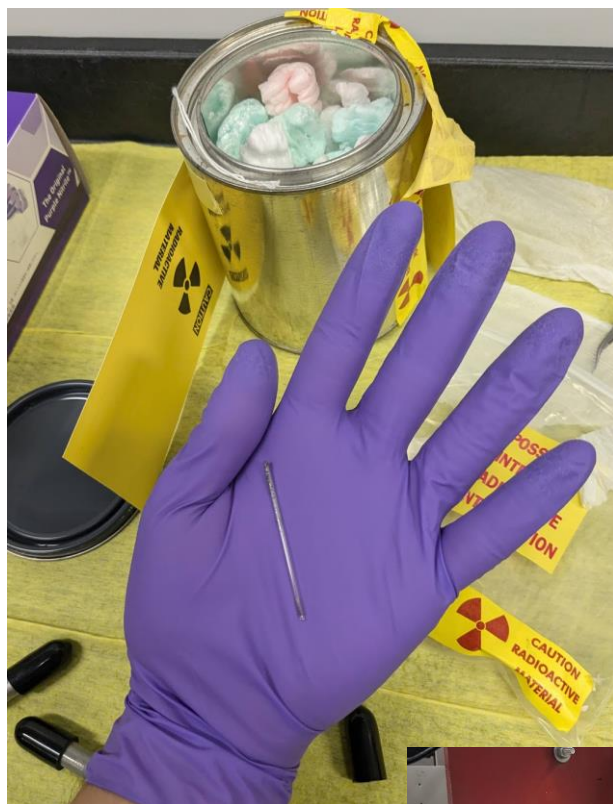
Techniques enabled with small beam:

- 1D & 2D mapping (phase, lattice, strain...)
- 3D X-ray diffraction computed tomography (XRD-CT)

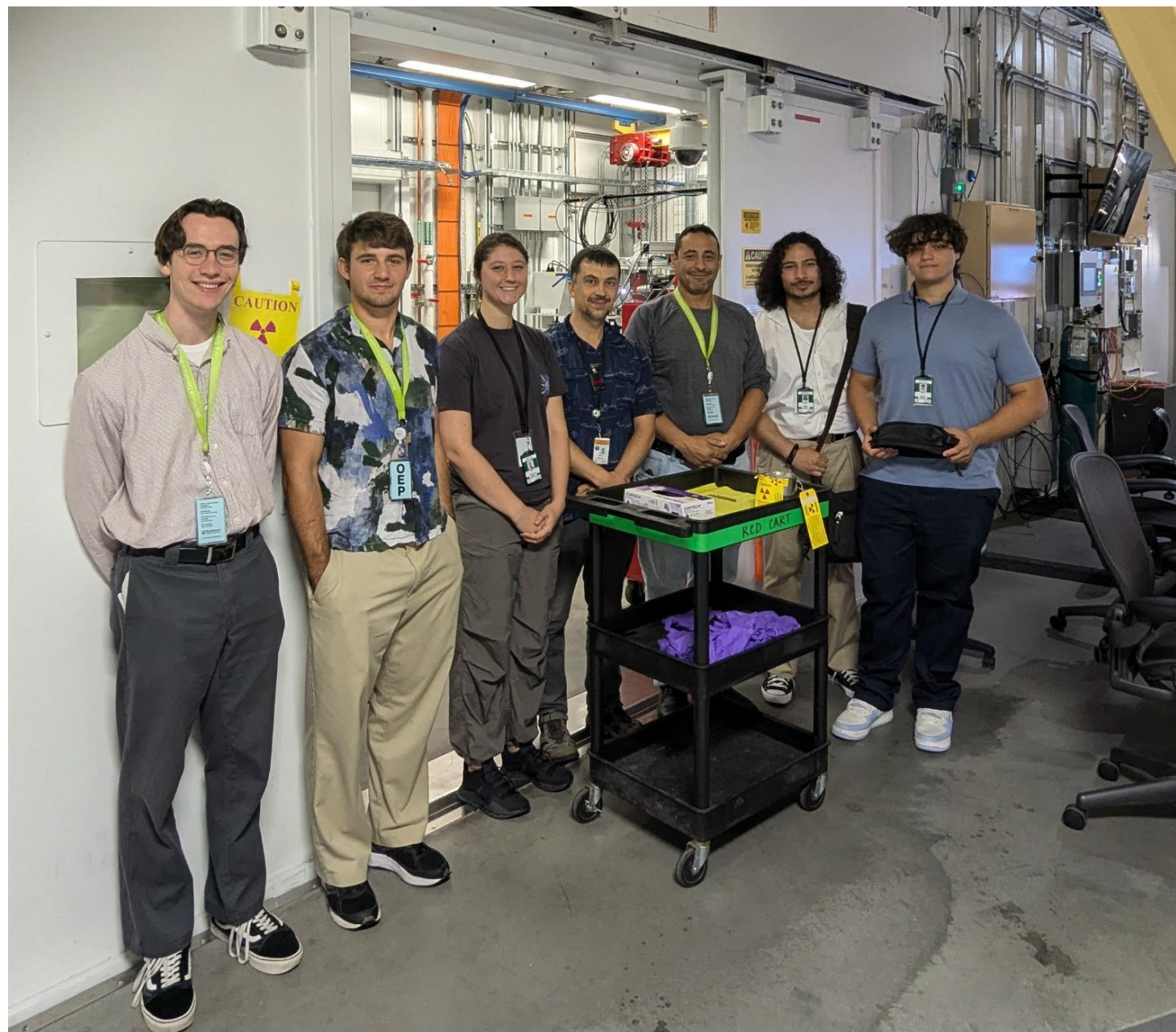
X-ray fluorescence spectroscopy with high-energy beam.

Multi-modal 1D, 2D, and 3D non-destructive characterization of nuclear materials.

Some pictures from a recent RTE (24-4876) experiment



UB₄ samples heated up to 900°C

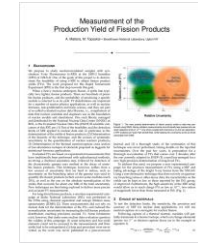
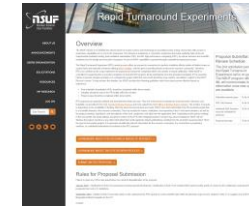


Outline of this talk

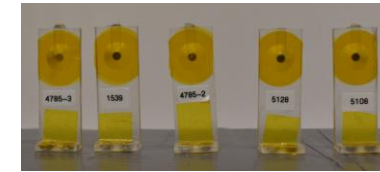
- Introduction to NSUF facility at NSLS-II
- What was the motivation of this development?
- How does the *easyXRD* work?

Efforts put on an NSUF-funded project at NSLS-II

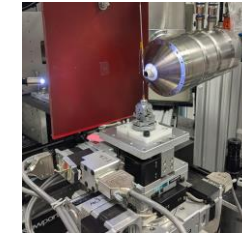
Proposal writing
(maybe a week)



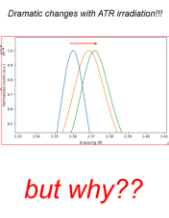
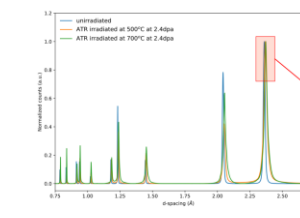
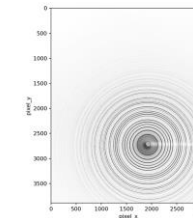
Sample preparation and shipment
(months, sometimes a year!)



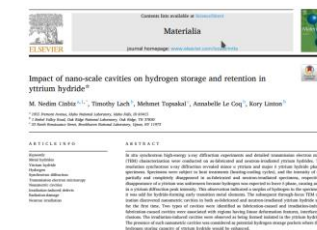
Beamtime experiment at NSLS-II
(2-3 days max)



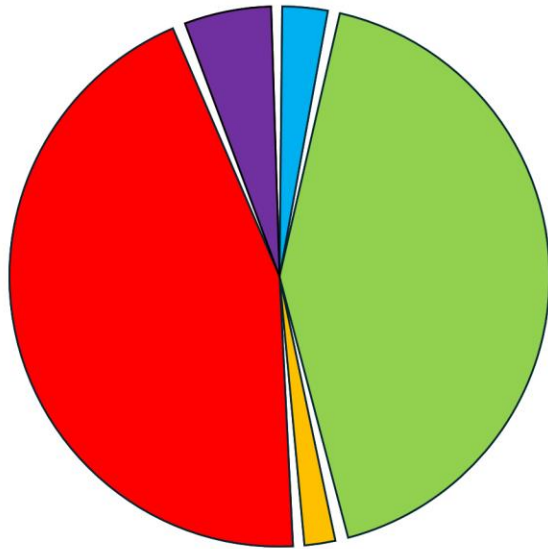
Data analysis
(no time info, sometimes forever!)



Manuscript/final report writing
(final step)



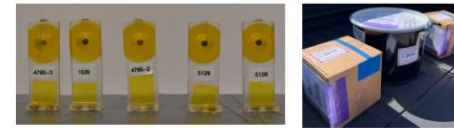
Efforts put on an NSUF-funded project at NSLS-II



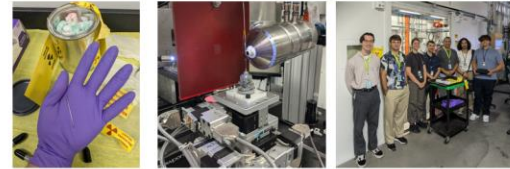
Proposal writing
(maybe a week)



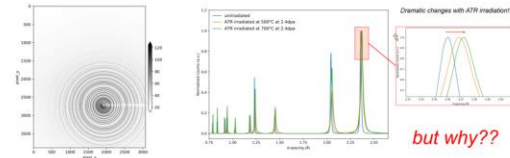
Sample preparation and shipment
(months, sometimes a year!)



Beamtime experiment at NSLS-II
(2-3 days max)



Data analysis
(no time info, sometimes forever!)



Manuscript/final report writing
(final step)



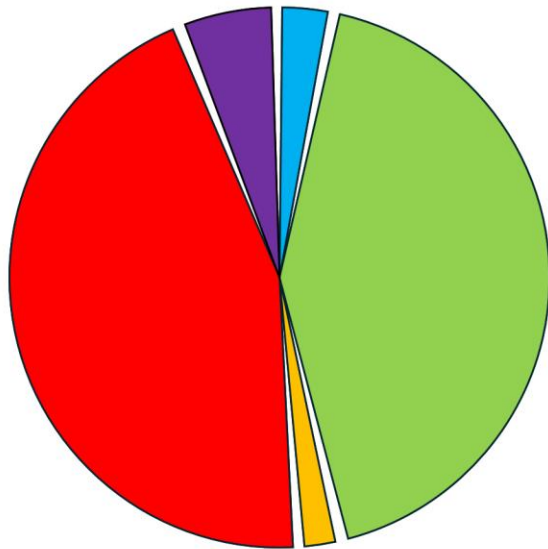
In case of a failure?

- proposal is not funded
- Experiment is delayed
- Experiment can be repeated or take longer



No output. All efforts above go to trash!!!

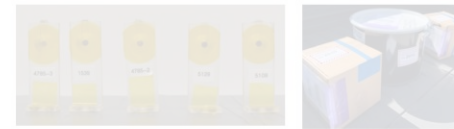
Efforts put on an NSUF-funded project at NSLS-II



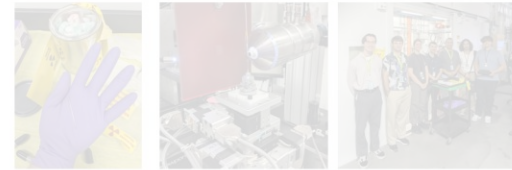
Proposal writing
(maybe a week)



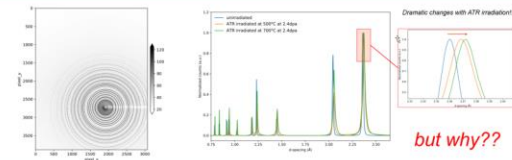
Sample preparation and shipment
(months, sometimes a year!)



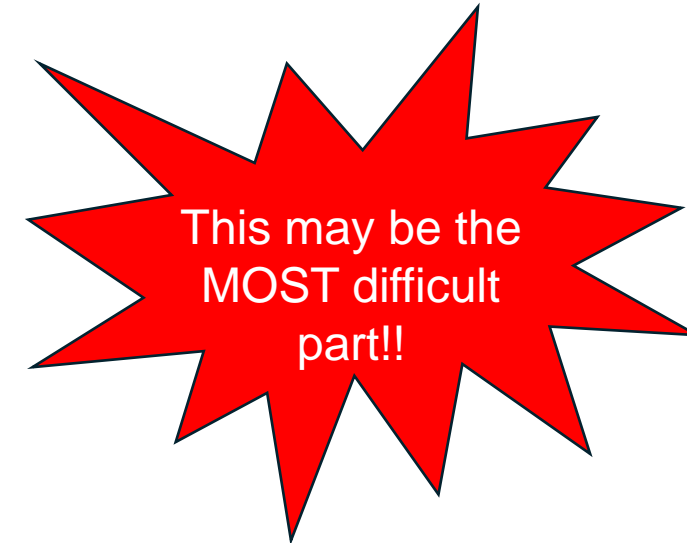
Beamtime experiment at NSLS-II
(2-3 days max)



Data analysis
(no time info, sometimes forever!)



Manuscript/final report writing
(final step)



GSAS-II and TOPAS software are commonly used in the XRD community. They are useful. But have some problems.....

GSAS-II web documentation 1.0 documentation » GSAS-II Home Page

Table of Contents

- GSAS-II Home Page
- Installation overview
- Available Documentation
- Mailing List
- Reporting Bugs
- Information for code developers
- Installation details

Related Topics

- Documentation index
- Next: 1. GSAS-II Installation Options

Quick search

Go

GSAS-II Home Page



Contents

1. GSAS-II Installation Options
2. Available Documentation for GSAS-II
3. Mailing List
4. Reporting GSAS-II Bugs
5. Information for GSAS-II Developers
6. Compiling GSAS-II
7. Customized Python Installation
8. Web Proxies

Welcome to the home page for GSAS-II, a unique and comprehensive open source Python project for determination of crystal structures and diffraction-based materials characterization for crystalline solids on all scales, from perovskites through proteins, using both powder and single-crystal diffraction and with both x-ray and neutron probes. Refinements can combine measurements from laboratory and synchrotron x-rays, as well as constant wavelength or time-of-flight neutron sources. It provides structure solution and refinement, as well as extensive visualization capabilities.

GSAS-II is made available for free use (see license) with open access to the source code.

Tip: Please help us by citing:

Toby, B. H., & Von Dreele, R. B. (2013). "GSAS-II: the genesis of a modern open-source all purpose crystallography software package". *Journal of Applied Crystallography*, 46(2), 544-549. doi:10.1107/S0021889813003531

Note that some sections of the program utilize externally provided codes or reference later work, with citations provided as they are used. Please cite them as well.

Also, please do sign up for the GSAS-II mailing list see below. We add new features to GSAS-II quite frequently, so we may break things from time to time (see bug reporting). Be sure to use the `Help-->update` capability frequently to stay abreast of new features and fixes as they are added and please make sure to use the latest version before reporting a bug to us, but please do report bugs.

Installation overview

There are several different ways to install GSAS-II, as are outlined below. Most people will use the GSAS2FULL installer

Installation information:

1. GSAS-II Installation Options
 - 1.1. GSAS2FULL installer
 - 1.1.1. Windows
 - 1.1.2. MacOS and Linux
 - 1.1.3. GSAS2FULL Additional Details
 - 1.1.3.1. Windows GSAS2FULL Installation Details
 - 1.1.3.2. MacOS GSAS2FULL Installation Details
 - 1.1.3.3. Linux GSAS2FULL Installation Details
 - 1.2. GSAS2PKG Conda Package
 - 1.3. Installing GSAS-II after Python Installation
 - 1.4. Installation for GSAS-II Software Developers
 - 1.5. Older methods

TOPAS-Academic V7

by Coelho Software

Brisbane, Australia

August, 2020

Australian Business Number

70 827 688 789

AlanCoelho@bigpond.com

Introduction

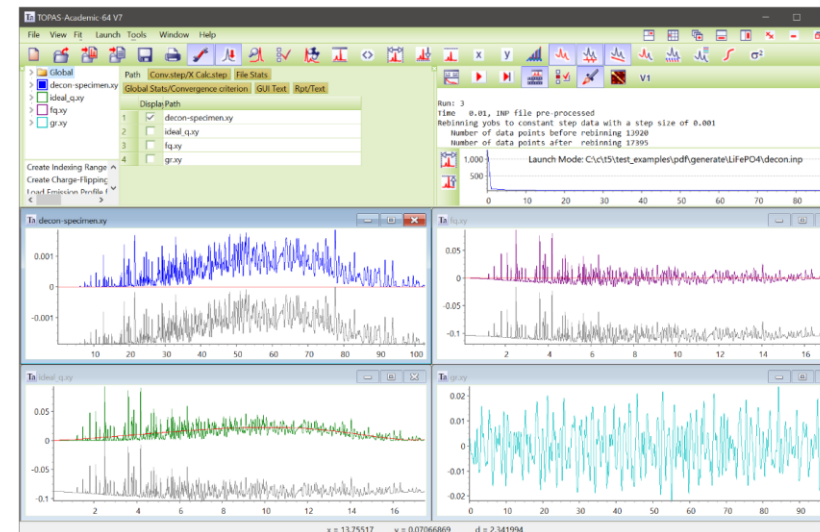
Version 7 of TOPAS-Academic [Technical Reference.pdf](#) is now available to:

- 1) Degree-granting institutions comprising universities, university run institutes, laboratories and schools.
- 2) Synchrotron and neutron research centers.

TOPAS-Academic comprises all the functionality of [BRUKER-AXS TOPAS](#) operating in Launch mode including all graphics and OpenGL structure editor and viewer. Launch mode is where the full functionality of TOPAS is realized.

*** Use of TOPAS-Academic requires computer literacy ***

What's new [New7.pdf](#)



*** Version 7 runs on 64 bit Windows ***

Pricing - Single computer/user licenses on a departmental basis

Each license will run one instance of the program on a specific computer. More than one person can of course use that computer but not at the same time; TA.EXE and TC.EXE can be both run at the same time. 10% VAT is charged for Australian customers only.

Version 7

New Users

- 1800 Euros for first license
- 600 Euros for each additional license

Version 6 upgrading to Version 7 and starting from the oldest

For first licence

- 110 Euros if purchased within the past 3 months
- 340 Euros if purchased within the past 6 months
- 450 Euros if purchased within the past 12 months
- 790 Euros if purchased within the past 24 months
- 1120 Euros if purchased previous to the past 24 months

For additional licences

- 60 Euros if purchased within the past 3 months
- 110 Euros if purchased within the past 6 months
- 170 Euros if purchased within the past 12 months
- 280 Euros if purchased within the past 24 months
- 450 Euros if purchased previous to the past 24 months

A fee may be charged for license transfers for computer upgrade reasons after an initial 2 year period

TOPAS is quite expensive and each user needs to purchase a license.

Outline of this talk

- Introduction to NSUF facility at NSLS-II
- What was the motivation of this development?
- How does the *easyXRD* work?

GitHub - MehmetTopsakal/easyXRD

Product Solutions Resources Open Source Enterprise Pricing

MehmetTopsakal / easyXRD Public

Code Issues Pull requests Actions Projects Security Insights

main 1 Branch 0 Tags

MehmetTopsakal minor fixes 7f7d599 · 17 hours ago 66 Commits

easyxrd	notebook paths were fixed	17 hours ago
.gitignore	Initial commit	last year
LICENSE	Initial commit	last year
README.md	minor fixes	17 hours ago

README GPL-3.0 license

We have developed a versatile X-ray diffraction (XRD) analysis tool that utilizes modern and open-source Python packages such as pyFAI, xarray, pymatgen, pybaselines,... for data processing/storage and interfaced to Jupyter notebooks powered with actively developed visualization packages such as ipywidgets, and matplotlib. It provides easy access to the Materials Project database which hosts thousands of crystal structures that can be used for phase identification - a critical part of XRD analysis - and utilizes GSAS-II suite for XRD refinements in a user-friendly and intuitive manner. Ultimate goal of this tool is to make X-ray diffraction analysis easy for users and help them to process, refine, store, and share their XRD data conveniently.

You can try the tutorial notebooks below on Google Colab:

- [01_basic.ipynb](#)
- [02_intermediate.ipynb](#)
- [03_advanced-part-1.ipynb](#)
- [03_advanced-part-2.ipynb](#)

About

No description, website, or topics provided.

Readme GPL-3.0 license Activity 0 stars 1 watching 0 forks Report repository

Releases

No releases published

Packages

No packages published

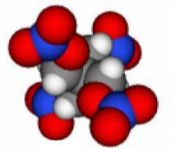
Languages

- Python 100.0%

Powered by



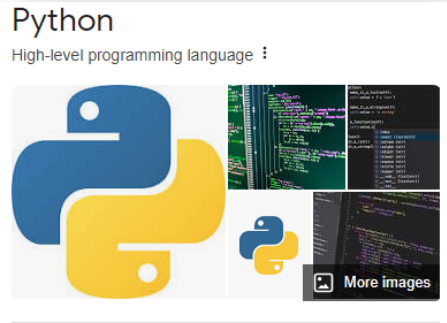
GSAS-II



provided to the user. This proposal aims to develop modern and easy-to-use analysis tools which are specifically optimized for high-throughput data generated at NSUF-supported end station at NSLS-II and aims to maximize NSUF-funded research outputs into high-impact publications.

The NSUF IS proposal states:

Proposed work: We propose to develop Python programming language based, state-of-the-art, open-source, well-documented and easy-to-use tools that will ease the analysis of the X-ray diffraction data by existing and future NSUF users. Major steps of XRD analysis of the data collected at NSUF-supported end station at NSLS-II consist of the following: i) Dark-field correction of sample-to-detector distance and detector orientation; ii) Calibration of sample-to-detector distance and detector orientation; iii) Integration of 2D data into 1D profile; iv)



Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. [Wikipedia](#)

Designed by: Guido van Rossum

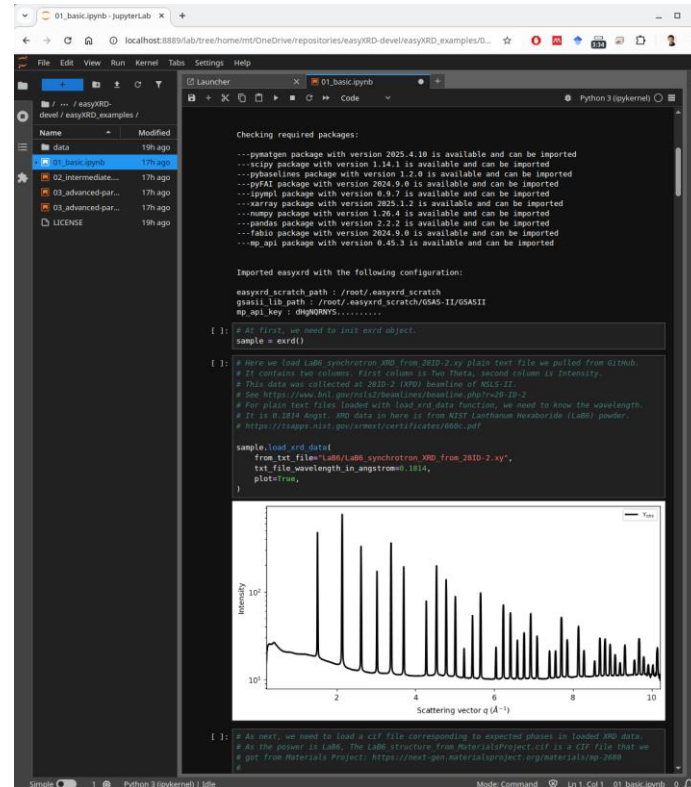
Developer: Python Software Foundation

First appeared: 20 February 1991; 33 years ago

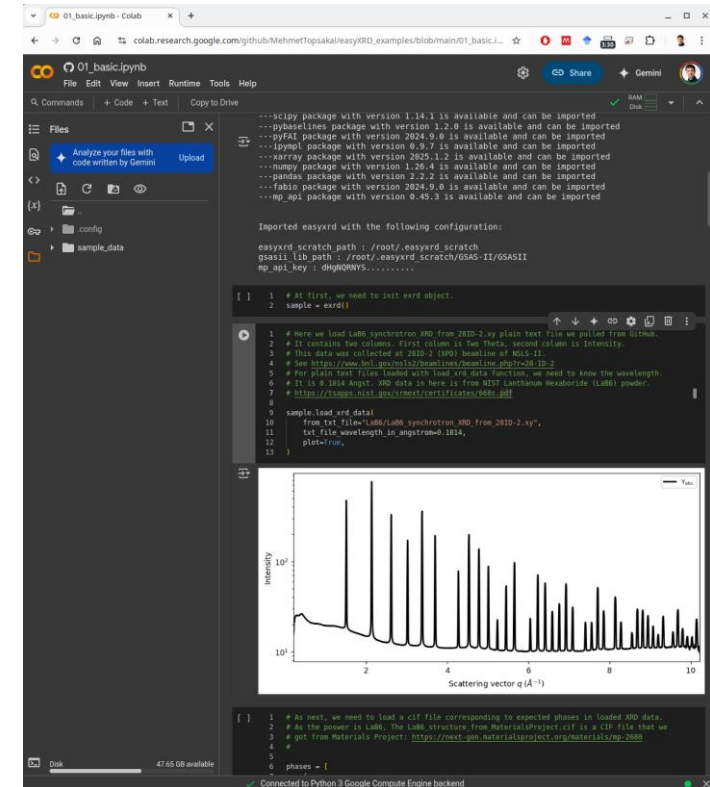
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), functional, structured, reflective

Stable release: 3.12.3 / 9 April 2024; 15 days ago

Typing discipline: duck, dynamic, strong; optional type annotations (since 3.5, but those hints are ignored, except with unofficial tools)



JupyterLab notebooks running locally



Or running on Google Colab service

Python is the state-of-the-art Scientific programming language

Some of NSUF users we worked in the past were struggling with installing Python and Jupyter Notebook packages onto their personal computers.

Using Google Colab services solves these problems.

Everything runs on the cloud and zero configuration is needed.

```
O2_intermediate.ipynb - Co x +
colab.research.google.com/github/MehmetTopsakal...
O2_intermediate.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text Copy to Drive Connect ^
[ ]
7 %matplotlib inline
8 import matplotlib.pyplot as plt
9
10 plt.rcParams["figure.constrained_layout.use"] = True # For better plots.
11
12 import os
13 import fabio
14
15 # change into data directory that is provided with easyXRD_examples repository
16 import os
17 os.chdir("easyXRD_examples/data")

Checking required packages:
---pymatgen package with version 2025.4.10 is available and can be imported
---scipy package with version 1.14.1 is available and can be imported
---pyFAI package with version 2024.9.0 is available and can be imported
---pybaselines package with version 1.2.0 is available and can be imported
---numpy package with version 1.26.4 is available and can be imported
---xarray package with version 2025.1.2 is available and can be imported
---ipyml package with version 0.9.7 is available and can be imported
---fabio package with version 2024.9.0 is available and can be imported
---pandas package with version 2.2.2 is available and can be imported
---mp_api package with version 0.45.3 is available and can be imported

Imported easyxrd with the following configuration:
easyxrd_scratch_path : /root/.easyxrd_scratch
gsasii_lib_path : /root/.easyxrd_scratch/GSAS-II/GSASII
mp_api_key : not found.....

[ ] 1 # We can list the contents of currant working directory with listdir command
2 os.listdir()

['Kapton', 'geometry.poni', 'mask.edf', 'LaB6_CeO2_mix', 'CeO2', 'LaB6']

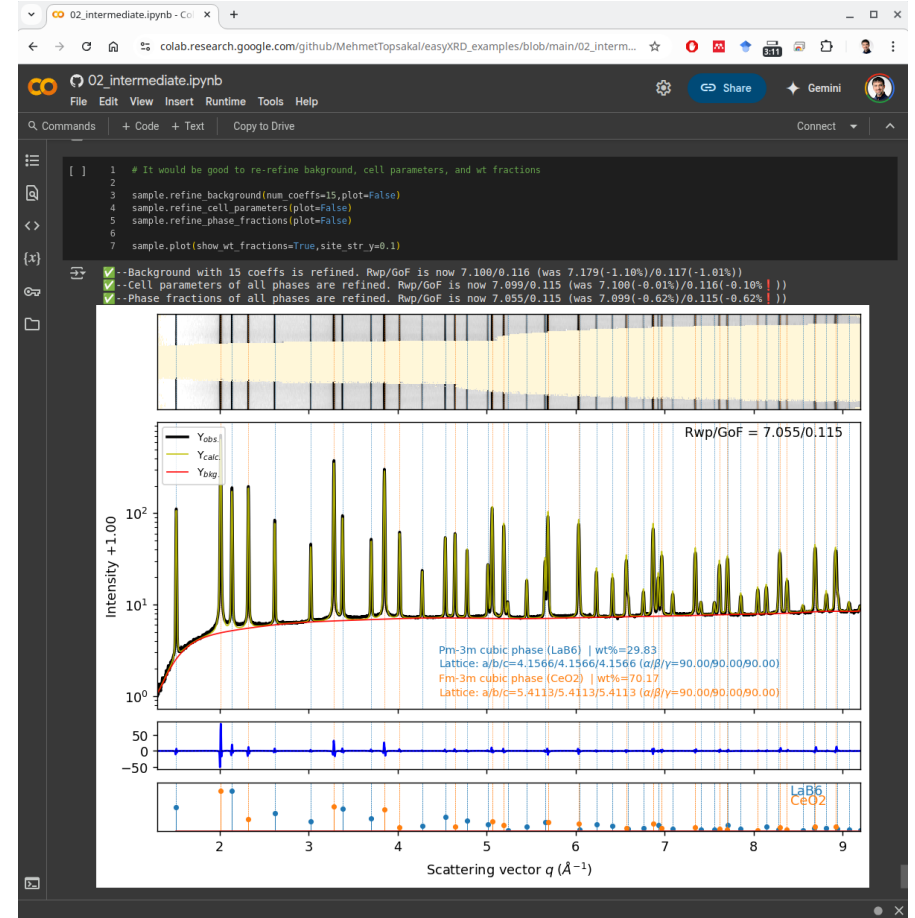
▼ LaB6

Here we start with LaB6/img.tiff which is actually 2D raw image of LaB6_synchrotron_XRD_from_281D-2.xy file we used in previous notebook.

[ ] 1 # We can display the LaB6/img.tiff as below.
2 plt.figure()
3 plt.imshow(fabio.open('LaB6/img.tiff').data, vmin=0, vmax=100)

<matplotlib.image.AxesImage at 0x7a17a88e5150>

In order to radially integrate this 2D Image, we need to use pyFAI.
Please refer to resources below about pyFAI usage.
• https://www.youtube.com/watch?v=hq0UyvepXk&ab_channel=ESRFsynchrotron
```



Built-in AI tools!

```
1 # The good thing with Jupyter notebooks is, we can do the entire refinement within a single cell as shown below.
2 # In order to simplify the notebook output, we set plot=False in some of the steps below. We also select scattering
3 # vector start from 1 as there are no peaks before and stop at 9.5 as the data gets noisy after that using the
4 # radial_range keyword. The symbol ⚡ highlights that there is improvement more than 10% in GoF
5 # while | highlights the change is less than 1% or positive.
6
7 sample = exrd()
8
9
10 sample.load_xrd_data(
11     from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",
12     txt_file_wavelength_in_angstrom=0.1814,
13     plot=False,
14     radial_range=[1, 9.5],
15 )
16
17 sample.load_phases(
18     [{"cif": "LaB6/LaB6_structure_from_MaterialsProject.cif", "label": "LaB6"}], plot=False
19 )
20
21 sample.setup_gsas2_refiner(plot=False)
22
23 sample.refine_background(plot=False)
24 sample.refine_cell_parameters(plot=False)
25 sample.refine_instrument_parameters(inst_pars_to_refine=["U"], plot=False)
26 sample.refine_instrument_parameters(inst_pars_to_refine=["V"], plot=False)
27 sample.refine_instrument_parameters(inst_pars_to_refine=["W"], plot=False)
28 sample.refine_instrument_parameters(inst_pars_to_refine=["Zero"], plot=False)
29
30 # We can re-refine background since we improved instrument parameters in previous steps
31 sample.refine_cell_parameters(plot=False)
32
33 # We can re-refine background with higher number of coefficients
34 sample.refine_background(num_coeffs=15, plot=False)
35
36 sample.plot()
37
38 # As in the final plot below, the refinement is fairly good. It should be noted that, this is
39 # LeBail refinement in which atom properties are not taken into consideration. Reitveld type
40 # refinement will be discussed in next notebook (02_intermediate.ipynb)
```

```
⚡--1st refinement with LeBail is completed. Rwp/GoF is 24.740/1.078
✔--Background with 10 coeffs is refined. Rwp/GoF is now 18.852/0.822 (was 24.740(-23.80%)/1.078(-23.75%⚡))
✔--Cell parameters are refined. Rwp/GoF is now 6.496/0.283 (was 18.852(-65.54%)/0.822(-65.56%⚡))
✔--Instrument parameter ['U'] is refined. Rwp/GoF is now 6.360/0.277 (was 6.496(-2.10%)/0.283(-2.10%))
✔--Instrument parameter ['V'] is refined. Rwp/GoF is now 6.307/0.275 (was 6.360(-0.83%)/0.277(-0.83%|))
✔--Instrument parameter ['W'] is refined. Rwp/GoF is now 6.237/0.272 (was 6.307(-1.11%)/0.275(-1.11%))
✔--Instrument parameter ['Zero'] is refined. Rwp/GoF is now 6.181/0.269 (was 6.237(-0.89%)/0.272(-0.89%|))
✔--Cell parameters are refined. Rwp/GoF is now 6.152/0.268 (was 6.181(-0.47%)/0.269(-0.47%|))
```

Explain code

Please explain this code.

Okay, let's break down this section of the code.

Refining XRD Data in a Single Cell

This code demonstrates how to perform a complete X-ray diffraction (XRD) data refinement process within a single Jupyter notebook cell. This approach makes the analysis more concise and easier to manage.

Steps:

- 1. Initialization:**

```
sample = exrd()
```

- A new `exrd` object named `sample` is created to store and manage the XRD data and refinement results.

- 2. Loading Data:**

```
sample.load_xrd_data(
    from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",
    txt_file_wavelength_in_angstrom=0.1814,
    plot=False,
    radial_range=[1, 9.5],
)
```

- This loads XRD data from a text file (`LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy`).
- `txt_file_wavelength_in_angstrom=0.1814` specifies the wavelength of the X-rays used in the experiment.
- `plot=False` prevents generating a plot at this stage.
- `radial_range=[1, 9.5]` limits the data analysis to the specified scattering vector (q) range, focusing region.

- 3. Loading Phases:**

```
sample.load_phases(
    [{"cif": "LaB6/LaB6_structure_from_MaterialsProject.cif", "label": "LaB6"}], plot=False
)
```

- Loads information about the expected material phases from a CIF file (`LaB6/LaB6_structure_from_MaterialsProject.cif`).
- `label="LaB6"` assigns a label to the phase for identification.
- `plot=False` again suppresses plot generation.

- 4. Setting up the Refiner:**

```
sample.setup_gsas2_refiner(plot=False)
```

- Initializes the GSAS-II refiner, a tool used for refining XRD data.
- `plot=False` prevents plotting.



5. Refinement Steps:

```
sample.refine_background(plot=False)
sample.refine_cell_parameters(plot=False)
sample.refine_instrument_parameters(inst_pars_to_refine=["U"], plot=False)
sample.refine_instrument_parameters(inst_pars_to_refine=["V"], plot=False)
sample.refine_instrument_parameters(inst_pars_to_refine=["W"], plot=False)
sample.refine_instrument_parameters(inst_pars_to_refine=["Zero"], plot=False)
```

- These lines perform the actual refinement:
 - `refine_background`: Refines the background of the XRD pattern.
 - `refine_cell_parameters`: Refines the unit cell parameters of the material.
 - `refine_instrument_parameters`: Refines specific instrument parameters (U, V, W, Zero).
 - `plot=False` is used to simplify output by suppressing intermediate plots.

6. Further Refinement:

```
# We can re-refine background since we improved instrument parameters in previous steps
sample.refine_cell_parameters(plot=False)

# We can re-refine background with higher number of coefficients
sample.refine_background(num_coeffs=15, plot=False)
```

- These lines refine the cell parameters and background again for better fitting.
- `num_coeffs=15` in `refine_background` increases the number of coefficients used for background fitting.

7. Final Plot:

```
sample.plot()
```

- Generates the final plot of the refined XRD pattern.

Summary

This code snippet showcases a streamlined approach to XRD data refinement within a single Jupyter notebook cell. By controlling plot generation and focusing on a specific data range, the analysis is made more efficient. The comments provide valuable insights into the refinement process and highlight potential areas for further exploration in advanced analysis.

Rate this answer

Enter a prompt here

AI assisted error correction

```
[14] 1 # We can estimate the background using pybaselines tools: https://pybaselines.readthedocs.io/en/latest/  
2 # Baseline estimation is implemented in easyXRD as shown below.  
3  
4  
5 sample = exrd()  
6  
7 sample.load_xrd_data(  
8     from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",  
9     txt_file_wavelength_in_angstrom=0.1814,  
10    plot=False,  
11    radial_range=[1, 9.5],  
12 )  
13  
14 sample.get_baseline()  
15  
16 # The subplot on the right side below shows the baseline subtracted and log scaled data.
```



intentional mistake

```
[15] 1 # We can estimate the background using pybaselines tools: https://pybaselines.readthedocs.io/en/latest/  
2 # Baseline estimation is implemented in easyXRD as shown below.  
3  
4  
5 sample = exrd()  
6  
7 sample.load_xrd_data(  
8     from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",  
9     txt_file_wavelength_in_angstrom=0.1814,  
10    plot=false,  
11    radial_range=[1, 9.5],  
12 )  
13  
14 sample.get_baseline()  
15  
16 # The subplot on the right side below shows the baseline subtracted and log scaled data.
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-15-97a978a29493> in <cell line: 0>()  
8     from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",  
9     txt_file_wavelength_in_angstrom=0.1814,  
--> 10    plot=false,  
11    radial_range=[1, 9.5],  
12 )  
  
NameError: name 'false' is not defined
```

Next steps: [Explain error](#)

NameError

Please explain the error:

```
NameError: name 'false' is not defined
```

Hypothesis

The error message `NameError: name 'false' is not defined` indicates that the variable `false` is being used without being defined. In Python, the boolean value for false is represented by `False`, not `false`. This error typically occurs when a variable is misspelled or when a boolean value is intended but not written in its correct form.

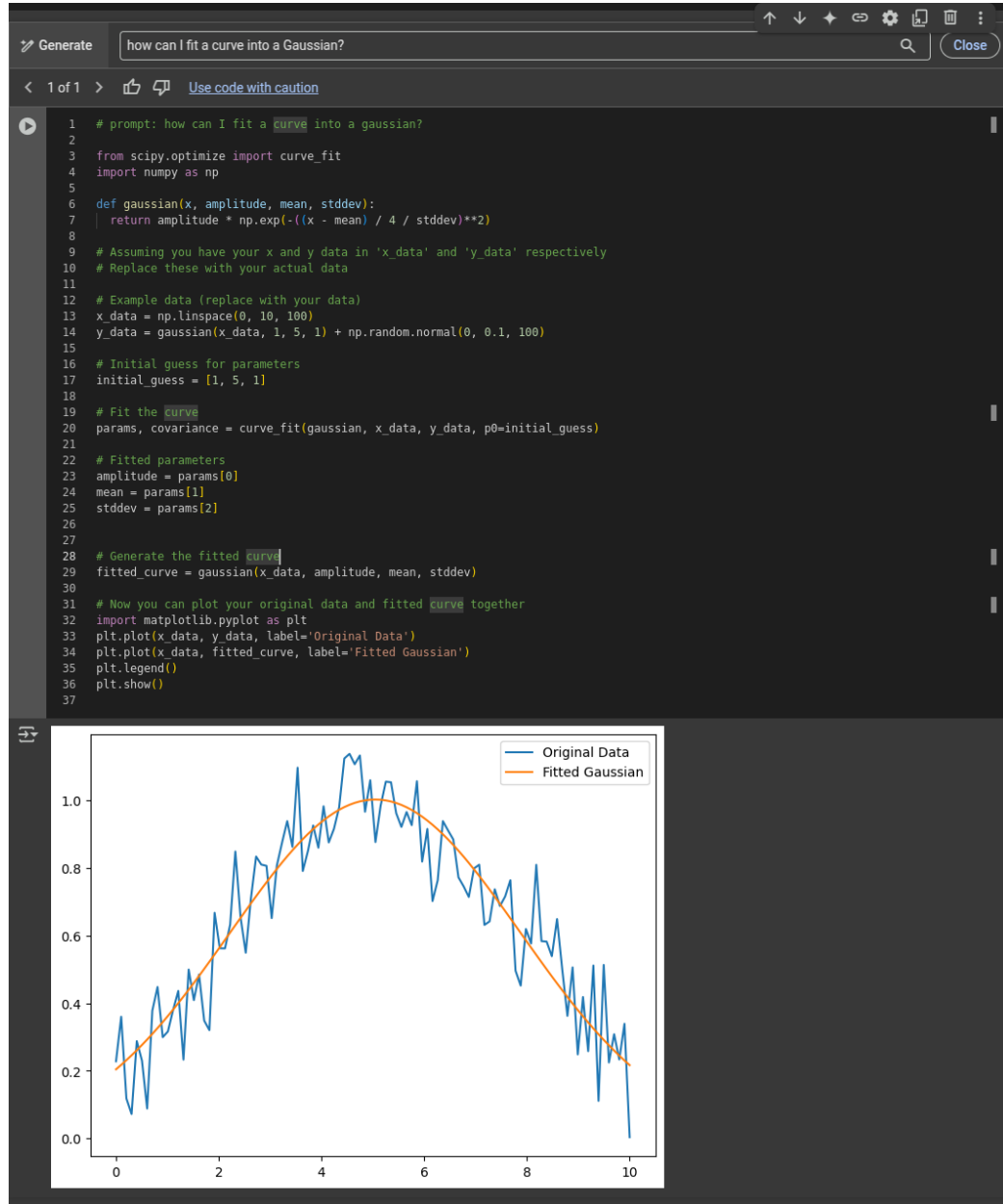
In this case, the user has written `false` on line 10 in `ipython-input-15-97a978a29493` of their notebook while attempting to set the plot parameter, which is case sensitive.

Suggested Changes

```
# We can estimate the background using pybaselines tools: https://pybaselines.readthedocs.io/en/latest/  
# Baseline estimation is implemented in easyXRD as shown below.  
  
sample = exrd()  
  
sample.load_xrd_data(  
    from_txt_file="LaB6/LaB6_synchrotron_XRD_from_28ID-2.xy",  
    txt_file_wavelength_in_angstrom=0.1814,  
    plot=False, # Changed 'false' to 'False'  
    radial_range=[1, 9.5],  
)  
  
sample.get_baseline()  
  
# The subplot on the right side below shows the baseline subtracted and log scaled data.
```

Super useful feature for inexperienced users

If you don't know something, just ask AI



Generate How can I estimate crystalline size from an XRD pattern?

```
1 # prompt: How can I estimate crystalline size from an XRD pattern?
2
3 # Scherrer equation
4 # The Scherrer equation relates the average crystallite size (L) to the broadening of the diffraction peaks.
5
6 # L = Kλ / (βcosθ)
7
8 # where:
9 # L is the average crystallite size
10 # K is the shape factor (typically 0.9)
11 # λ is the wavelength of the X-rays
12 # β is the full width at half maximum (FWHM) of the diffraction peak in radians
13 # θ is the Bragg angle
14
15 # To estimate crystalline size from an XRD pattern using the Scherrer equation in Python:
16
17 import numpy as np
18
19 def calculate_crystallite_size(wavelength, fwhm_radians, bragg_angle, shape_factor=0.9):
20     """Estimates crystallite size using the Scherrer equation.
21
22     Args:
23         wavelength: Wavelength of X-rays in angstroms.
24         fwhm_radians: Full width at half maximum (FWHM) of the diffraction peak in radians.
25         bragg_angle: Bragg angle in degrees.
26         shape_factor: Shape factor (default: 0.9).
27
28     Returns:
29         Crystallite size in angstroms.
30     """
31
32     crystallite_size = (shape_factor * wavelength) / (fwhm_radians * np.cos(np.radians(bragg_angle)))
33     return crystallite_size
34
35 # Example usage (replace with your data):
36 wavelength = 1.5406 # Example wavelength (Cu Kα) in angstroms
37 fwhm_degrees = 0.2 # Example FWHM in degrees
38 bragg_angle = 30 # Example Bragg angle in degrees
39
40 # Convert FWHM from degrees to radians
41 fwhm_radians = np.radians(fwhm_degrees)
42
43 crystallite_size = calculate_crystallite_size(wavelength, fwhm_radians, bragg_angle)
44 print(f"Estimated crystallite size: {crystallite_size:.2f} angstroms")
45
46 # Note: The accuracy of the Scherrer equation is limited.
47 # Factors such as instrumental broadening, non-uniform strain, and crystallite shape can influence the results.
48 # For more accurate crystallite size determination, more sophisticated methods like Whole Powder Pattern Modeling
49 # (WPPM) should be considered, as demonstrated in your provided code using GSAS-II.
50
51
```

Estimated crystallite size: 458.66 angstroms

The word “colab” originates from collaboration and Google Colab has excellent features for enabling collaboration between users.

The screenshot displays a Google Colab notebook interface. At the top, the browser address bar shows the URL: `colab.research.google.com/drive/1-OVFeTxglp2kmTxvRdIAmH2vMnrMrp#scrollTo=joZShujSE35-`. The notebook header includes the Colab logo, the text "colab.ipynb", and a menu with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help".

The code cell contains the following Python code:

```
1 # As next, we need to load a cif file corresponding to expected phases in loaded XRD data.
2 # As the powder is LaB6, The LaB6_structure_from_MaterialsProject.cif is a CIF file that we
3 # got from Materials Project: https://next-gen.materialsproject.org/materials/mp-2688
4 #
5
6 phases = [
7     {
8         "cif": "LaB6/LaB6_structure_from_MaterialsProject.cif",
9         "label": "LaB6",
10    },
11 ]
12 sample.load_phases(phases, plot=True)
13
14 # As you see below, now we have a new subplot with blue stem lines calculated for provided cif file.
15 # Agreement is good and there is no unidentified peaks except a background peak at low q.
```

The notebook displays two XRD plots. The top plot shows the observed intensity (Y_obs) as a function of the scattering vector q (\AA^{-1}). The y-axis is logarithmic, ranging from 10^1 to 10^2 . The x-axis ranges from 0 to 10. The plot shows a series of sharp peaks characteristic of a crystalline material. The bottom plot shows the same XRD data with a fitted model overlaid. The fitted model is represented by blue vertical lines (stem lines) that align with the observed peaks. A legend in the top right corner of the bottom plot identifies the observed data as "Y_obs".

On the right side of the notebook, there is a chat window from Mehmet Topsakal, dated 6:52 PM Today. The message reads: "@mtopsaka@umn.edu Hi Mehmet, will this work for you?".

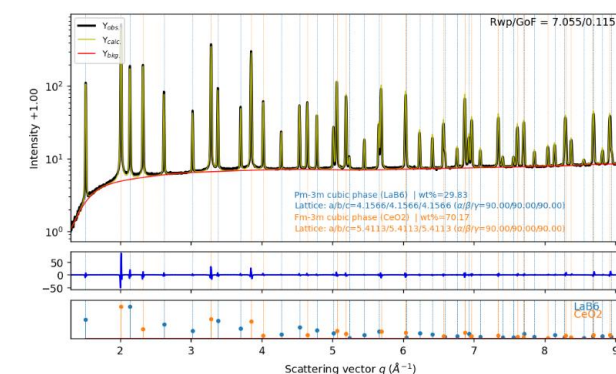
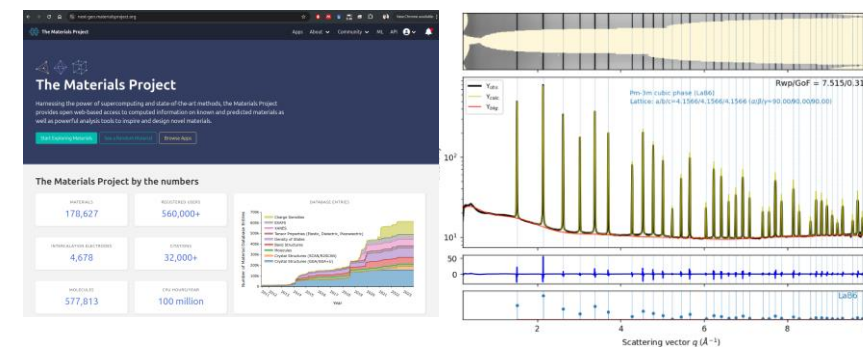
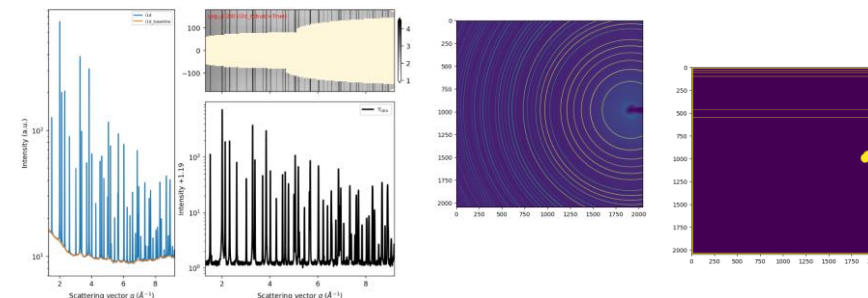
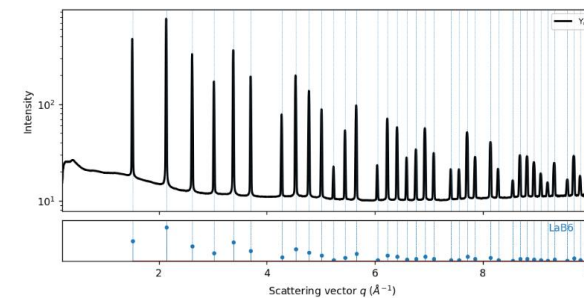
Main functionalities of easyXRD:

Before XRD refinement:

- loading and plotting XRD data from plain text files, tiff files, 1D/2D numpy arrays, xarray datasets, netcdf files...
- Automatic background subtraction and baseline estimation,
- Phase identification using a user provided .cif file or pulling structures from Materials Project,

For XRD refinement (interface to GSAS-II):

- Background refinement,
- Cell refinement,
- Crystalline size / micro strain refinement,
- Phase fractions refinement,
- Site properties (Uiso, occupations) refinement,
- Preferred orientation/texture refinement,
- Advanced plotting...



GitHub - MehmetTopsakal/easyXRD

Product Solutions Resources Open Source Enterprise Pricing

MehmetTopsakal / easyXRD Public

Code Issues Pull requests Actions Projects Security Insights

main 1 Branch 0 Tags

Go to file

About

No description, website, or topics provided.

Readme

GPL-3.0 license

Activity

0 stars

1 watching

0 forks

Report repository

Releases

No releases published

Packages

No packages published

Languages

- Python 100.0%

MehmetTopsakal minor fixes 7f7d599 · 17 hours ago 66 Commits

easyxrd	notebook paths were fixed	17 hours ago
.gitignore	Initial commit	last year
LICENSE	Initial commit	last year
README.md	minor fixes	17 hours ago

README GPL-3.0 license

We have developed a versatile X-ray diffraction (XRD) analysis tool that utilizes modern and open-source Python packages such as pyFAI, xarray, pymatgen, pybaselines,... for data processing/storage and interfaced to Jupyter notebooks powered with actively developed visualization packages such as ipywidgets, and matplotlib. It provides easy access to the Materials Project database which hosts thousands of crystal structures that can be used for phase identification - a critical part of XRD analysis - and utilizes GSAS-II suite for XRD refinements in a user-friendly and intuitive manner. Ultimate goal of this tool is to make X-ray diffraction analysis easy for users and help them to process, refine, store, and share their XRD data conveniently.

You can try the tutorial notebooks below on Google Colab:

- [01_basic.ipynb](#)
- [02_intermediate.ipynb](#)
- [03_advanced-part-1.ipynb](#)
- [03_advanced-part-2.ipynb](#)

You can try these on your own

No installation is required

Thanks



*Special thanks to Collin Knight and NSUF management for Instrument Scientist support program