

# DIFFPY-CMI - A SOFTWARE TOOLBOX FOR REAL-SPACE STRUCTURE ANALYSIS

Simon J. L. Billinge

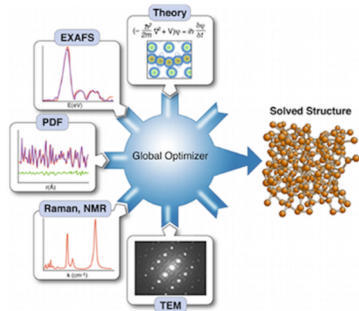
<sup>1</sup>Materials Department,  
University of California, Santa Barbara

January 13, 2026

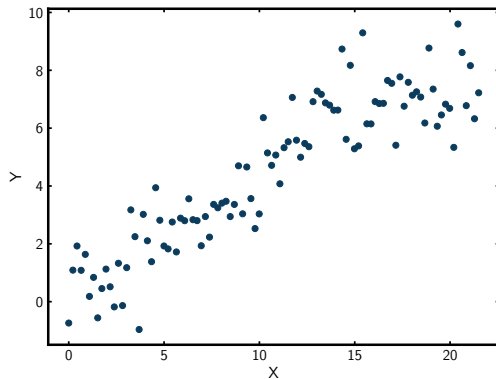
# DIFFPY.CMI

diffpy.cmi

- Who is it for?
- What does it help them do?
- How does it work?
- How do I get it?



# ANATOMY OF A REGRESSION

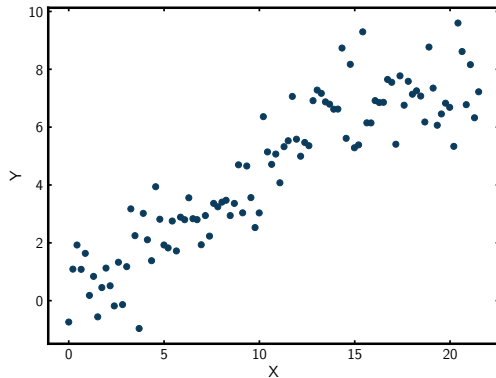


What is the task?

- I have measured some data
- It contains a signal and some noise
- I would like to know the signal

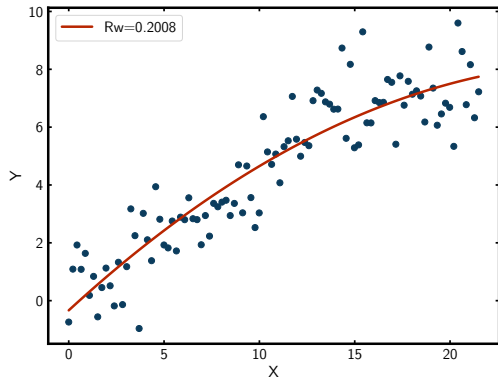
The signal is “the physics” of the problem

# HOW DO I GET THE SIGNAL?



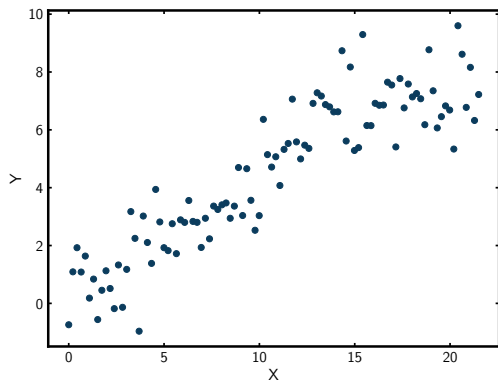
Simple, just draw a smooth line through the noisy data!  
That's it, the smooth line you draw is the signal. Humans are good at this!

# HOW DO I GET THE SIGNAL?



Simple, just draw a smooth line through the noisy data!  
That's it, the smooth line you draw is the signal. Humans are good at this!

# WHAT IF I WANT TO DESIGN A MACHINE TO DO THIS INSTEAD?

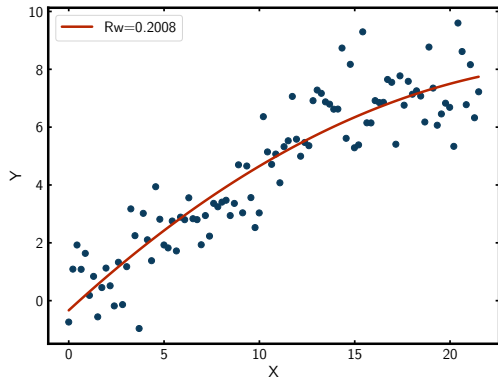


Have the machine mimic the human?

- How did I do it?
- Used my feelings and my innate sense or aesthetics

but machines don't have feelings or innate senses of aesthetics.

# WE HAVE TO BE A BIT CLEVER...



## Assumption:

- there is a function,  $f(x)$ , that maps the point at each value of  $x$  to a value of  $y$
- i.e.,  $y = f(x)$
- We can call  $f(x)$  the “the physics function” because it is the “physics” of the problem.

Task is now to discover  $f(x)$ .

# TASK IS TO DISCOVER $f(x)$

We often have a functional form for  $f(x)$ , e.g., from some known physics,

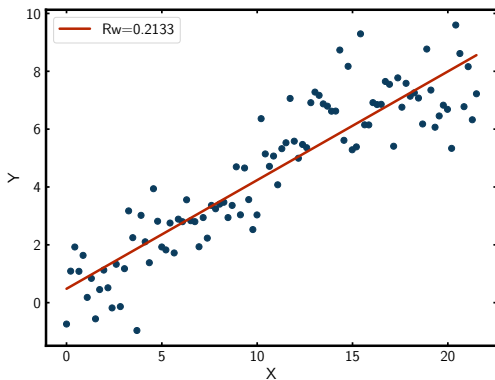
- let's say  $f'(x, \mathbf{p}')$ , that depends on a bunch of parameters,  $\mathbf{p}' = \{p'_1, p'_2, \dots, p'_n\}$
- e.g.  $f$  might be diffraction or PDF equations
- New Task: to estimate the “best values”, for the parameters in  $\mathbf{p}'$
- Note, another possible task is deciding if the function  $f'(x, \mathbf{p}')$  is best, or maybe some other slightly different physics functions,  $f''(x, \mathbf{p}'')$ ,  $f'''(x, \mathbf{p}''')$  and so on, are better.

The former task is called “parameter estimation” and the latter is called “model selection” (more on this later)



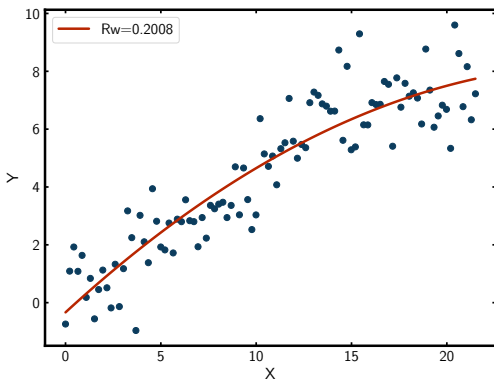
TASK IS TO DISCOVER  $f(x)$

- $f'(x, \mathbf{p}') = mx + c$



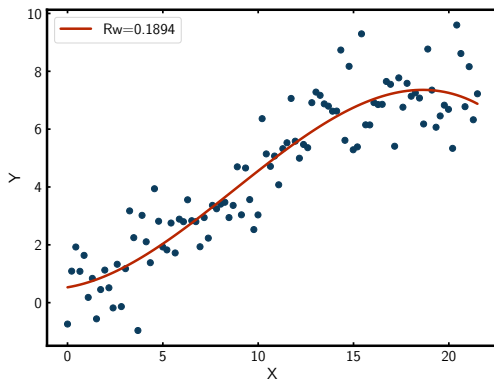
TASK IS TO DISCOVER  $f(x)$ 

- $f''(x, \mathbf{p}'') = p_2x^2 + p_1x + p_0$



TASK IS TO DISCOVER  $f(x)$ 

- $f'''(x, \mathbf{p}''') = p_3x^3 + p_2x^2 + p_1x + p_0$



# BUT HOW DO WE PERSUADE A MACHINE TO ESTIMATE THE PARAMETERS?

Machines are kind of dumb.

- We really only know how to make machines do a few things...
- find extrema (maxima and minima) of functions
- (we also know how to have machines apply logic and do mathematical operations)

# BUT HOW DO WE PERSUADE A MACHINE TO ESTIMATE THE PARAMETERS?

We have to make every problem look like a function extremization problem!

- We don't want to find the maximum of our physics function,
- so we need another function that has the property "the function is extremized when the values of the parameters are "the best they can be"."

We call this function, the Objective function.

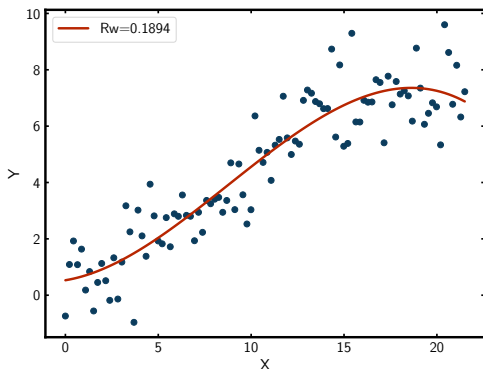
# OBJECTIVE FUNCTION

An Objective function is any function that is at an extremum when we achieve our objective.

- The Objective function is not unique. We can choose it.
- Common ones are “mean square error (L2 norm)”, “absolute error (L1 norm)”,  $\chi^2$ ,  $R_{wp}$ , but there are many many variants. Adding constraints and restraints modifies the Objective function.
- The combination of the “physics function” and the “objective function” we can call the “math(s) model” or just “the model”.

Important: every time you change the objective function, even if you don't change the physics function, you are changing the problem that you are solving. This is not well understood by people.

# HOW DOES THE MACHINE FIND THE EXTREMA OF THE OBJECTIVE FUNCTION?



The machine finds the extrema of the objective function using a regression engine

- The regression engine, or regressor for short, is just an algorithm (and code) the machine uses for finding the optimum of the objective function
- examples are Newton's method, Levenberg-Marquart, BFGS, Metropolis, ...
- Why is there more than one regressor?

## WHY IS THERE MORE THAN ONE REGRESSOR?

- different regressors work better in different situations with different models, for example, depending on the shape of the objective surface as a function of  $\mathbf{p}$ , the number of parameters, the amount of data, the nature of any constraints, and so on.
- Newton's method (if you have analytic derivatives)
- Levenberg-Marquart, BFGS (better), conjugate gradient if you don't
- Nelder-Mead/Powell if data are noisy,
- Metropolis (a variant is used in RMC), stochastic gradient descent, if you have many parameters



# WHY IS THERE MORE THAN ONE REGRESSOR?

Important: Regressors can

- Converge to the right solution
- Converge to the wrong solution (e.g., a metastable minimum),
- Or not converge.
- But **if they converge to the right solution, they will all give the same answer** to the same model. The model is where you are going, the regressor is the vehicle you are travelling in.

Simon says: Describe your **model** in detail not your regressor. Mention your regressor in the context of performance and convergence.

# LAST SLIDE: DON'T MIX UP PARAMETERS AND VARIABLES

We tend to use the words parameters and variables interchangeably but keep them separate in your mind.

- parameters are the  $\mathbf{p}' = \{p'_1, p'_2, \dots, p'_n\}$  that appear in your physics function
- variables are things that the regressor varies
- but wait, doesn't the regressor vary the parameters to find the optimal values?
- yes and no. Variables are mapped on to parameters through constraint equations
- e.g.,  $V_1 = p_1$  would be the most simple such relation
- but if we have  $V_1 = a$ ,  $V_1 = c$ ,  $V_2 = b$ , (and  $\alpha = \beta = \gamma = 90$  and not varying) this implies a tetragonal symmetry. Here we have six parameters but only two variables in our model.

## SUMMARY

- Define your model carefully, including the physics description **and** the objective function including all constraints and restraints
- mention which regressor you are using. If you like, compare a few different regressors to get a better feeling about their performance and convergence characteristics (this is done very rarely but should be done more often)
- Remember, when you change anything in your model (objective or physics) you are solving a different problem

*All models are wrong, some models are useful* (George Box). Get into this mindset and use your models to learn something you want to know (i.e., make them useful).

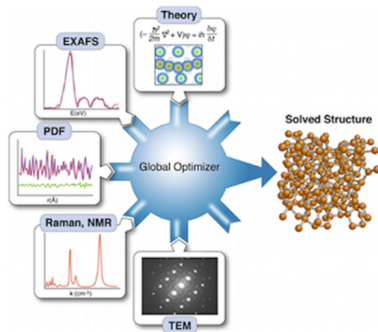
# DIFFPY.CMI

Who is it for?



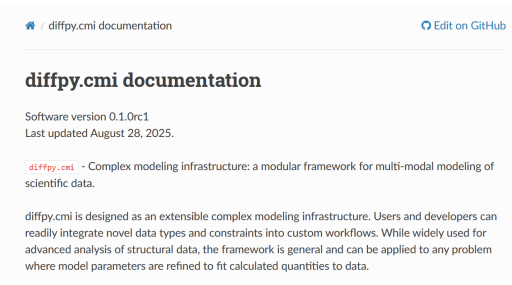
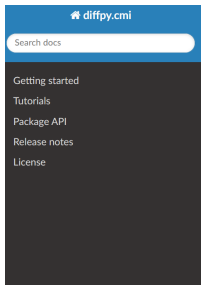
# DIFFPY.CMI

What does it help them do?




# DIFFPY.CMI

## How do I get it?



- 1 `conda install diffpy.cmi`
- 2 `diffpy.org/diffpy.cmi`
- 3 (`https://github.com/diffpy/diffpy.cmi`)

# DIFFPY.CMI

 diffpy.cmi

Search docs

Getting started

- Installation
- Pack and Profile Installation
- Data and Examples

Tutorials

- Package API
- Release notes
- License

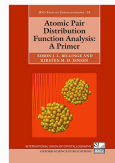
[Home](#) / Getting started[Edit on GitHub](#)

## Getting started

For detailed instructions and in-depth examples of modeling with `diffpy.cmi`, we highly recommend the book,

*Atomic Pair Distribution Function Analysis: A Primer* by Simon J. L. Billinge and Kirsten M. Ø. Jensen (Oxford University Press, 2023).

To purchase this book, please visit [this link](#).

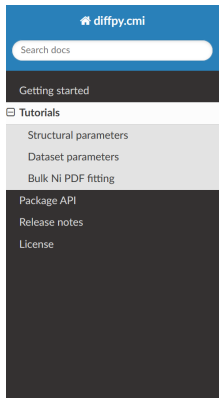


## Installation

To install `diffpy.cmi`, create a new conda environment or activate an existing environment and install the package from the conda-forge channel.

```
conda create -n diffpy.cmi-env
conda install -c conda-forge diffpy.cmi
conda activate diffpy.cmi-env
```

# DIFFPY.CMI

[Home](#) / Tutorials[Edit on GitHub](#)

## Tutorials

The atomic pair distribution function (PDF) provides a powerful route to understanding the local atomic structure of materials in real space. Despite its utility, PDF fitting and analysis can be challenging. The process of understanding PDFs requires time, care, and the development of intuition to connect structural models to experimental data. Hopefully by the end of this tutorial series, PDF fitting will go from being a mysterious black box to a powerful tool in your structural analysis.

Example usage of `diffpy.cmi` can be found at [this GitHub repo](#).

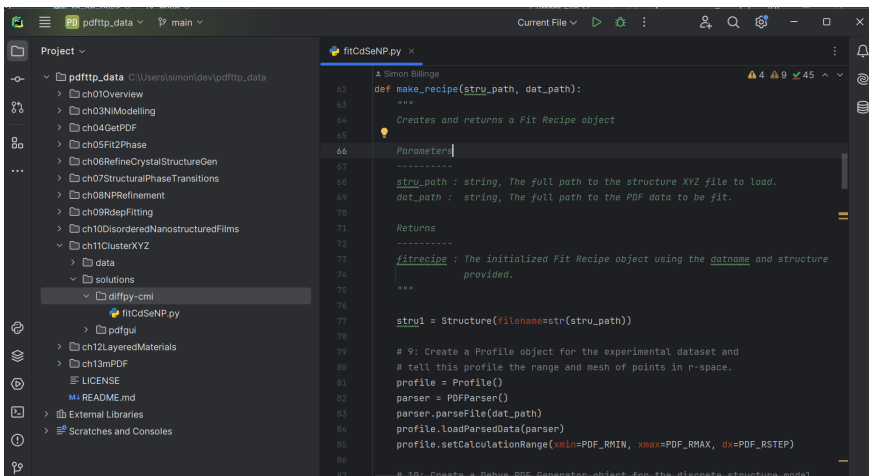
## Structural parameters

Before we start fitting PDFs, let's first understand the parameters that will be refined on and what they mean. It is important to understand these parameters as they will help you gain better insight into the fitting process and how to interpret the results.

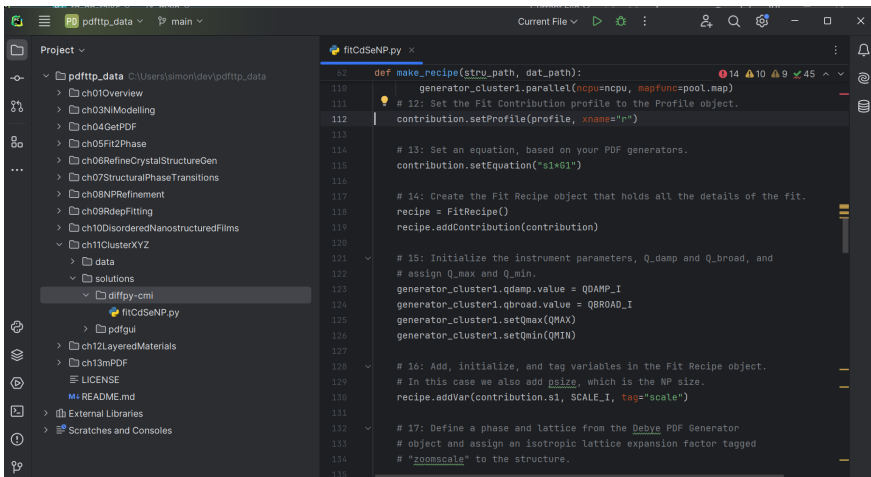
PDF Parameter	Description	Refineable?	Structural or Instrumental?
------------------	-------------	-------------	--------------------------------



# DIFFPY-CMI SCRIPT EXAMPLE



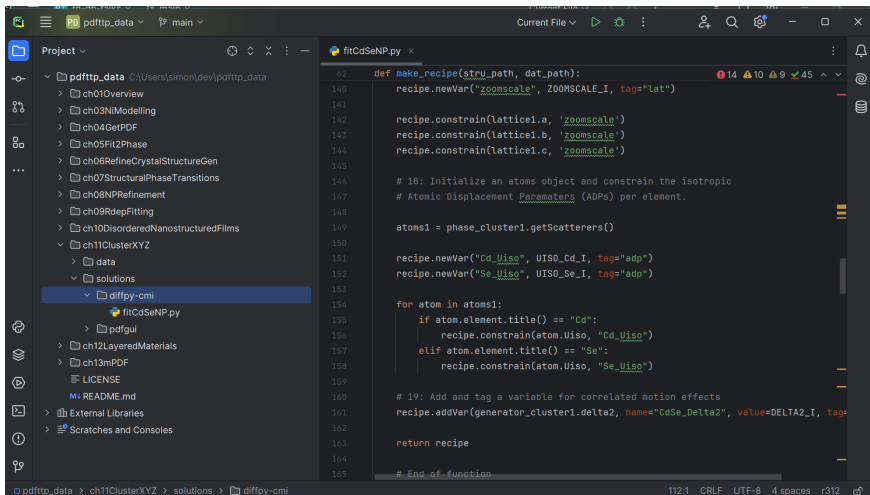
# DIFFPY-CMI SCRIPT EXAMPLE



The screenshot shows an IDE with a project structure on the left and a Python script on the right. The project structure includes a folder named 'diffpy-cmi' containing a file 'fitCdSeNP.py'. The script 'fitCdSeNP.py' contains the following code:

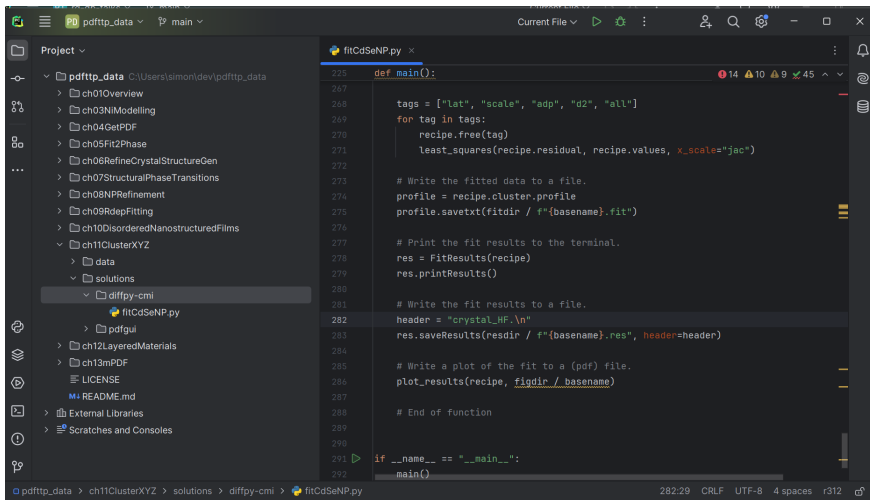
```
62 def make_recipe(stru_path, dat_path):
110     generator_cluster1.parallel(ncpu=ncpu, mapfunc=pool.map)
111     # 12: Set the Fit Contribution profile to the Profile object.
112     contribution.setProfile(profile, xname="r")
113
114     # 13: Set an equation, based on your PDF generators.
115     contribution.setEquation("s1*G1")
116
117     # 14: Create the Fit Recipe object that holds all the details of the fit.
118     recipe = FitRecipe()
119     recipe.addContribution(contribution)
120
121     # 15: Initialize the instrument parameters, Q_damp and Q_broad, and
122     # assign Q_max and Q_min.
123     generator_cluster1.qdamp.value = QDAMP_I
124     generator_cluster1.qbroad.value = QBROAD_I
125     generator_cluster1.setQmax(QMAX)
126     generator_cluster1.setQmin(QMIN)
127
128     # 16: Add, initialize, and tag variables in the Fit Recipe object.
129     # In this case we also add psize, which is the NP size.
130     recipe.addVar(contribution.s1, SCALE_I, tag="scale")
131
132     # 17: Define a phase and lattice from the Debye PDF Generator
133     # object and assign an isotropic lattice expansion factor tagged
134     # "zoomscale" to the structure.
135
```

# DIFFPY-CMI SCRIPT EXAMPLE



```
62 def make_recipe(stru_path, dat_path):
140     recipe.newVar("zoomscale", ZOOMSCALE_I, tag="lat")
141
142     recipe.constrain(lattice1.a, 'zoomscale')
143     recipe.constrain(lattice1.b, 'zoomscale')
144     recipe.constrain(lattice1.c, 'zoomscale')
145
146     # 18: Initialize an atoms object and constrain the isotropic
147     # Atomic Displacement Parameters (ADPs) per element.
148
149     atoms1 = phase_cluster1.getScatterers()
150
151     recipe.newVar("Cd_Uiso", UIISO_Cd_I, tag="adp")
152     recipe.newVar("Se_Uiso", UIISO_Se_I, tag="adp")
153
154     for atom in atoms1:
155         if atom.element.title() == "Cd":
156             recipe.constrain(atom.Uiso, "Cd_Uiso")
157         elif atom.element.title() == "Se":
158             recipe.constrain(atom.Uiso, "Se_Uiso")
159
160     # 19: Add and tag a variable for correlated motion effects
161     recipe.addVar(generator_cluster1.delta2, name="CdSe_Delta2", value=DELTA2_I, tag=
162
163     return recipe
164
165 # End of function
```

# DIFFPY-CMI SCRIPT EXAMPLE



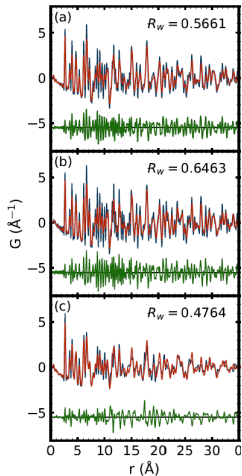
```
fitCdSeNP.py x
225 def main():
267
268     tags = ["lat", "scale", "adp", "d2", "all"]
269     for tag in tags:
270         recipe.free(tag)
271         least_squares(recipe.residual, recipe.values, x_scale="jac")
272
273     # Write the fitted data to a file.
274     profile = recipe.cluster.profile
275     profile.savetxt(fitdir / f"{basename}.fit")
276
277     # Print the fit results to the terminal.
278     res = FitResults(recipe)
279     res.printResults()
280
281     # Write the fit results to a file.
282     header = "crystal_HF.\n"
283     res.saveResults(resdir / f"{basename}.res", header=header)
284
285     # Write a plot of the fit to a (pdf) file.
286     plot_results(recipe, figdir / basename)
287
288     # End of function
289
290
291 if __name__ == "__main__":
292     main()
```

pdfftp\_data > ch11ClusterXYZ > solutions > diffpy-cmi > fitCdSeNP.py 282:29 CRLF UTF-8 4 spaces r312

# DIFFPY.MORPH

## DIFFPY.MORPH

# THREE SAMPLES, WHICH HAS A PHASE TRANSITION?

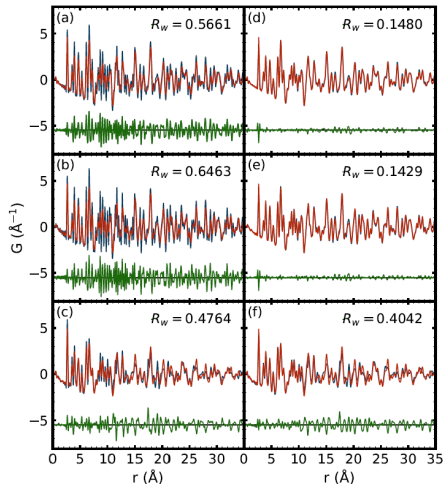


Three samples, comparing low (blue) and high (red) temperature PDFs:

- $\text{Ir}_{0.8}\text{Rh}_{0.2}\text{Te}_2$
- $\text{Ir}_{0.95}\text{Pt}_{0.05}\text{Te}_2$
- $\text{IrTe}_2$

which has a phase transition?

# THREE SAMPLES, WHICH HAS A PHASE TRANSITION?

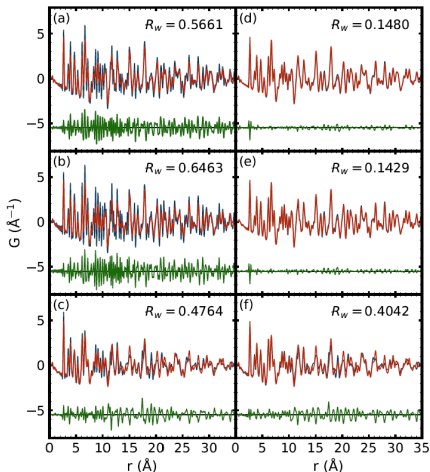


Three samples, comparing low (blue) and high (red) temperature PDFs:

- $\text{Ir}_{0.8}\text{Rh}_{0.2}\text{Te}_2$
- $\text{Ir}_{0.95}\text{Pt}_{0.05}\text{Te}_2$
- $\text{IrTe}_2$

$\text{IrTe}_2$  has a phase transition!

# HOW DOES DIFFPY.MORPH WORK?



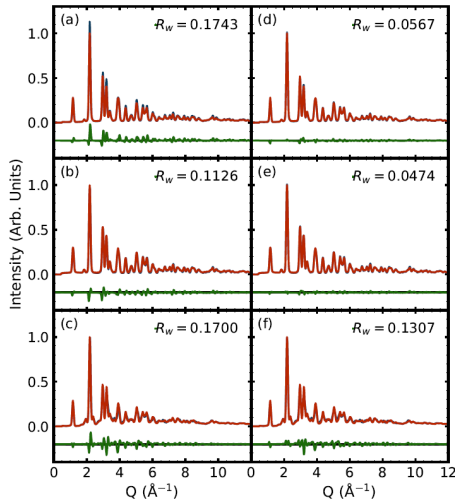
Apply “morphs” to account for “boring” effects such as thermal expansion

- stretch morph (linearly) stretches the x-axis
- scale morph scales the y-axis
- smear morph convolutes with a Gaussian to change peak width
- Vary the magnitude of the morphs until a “morphed” signal best matches a “target” signal.

In the PDF, stretch mimics thermal expansion, smear mimics increased thermal motion

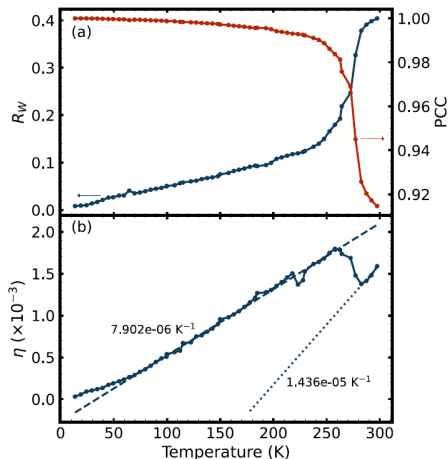


# DOES IT JUST WORK ON PDF?



No, it works on diffraction data too  
in fact, it works on any signal in principle

# I HAVE A FOLDER FULL OF T-DEP DATA

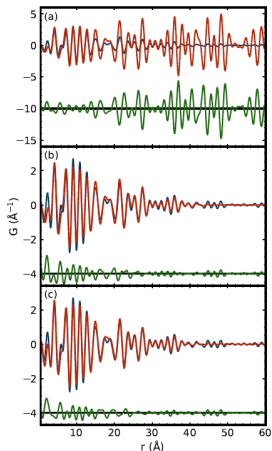


Use DIFFPY.MORPH to extract the thermal expansivity

- IrTe<sub>2</sub>

in fact, find  $T_c$  and get the thermal expansivity of each phase

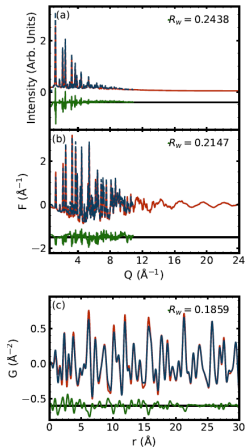
# LET'S GET GREEDY AND HAVE DIFFPY.MORPH DO MORE AND MORE



## Additional morphs

- particle shape function in the PDF, refine (major and minor) radius
- shift in x and y
- non-linear (polynomial) stretch (squeeze)
- funcx, funcy, funcxy - user-defined python function morphs
- multiple morphs (chain the morphs)

# LET'S GET GREEDY AND HAVE DIFFPY.MORPH DO MORE AND MORE



## Fancy use of funcxy

- let's use DIFFPY.MORPH to find optimal parameters in PDFGETX3!
- Give it a target  $F(Q)$  (e.g., measured at a synchrotron or calculated)
- Give it pdfgetter() as the function in funcxy.
- Have it vary rpoly, background scale, etc.

# SCIKIT PACKAGE

## Scikit Package

- Who is it for?
- What does it help them do?
- How does it work?
- How do I get it?



# SCIKIT PACKAGE

Who is it for?



# SCIKIT PACKAGE

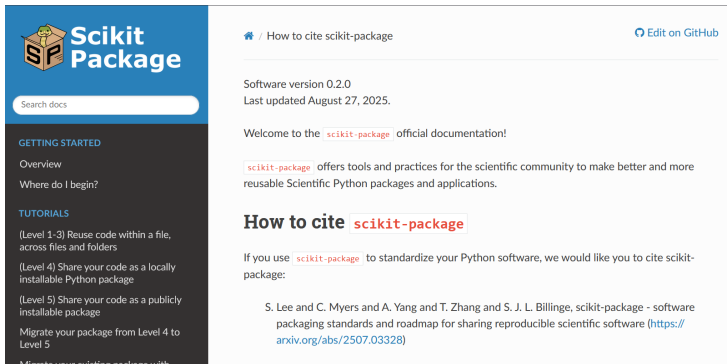
What does it help them do?



**"I wrote a software program and sold it for two million dollars."**

# SCIKIT PACKAGE

## How do I get it?



**Scikit Package**

Search docs

**GETTING STARTED**

- Overview
- Where do I begin?

**TUTORIALS**

- (Level 1-3) Reuse code within a file, across files and folders
- (Level 4) Share your code as a locally installable Python package
- (Level 5) Share your code as a publicly installable package
- Migrate your package from Level 4 to Level 5
- Migrate your existing package with

[How to cite scikit-package](#) [Edit on GitHub](#)

Software version 0.2.0  
Last updated August 27, 2025.

Welcome to the `scikit-package` official documentation!

`scikit-package` offers tools and practices for the scientific community to make better and more reusable Scientific Python packages and applications.

### How to cite `scikit-package`

If you use `scikit-package` to standardize your Python software, we would like you to cite scikit-package:

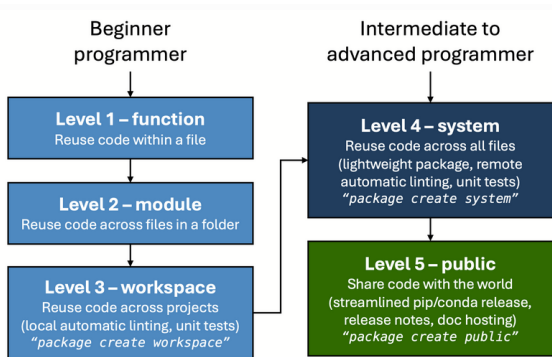
S. Lee and C. Myers and A. Yang and T. Zhang and S. J. L. Billinge, scikit-package - software packaging standards and roadmap for sharing reproducible scientific software (<https://arxiv.org/abs/2507.03328>)

- ① `conda install scikit-package`
- ② `scikit-package.github.io/scikit-package`
- ③ (<https://github.com/scikit-package/scikit-package>)



# SCIKIT PACKAGE

How does it work?



# SCIKIT PACKAGE

- Level 1, `function`, which is already widely used, consists of simply defining functions within the same file or module.
- Level 2, `module`, expands on Level 1 by reusing functions across separate module files within the same directory.
- Level 3, `workspace`, restructures the organization so that a block of code can be reused across multiple projects.
- Level 4, `system`, enables users to create a lightweight package so that the code can be reused across all files locally.
- Level 5, `public`, is the final step, where the source code is uploaded online so that anyone in the world can install the package, sourced from PyPI or conda-forge.

# SCIKIT PACKAGE

## EXAMPLES

### Overview

Example 1. Create your first Level 4 package

Example 2. Create your first Level 5 public package

Example 3. Create a package with branded namespace import at Level 5

Example 4. Migrate an existing package to Level 5

## PROGRAMMING GUIDES

## Overview

Here we provide 4 representative examples to illustrate how to use `scikit-package` for building shareable packages at Levels 4 and 5. We also include an example of how to use `scikit-package` to migrate your existing packages to the `scikit-package` standards.

- [Example 1. Create your first Level 4 package](#)
- [Example 2. Create your first Level 5 public package](#)
- [Example 3. Create a package with branded namespace import at Level 5](#)
- [Example 4. Migrate an existing package to Level 5](#)

# SCIKIT PACKAGE

## Example 1. Create your first Level 4 package

In this example, we assume **Mr Neutron** previously initiated a project called `diffraction-utils` using `scikit-package` Level 3 and developed a shared class called `DiffractionObject`. This `DiffractionObject` is used in code analyzing various diffraction data sourced from x-ray, neutron, and electron instruments. This class is written in a module called `diffraction_objects.py` which is reused in Python scripts in the `scattering` sub-project folder. **Mr Neutron's** folder structure looks like the following:

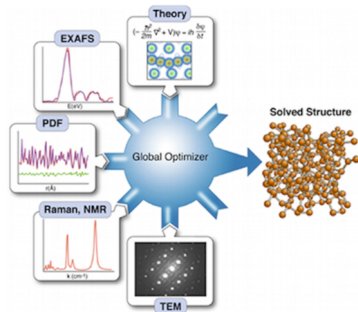
```
somewhere/  
|-- on/  
    |-- my/  
        |-- computer/  
            |-- diffraction-utils/  
                |-- README.md  
                |-- diffraction_objects.py  
                |-- scattering/  
                    |-- __init__.py  
                    |-- neutron.py  
                    |-- xray.py  
                    |-- electron.py
```

Since the project is a data analysis project, **Mr Neutron** followed common practice to place this

# DIFFPY.CMI TUTORIAL

Quick view of the workshop  
tomorrow/Wednesday

- Live demonstration by Caden Myers
- Installation
- Hands-on examples for you to code
- Simple fitting of a user-defined function
- Structural fitting to a PDF
- T-dep fitting



## Temporary page!

$\text{\LaTeX}$  was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it. If you rerun the document (without altering it) this surplus page will go away, because  $\text{\LaTeX}$  now knows how many pages to expect for this document.