DiffPy-CMI – a software toolbox for real-space structure analysis and Complex Modeling

Pavol Juhas
Brookhaven National Laboratory

School and Conference on Analysis of Diffraction Data in Real Space
Grenoble, 17-22 March 2019

March 8, 2016

Methods for analyzing experimental PDFs

direct readout

- bond distances and their variations

big box modeling

- Reverse Monte Carlo (RMC) [Pusztai & McGreevy, Physica B, 234-236, (1997)]
- ~10^4 atoms in a large box with periodic boundary conditions
- MC position optimization → excellent fit to the experimental PDF
- many degrees of freedom – RMC modeling requires constraints to produce physically feasible structures
- interpretation of 10^4 coordinates – bond length and angle statistics

small box modeling

- up to ~100 atoms in a small cell with periodic boundary conditions
- PDF modeling can be focused to a short, specific length scale
- **simple refinement**
  - start with reasonably accurate initial structure
  - downhill minimization of the model variables to fit observed PDF
- **structure determination**
  - extract experimental pair-distances from the PDF
  - find shape that reproduces the same set of pair distances
- **complex refinement**
  - additional cost terms (atom overlap, bond valence sums,…)
  - mixed contributions from molecules and crystalline phases

PDFgui limitations

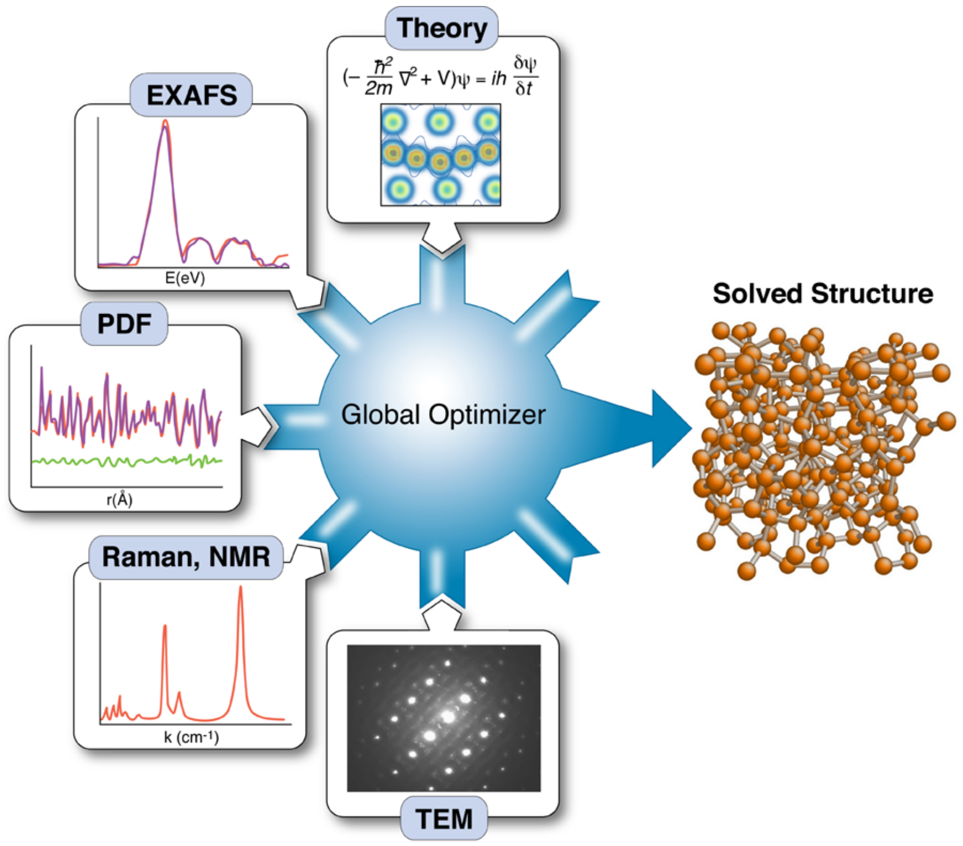
- periodic structures only, no support for isolated particles
- particle size corrections only for spherical shape,
  no option for ellipsoid, cylinder, prism, etc.
- internal structure representation is very basic – no options for molecules, rigid groups, bond length/angle restraints, no internal support for representation as space group with asymmetric unit
- limited math support within constraint formulas
- no way to adjust PDF calculations by user-defined profile functions or improved peak-width formulas
- no option to incorporate other information sources (theory or experiment) into optimization
- PDF-calculation engine PDFfit2 is hard to maintain or extend

advantages

- easy to learn, no need to know any programming language
- [simple] simulations are easy to setup and process

BROOKHAVEN
NATIONAL LABORATORY

Complex modeling



Problem

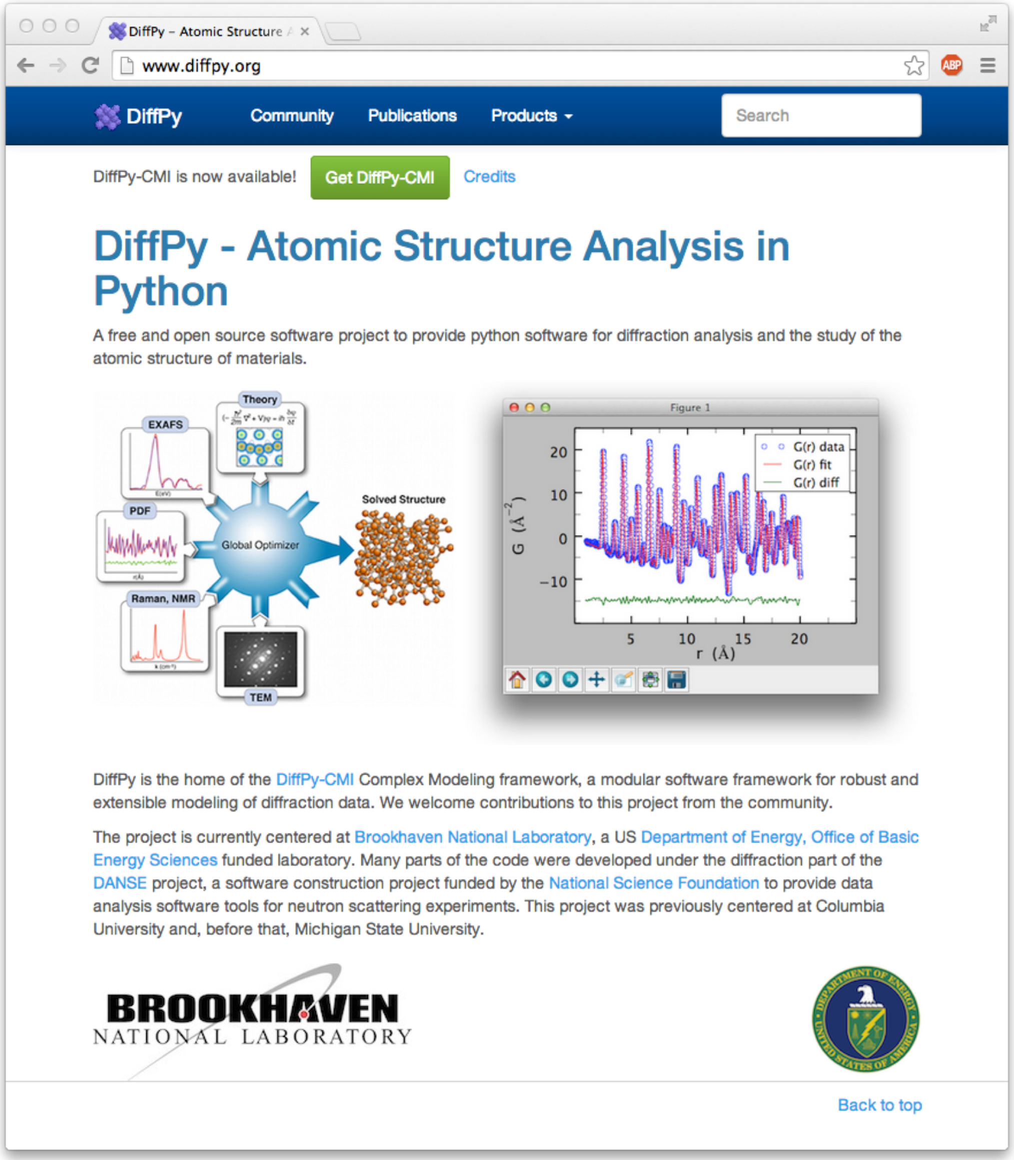
- not enough information in the available experimental data

Remedy

- collect data from multiple experimental techniques
- use additional knowledge about the studied material - chemical constraints, rigid units, bond-valence sums, energy calculation
- combine all experimental and theoretical inputs about the structure in one optimization scheme
- requires flexible software tools to setup custom models adaptable for specifics of studied materials.

BROOKHAVEN NATIONAL LABORATORY

DiffPy-CMI – Complex Modeling Infrastructure



DiffPy-CMI is now available!

DiffPy - Atomic Structure Analysis in Python

A free and open source software project to provide python software for diffraction analysis and the study of the atomic structure of materials.

DiffPy is the home of the DiffPy-CMI Complex Modeling framework, a modular software framework for robust and extensible modeling of diffraction data. We welcome contributions to this project from the community.

The project is currently centered at Brookhaven National Laboratory, a US Department of Energy, Office of Basic Energy Sciences funded laboratory. Many parts of the code were developed under the diffraction part of the DANSE project, a software construction project funded by the National Science Foundation to provide data analysis software tools for neutron scattering experiments. This project was previously centered at Columbia University and, before that, Michigan State University.

BROOKHAVEN NATIONAL LABORATORY

- tools for PDF, BVS, SAS simulations, structure data handling, multi-input optimizations
- Python and C++, object-oriented, reusable, extensible libraries
- available from **https://www.diffpy.org** for Linux, Mac, UNIX systems

upgrade release March 2019

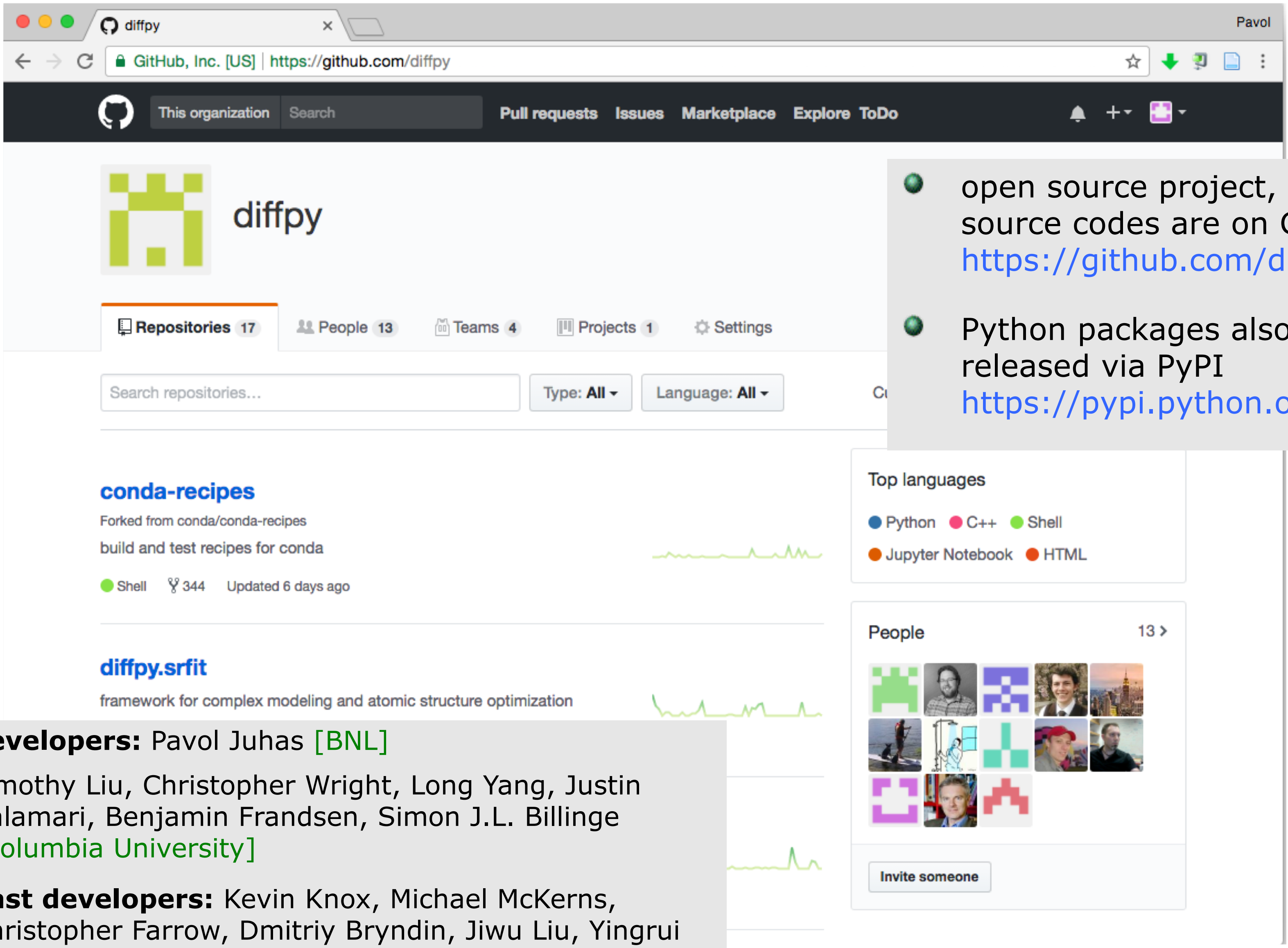
- added support for Python 3

Anaconda Python installation

```
$ conda install –c diffpy diffpy-cmi
```

BROOKHAVEN NATIONAL LABORATORY

DiffPy-CMI – open source project



diffpy

GitHub, Inc. [US] | https://github.com/diffpy

This organization | Search | Pull requests | Issues | Marketplace | Explore | ToDo

diffpy

Repositories 17 | People 13 | Teams 4 | Projects 1 | Settings

Search repositories... | Type: All | Language: All

conda-recipes
Forked from conda/conda-recipes
build and test recipes for conda
● Shell   344   Updated 6 days ago

diffpy.srfit
framework for complex modeling and atomic structure optimization

Top languages
● Python ● C++ ● Shell ● Jupyter Notebook ● HTML

People 13

Invite someone

- open source project, all source codes are on GitHub https://github.com/diffpy
- Python packages also released via PyPI https://pypi.python.org

developers: Pavol Juhas [BNL]

Timothy Liu, Christopher Wright, Long Yang, Justin Calamari, Benjamin Frandsen, Simon J.L. Billinge [Columbia University]

past developers: Kevin Knox, Michael McKerns, Christopher Farrow, Dmitriy Bryndin, Jiwu Liu, Yingrui Shang, Peng Tian, Wenduo Zhou, Milinda Abeykoon, Emil Bozin, Timur Davis

DiffPy-CMI software

DiffPy-CMI is

- Complex Modeling Infrastructure – a software toolbox for multi-probe structure analysis
- collection of Python and C++ libraries responsible for tasks needed in structure analysis (structure representation, forward calculators, refinement configuration)
- highly configurable, extensible. Amendable for extensions, such as more calculators and more structure representations. Calculations of structure-derived properties can be tweaked at runtime.

advantages

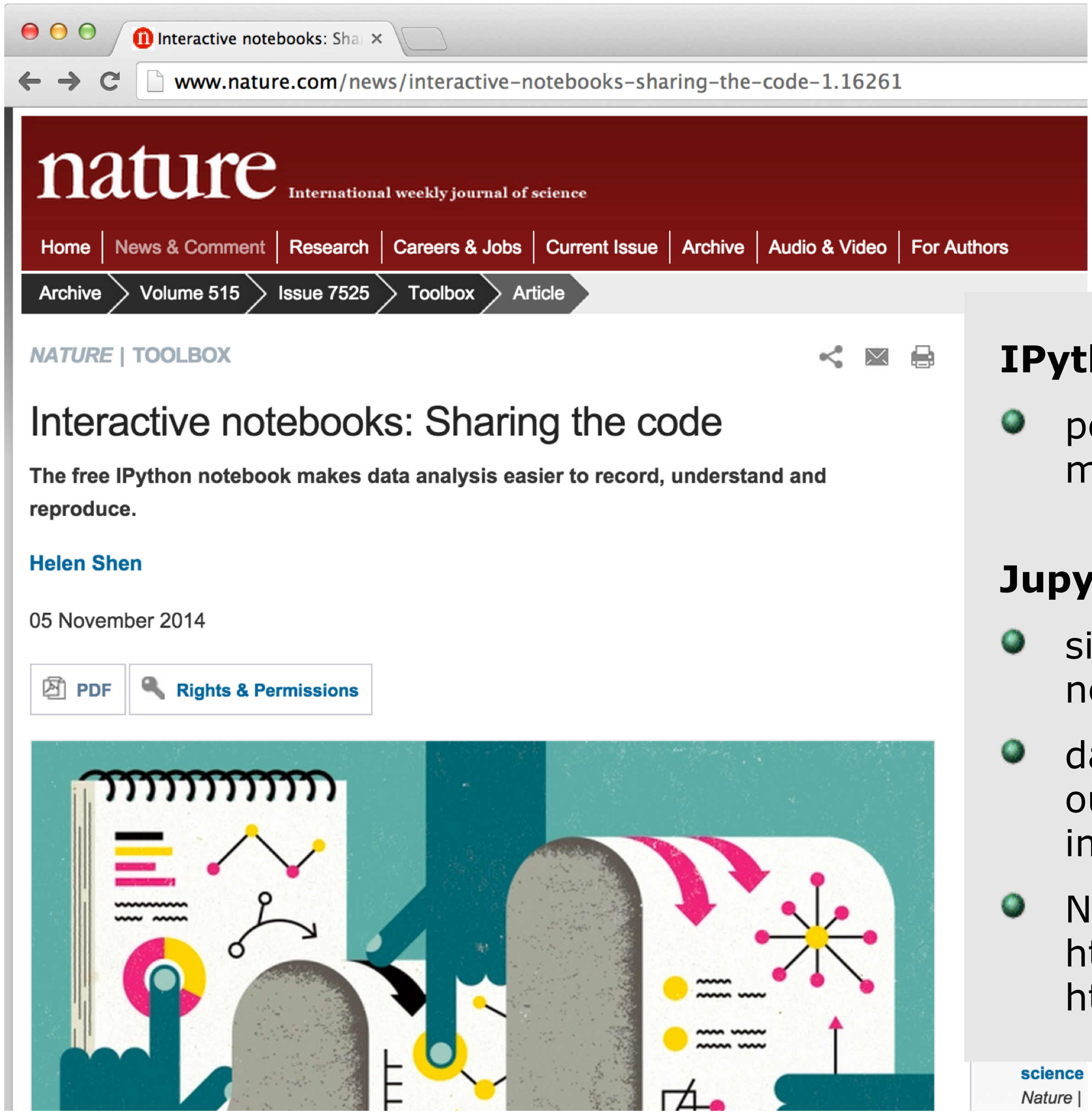
- allows simulations not viable in PDFgui
- plenty of options for configuring refinements, structure models, simulations, interfacing with other analysis programs…

drawbacks

- steep learning curve
- no GUI, can be only used from Python scripts
- requires fair knowledge of Python

BROOKHAVEN
NATIONAL LABORATORY

IPython notebooks – data analysis sharing



Interactive notebooks: Sha ×

www.nature.com/news/interactive-notebooks-sharing-the-code-1.16261

nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

Archive > Volume 515 > Issue 7525 > Toolbox > Article

NATURE | TOOLBOX

Interactive notebooks: Sharing the code

The free IPython notebook makes data analysis easier to record, understand and reproduce.

Helen Shen

05 November 2014

PDF | Rights & Permissions

science

Nature |

IPython

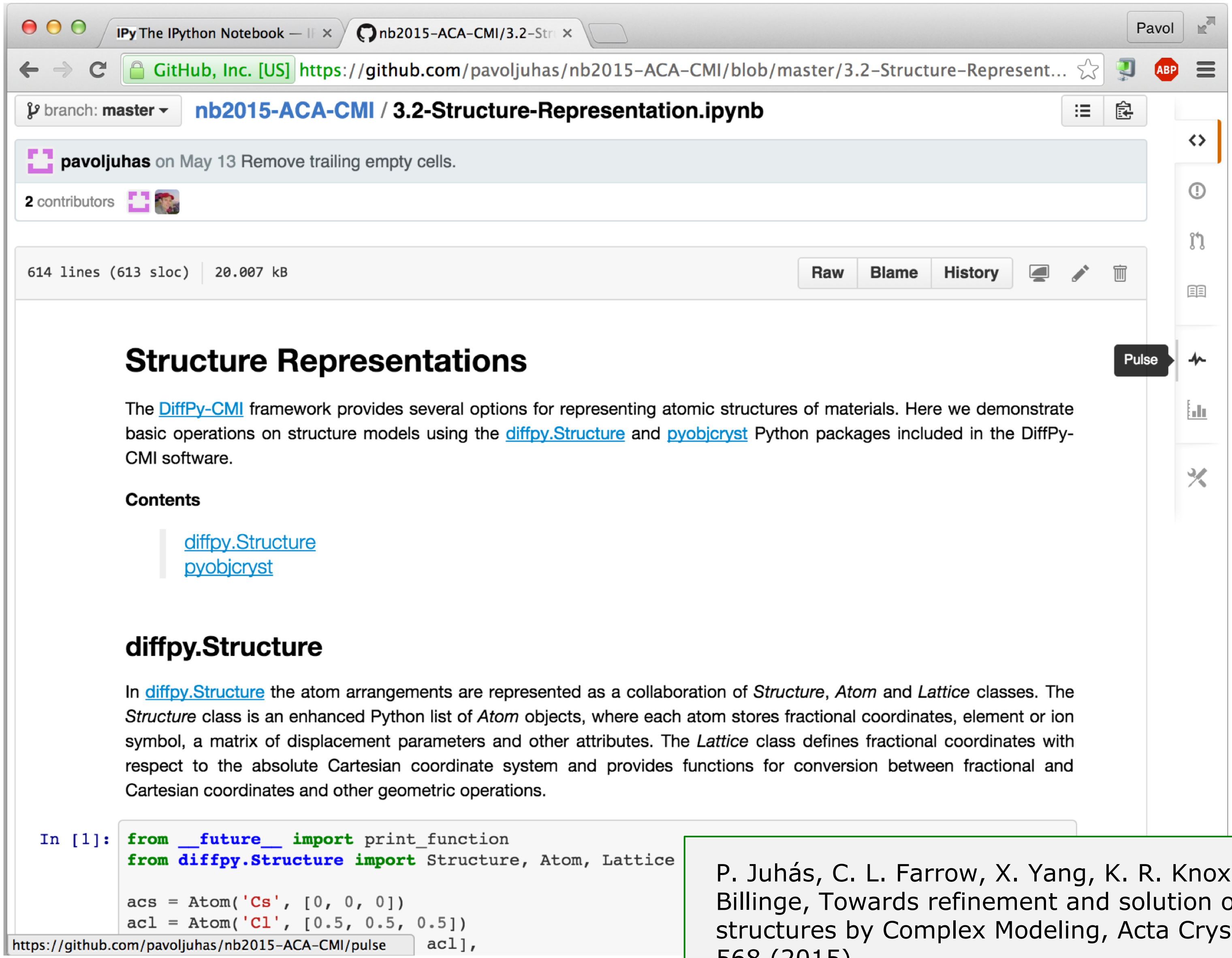
- powerfull Python shell with many data analysis features

Jupyter notebooks

- similar to Mathematica notebook format
- data analysis code and its outputs, formatted notes, images, media
- NB rendering and sharing at https://nbviewer.jupyter.org https://github.com

BROOKHAVEN NATIONAL LABORATORY

IPython notebooks – data analysis sharing



IPy The IPython Notebook — I ×
nb2015-ACA-CMI/3.2-Str ×
Pavol
GitHub, Inc. [US] https://github.com/pavoljuhas/nb2015-ACA-CMI/blob/master/3.2-Structure-Represent...
branch: master
nb2015-ACA-CMI / 3.2-Structure-Representation.ipynb
pavoljuhas on May 13 Remove trailing empty cells.
2 contributors
614 lines (613 sloc) | 20.007 kB
Raw
Blame
History
Pulse
https://github.com/pavoljuhas/nb2015-ACA-CMI/pulse

Structure Representations

The DiffPy-CMI framework provides several options for representing atomic structures of materials. Here we demonstrate basic operations on structure models using the diffpy.Structure and pyobjcryst Python packages included in the DiffPy-CMI software.

Contents

> diffpy.Structure
> pyobjcryst

diffpy.Structure

In diffpy.Structure the atom arrangements are represented as a collaboration of *Structure*, *Atom* and *Lattice* classes. The *Structure* class is an enhanced Python list of *Atom* objects, where each atom stores fractional coordinates, element or ion symbol, a matrix of displacement parameters and other attributes. The *Lattice* class defines fractional coordinates with respect to the absolute Cartesian coordinate system and provides functions for conversion between fractional and Cartesian coordinates and other geometric operations.

```
In [1]: from __future__ import print_function
        from diffpy.Structure import Structure, Atom, Lattice

        acs = Atom('Cs', [0, 0, 0])
        acl = Atom('Cl', [0.5, 0.5, 0.5])
                          acl],
```

P. Juhás, C. L. Farrow, X. Yang, K. R. Knox and S. J. L. Billinge, Towards refinement and solution of complicated structures by Complex Modeling, Acta Crystallogr. A 71, 562-568 (2015).

DiffPy-CMI – functionality overview

Structure Representation

- **diffpy.structure** → simple storage of P1 periodic structures, finite clusters, input and ouput for CIF, PDB, xyz, pdffit, discus formats. Space group definitions, symmetry expansion, generation of symmetry-based constraints.
- **pyobjcryst** → advanced structure representations, crystals with space group, crystals containing rigid molecules, bond-length and bond-angle restraints, z-matrix representation. Input and output in custom XML and CIF formats. Python interface to the ObjCryst++ crystallographic library by V. Favre-Nicolin.

Forward Calculators

- **diffpy.srreal** → calculators of pair-interaction-derived quantities, such as PDF, Debye sum, bond lengths, bond valence sums, overlap of empirical atom radii.
- **pyobjcryst** → powder and single-crystal diffraction patterns.
- **srfit-sasview** → selected functions for Small Angle Scattering simulations from the SasView program, http://www.sasview.org

Fit configuration and management

- **diffpy.srfit** → setup and control of general fitting problems, control of constraints and restraints, setup of refinements to multiple data sources, simple analysis of fit results.

C++ libraries

- **libdiffpy** – computationally expensive parts - PDF, BVS, etc.
- **libObjCryst** – free objects for crystallography by Vincent Favre-Nicolin, [J. Appl. Cryst. 35 (2002), 734-743].

BROOKHAVEN
NATIONAL LABORATORY

PairQuantity

- the base calculator – shared recipe for evaluating physical quantities based on pair-interactions. Used as a blueprint for all calculators.

$$P(r_1, r_2, \ldots, r_N) = \sum_{i,j}^{N} p(r_{ij})$$

- support for partial sums
- optional upper and lower distance bounds
- support for parallel evaluation

related:

- class StructureAdapter
  - translate structure representations to a calculator-compatible format
  - implemented for PDFgui Structure representation and for Crystal and Molecule objects in the ObjCryst C++ library
    [V. Favre-Nicolin, J. Appl. Cryst. 35 (2002), 734-743]
  - extensible for any structure representation method in Python or C++

BondCalculator

- calculate oriented bond vectors up to a specified distance limit
- optional filtering by atom types, site indices, direction cones

example:

```
>>> from pyobjcryst import loadCrystal
>>> from diffpy.srreal.bondcalculator import BondCalculator
>>> rutile = loadCrystal('TiO2_rutile.cif')
>>> bc = BondCalculator(rmax=2)
>>> bc(rutile)
array([ 1.94720295,  1.94720295,  1.94720295,  1.94720295,  1.94720295,
        1.94720295,  1.98177183,  1.98177183,  1.98177183])
>>> for i in zip(bc.distances, bc.types0, bc.types1, bc.directions):
...     print(i)
...
(1.9472029472402153, 'Ti', 'O', array([-0.8951757,  0.8951757, -1.4795]))
(1.9472029472402153, 'Ti', 'O', array([-0.8951757,  0.8951757,  1.4795]))
(1.9472029472402153, 'Ti', 'O', array([ 0.8951757, -0.8951757, -1.4795]))
(1.9472029472402153, 'Ti', 'O', array([ 0.8951757, -0.8951757,  1.4795]))
(1.9472029472402153, 'O', 'Ti', array([ 0.8951757,  0.8951757, -1.4795]))
(1.9472029472402153, 'O', 'Ti', array([ 0.8951757,  0.8951757,  1.4795]))
(1.9817718303429834, 'Ti', 'O', array([-1.4013243, -1.4013243,  0.       ]))
(1.9817718303429837, 'Ti', 'O', array([ 1.4013243,  1.4013243,  0.       ]))
(1.9817718303429837, 'O', 'Ti', array([-1.4013243, -1.4013243,  0.       ]))
```

OverlapCalculator

- calculate overlap of empirical atom radii
- other results: site square overlap, coordination numbers, coordination histograms, neighborhoods of touching sites, overlap gradients
- related: class *AtomRadiiTable* and its specializations *ConstantRadiiTable*, *CovalentRadiiTable*
  - radius lookup by atom symbol
  - support for custom atom radii

example:

```
>>> from diffpy.structure import loadStructure
>>> from diffpy.srreal.overlapcalculator import OverlapCalculator
>>> sto = loadStructure('SrTiO3.cif')
>>> oc = OverlapCalculator()
>>> oc.atomradiitable.fromString("Sr2+:1.44,  Ti4+:0.605,  O2-:1.35")

>>> oc(sto)
array([  4.89062513e-03,   1.67088000e-05,   1.63577798e-03,
         1.63577798e-03,   1.63577798e-03])
>>> oc.meansquareoverlap
0.0019629335719733893
>>> oc.coordinations
array([ 12.,   6.,   6.,   6.,   6.])
>>> oc.coordinationByTypes(4)
{'Ti4+': 2.0, 'Sr2+': 4.0}
```

BROOKHAVEN
NATIONAL LABORATORY

BVSCalculator

- bond valence sums – approximate formula for ion valences

Brese, Acta Cryst. B47, 192-197 (1991)

$$v_{ij} = \exp\left[\frac{R_{ij} - d_{ij}}{b}\right]$$
$$V_i = \sum_j v_{ij}$$

- evaluates valence at each site, BVS difference, mean square BVS difference which accounts for partial occupancies and site multiplicities
- related: class BVParametersTable
  - lookup of bond valence parameters, [bvparm2009.cif by I. D. Brown]
  - option to define and revert custom BVS parameters

example:

```
>>> from pyobjcryst import loadCrystal
>>> from diffpy.srreal.bvscalculator import BVSCalculator
>>> sto = loadCrystal('SrTiO3.cif')
>>> bvsc = BVSCalculator()
>>> bvsc(sto)
array([ 2.12652479,  4.16096701, -2.0958306 ])
>>> bvsc.bvdiff
array([-0.12652479, -0.16096701, -0.0958306 ])
>>> bvsc.bvmsdiff
0.013893882037591496
```

BROOKHAVEN NATIONAL LABORATORY

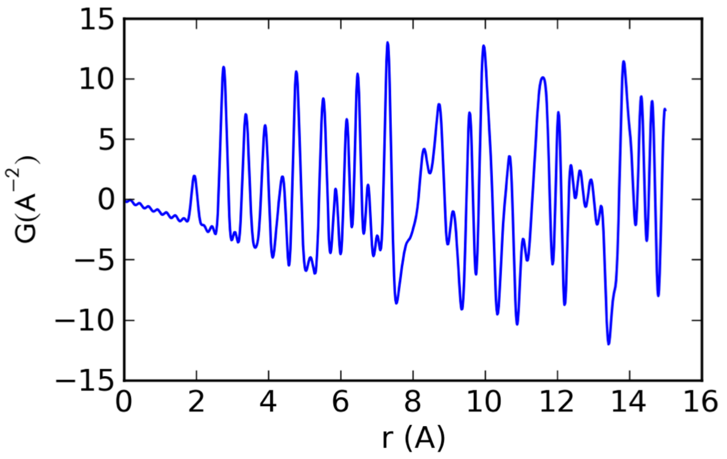
PDFCalculator

$$G(r) = \frac{1}{N r \langle b \rangle^2} \sum_{i \neq j} b_i b_j \, \delta(r - r_{ij}) \; - \; 4\pi r \rho_0$$

- PDF calculation in real-space
  - suitable for periodic systems
  - one structure per calculator → mixed-phase PDFs obtained by summing single-phase PDFs
- other results: radial distribution function, partial PDFs, *F*(*Q*)
- class ScatteringFactorTable
  - lookup of xray, netron or electron scattering factors
  - support for custom scattering factors
- class PeakProfile – the profile function for a pair contribution
- class PeakWidthModel – calculates profile width for a given atom pair
- class PDFEnvelope – one or more r-dependent scaling envelopes
- class PDFBaseline – the baseline function, by default $-4\pi\rho_0 r$

example:

```
>>> from diffpy.Structure import Structure
>>> from diffpy.srreal.pdfcalculator import PDFCalculator
>>> sto = Structure(filename='SrTiO3.cif')
>>> pdfc = PDFCalculator(rmax=15, qmax=25)
>>> r, g = pdfc(sto)
>>> import pylab
>>> pylab.plot(r, g)
```



DebyePDFCalculator

- PDF calculation in *Q*-space – *F*(*Q*) calculated by Debye scattering equation and Fourier transformed to *G*(*r*)

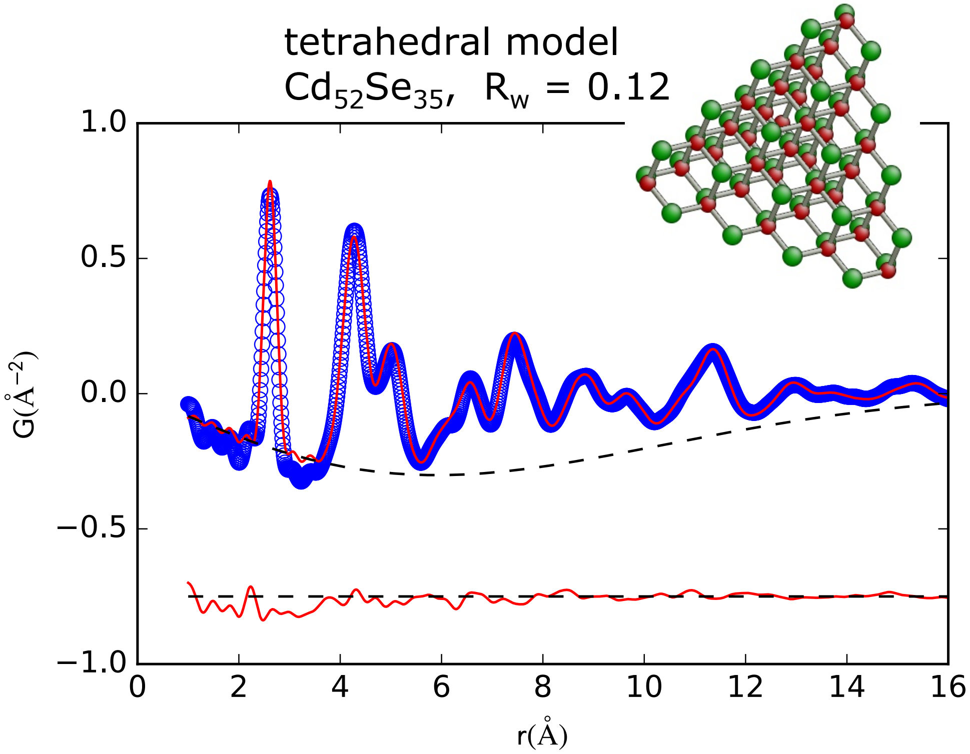
$$F(Q) = \frac{1}{N\langle f(Q)\rangle^2} \sum_{i,j} f_i(Q) f_j(Q) \frac{\sin Q r_{ij}}{r_{ij}} \; \exp\left[-\frac{1}{2}\,\sigma_{ij}^2\, Q^2\right] \qquad G(r) = \frac{2}{\pi} \int_{Q_{\mathrm{min}}}^{Q_{\mathrm{max}}} F(Q) \sin Qr \, \mathrm{d}Q$$

- suitable for molecules or nano-clusters
- PDF baseline simulated by Q_{min} cutoff in the calculated *S*(*Q*)

example:

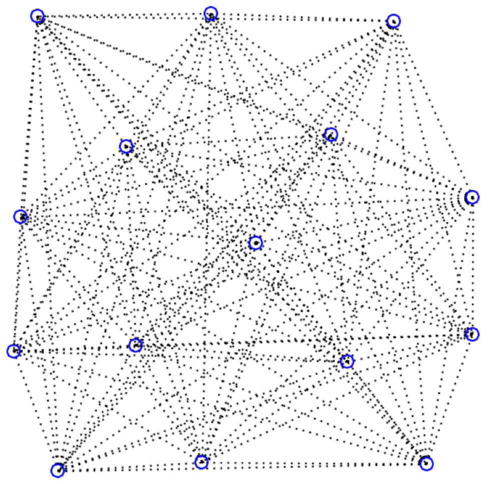
- structure refinement of CdSe quantum dots
- simulated PDF reflects particle shape in amplitude dampening and baseline shape

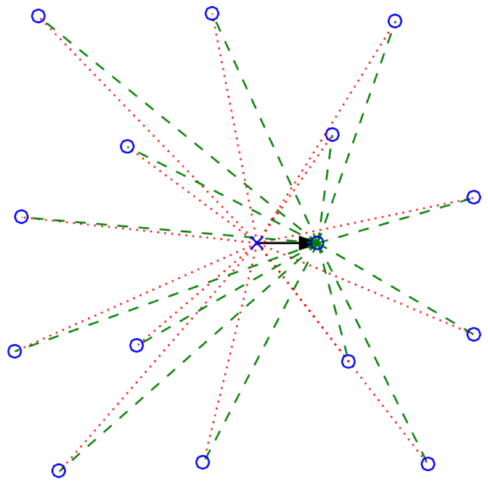
A. Beecher, J. Am. Chem. Soc., 2014, 136 (30), 10645–10653



Optimized PDF evaluation

- PDF is calculated from a sum of N^2 pair contributions → it is computationally expensive for larger models

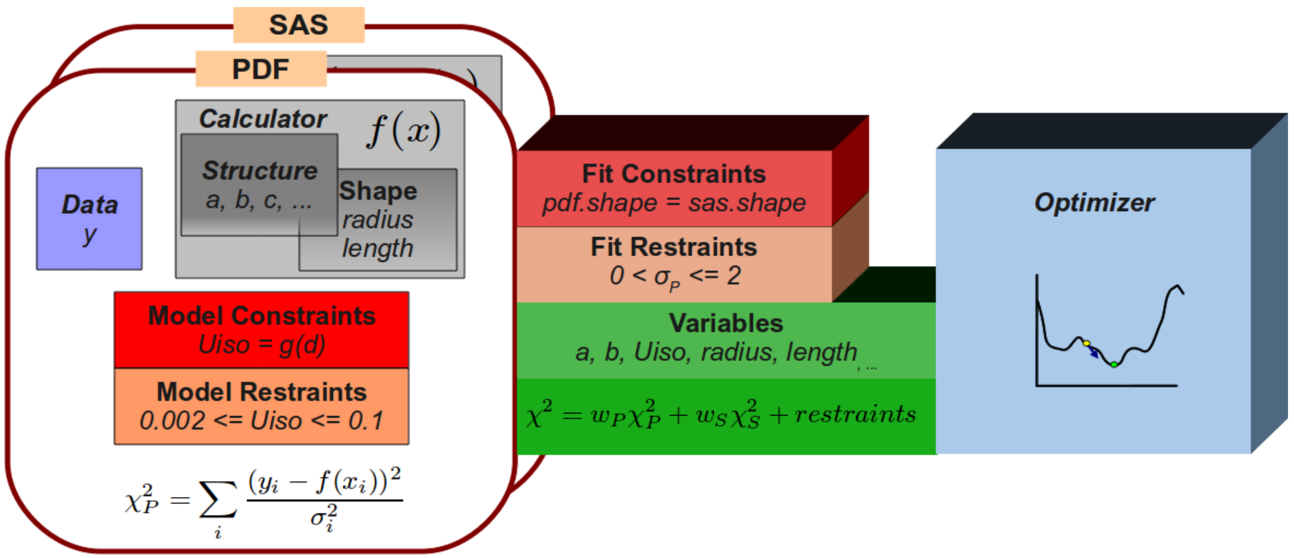




- real-space and Debye PDF calculators support optimized PDF calculation.
- when few atoms change, their old contributions are subtracted and new ones added to the PDF. Computational cost is reduced from $N^2 \rightarrow N$.
- optimized PDF calculation is about 1000 faster for one-atom updates in 10,000-atom structure.

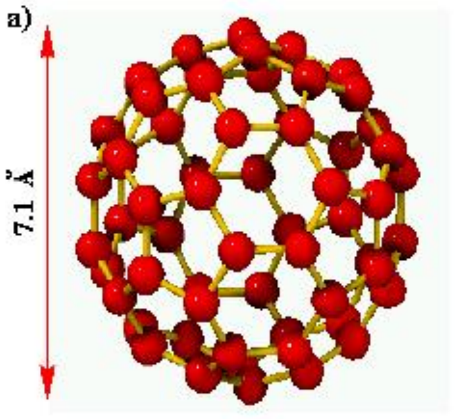
SrFit – multi-component fit manager

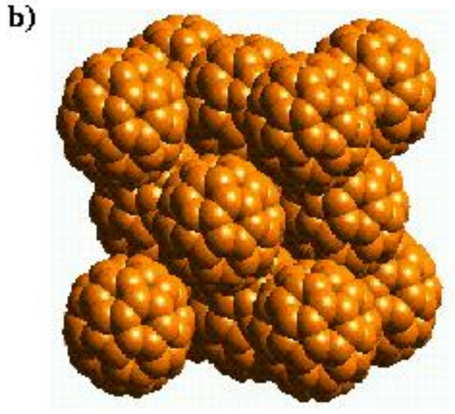
- Python module for general multi-component data refinement
- construct FitContribution by associating observed data with simulation
  - models can be defined with built-in calculators, math expressions, Python functions
  - model parameters are exposed to SrFit. Parameters can be constrained or restrained, e.g., "*a* = *b* = *c*" for cubic structure
- FitContributions are combined to a single total cost function (residual vector or scalar value) with interface suitable for optimization routines
- control functions to fix/free variables, define constraints, restraints, hook functions
- post-processing to generate fit result reports – partial costs per each contribution, error estimates and correlations of the fit variables.

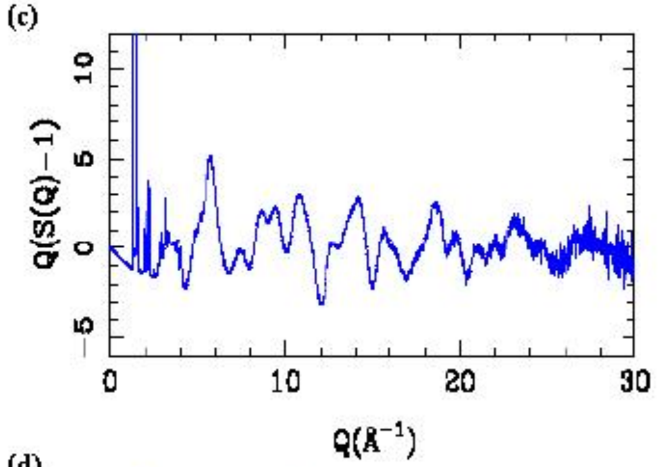


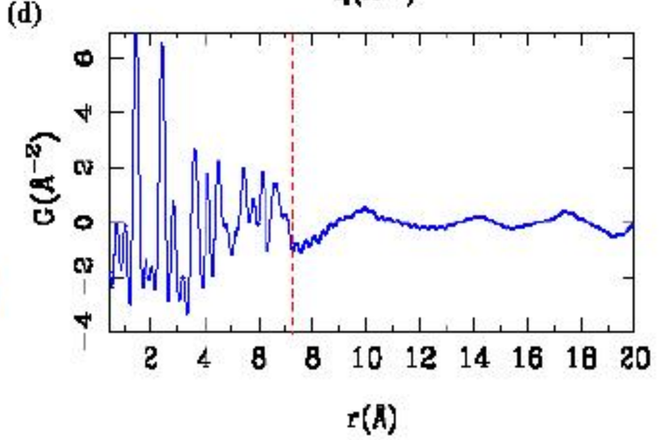
BROOKHAVEN NATIONAL LABORATORY

PDF modeling of fcc-C_{60}







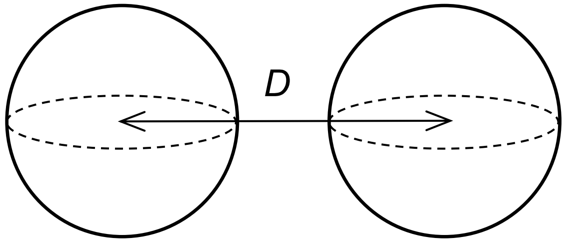


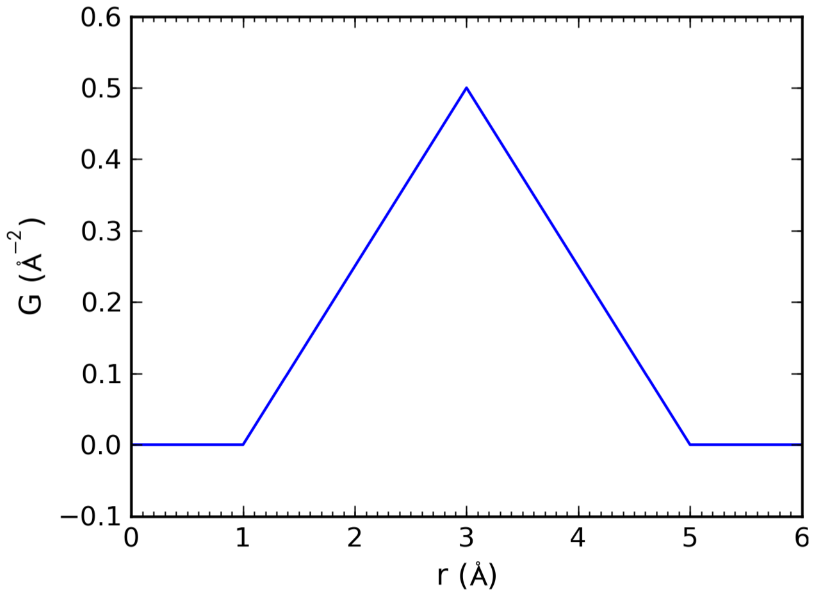
- neutron PDF measured on C_{60} fcc structure [GLAD IPNS, E. Bozin]
- low-*r* sharp peaks – correlations within C_{60}
- high-*r* broad peaks – correlations between randomly oriented balls

Can we simulate PDF on a full measured range?

- calculate as a sum of single particle PDF and PDF from a lattice of spherical shells

PDF peak profile for spherical shells





- PDF of two spherical shells can be calculated analytically

$$G(r) = \frac{1}{S_1 S_2 r} \iint\limits_{S_1 S_2} \delta(r - r_{12})\, \mathrm{d}S_1 \mathrm{d}S_2$$

triangular profile centered at spheres' distance *D*

- cluster of spherical shells → PDF calculation requires <u>triangular profile function</u>
- non-standard PDF profile requires
  - definition of the profile function
  - select it for a PDFCalculator instance

BROOKHAVEN NATIONAL LABORATORY

Custom PDF peak profile

profile defined in C++

```
#include <cmath>
#include <diffpy/srreal/PeakProfile.hpp>

using diffpy::srreal::PeakProfile;
using diffpy::srreal::PeakProfilePtr;

class SphericalShellsProfile : public PeakProfile {
public:
    PeakProfilePtr create() const {
        return PeakProfilePtr(new SphericalShellsProfile());
    }

    PeakProfilePtr clone() const {
        return PeakProfilePtr(new SphericalShellsProfile(*this));
    }

    const std::string& type() const {
        static std::string tp = "sphericalshells-cpp";
        return tp;
    }

    double yvalue(double x, double fwhm) const
    {
        if (fabs(x) > fwhm)  return 0.0;
        double rv = (fwhm - fabs(x)) / (1.0 * fwhm * fwhm);
        return rv;
    }

    double xboundlo(double fwhm) const   { return -fwhm; }

    double xboundhi(double fwhm) const   { return +fwhm; }
};

bool reg_SawToothProfile = SphericalShellsProfile().registerThisType();
```

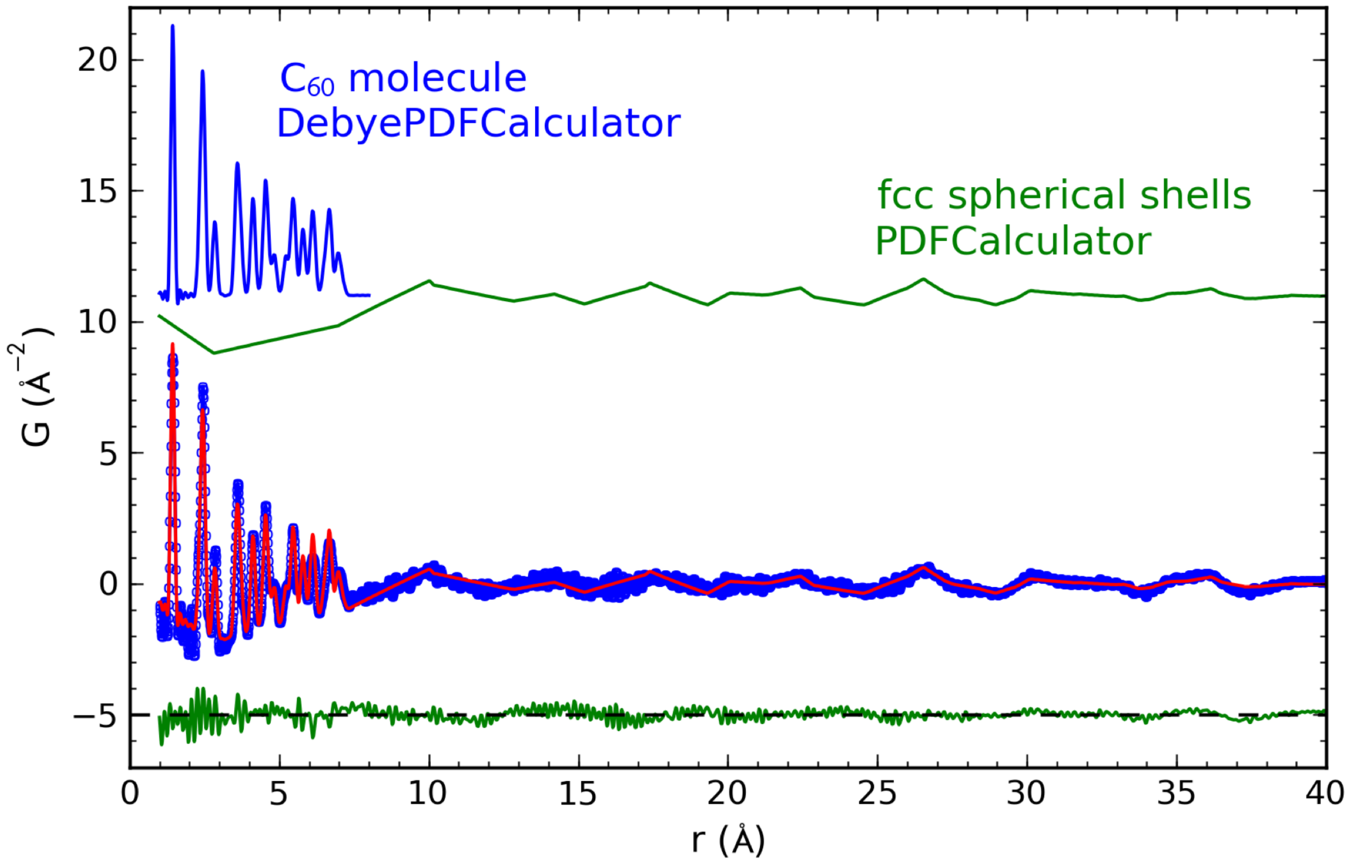
profile used in Python

```
>>> from diffpy.srreal.pdfcalculator import PeakProfile, PDFCalculator
>>> PeakProfile.getRegisteredTypes()
set(['croppedgaussian', 'gaussian'])
>>> import ctypes
>>> ctypes.cdll.LoadLibrary('./sphericalshells-cpp.so')
>>> PeakProfile.getRegisteredTypes()
set(['sphericalshells-cpp', 'croppedgaussian', 'gaussian'])
>>> pdfcalc = PDFCalculator()
>>> pdfcalc.setPeakProfileByType('sphericalshells-cpp')
```

- new profile functions can be defined either in Python or C++
- for C++ the profile function is compiled as a dynamic link library sphericalshells-cpp.so
- on loading the library adds new profile to the global registry → profile ready for use in Python
- no need to compile any other C++ sources related to PDFCalculator
- no need to write any Python wrappers for the new profile function

BROOKHAVEN NATIONAL LABORATORY

PDF refinement of fcc C_{60}



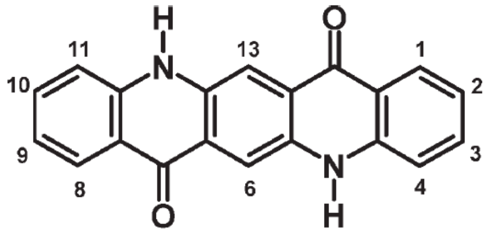
fit residuum R_w = 0.26 scale ratio = 60.0(1) U_{iso} = 0.00323(4)

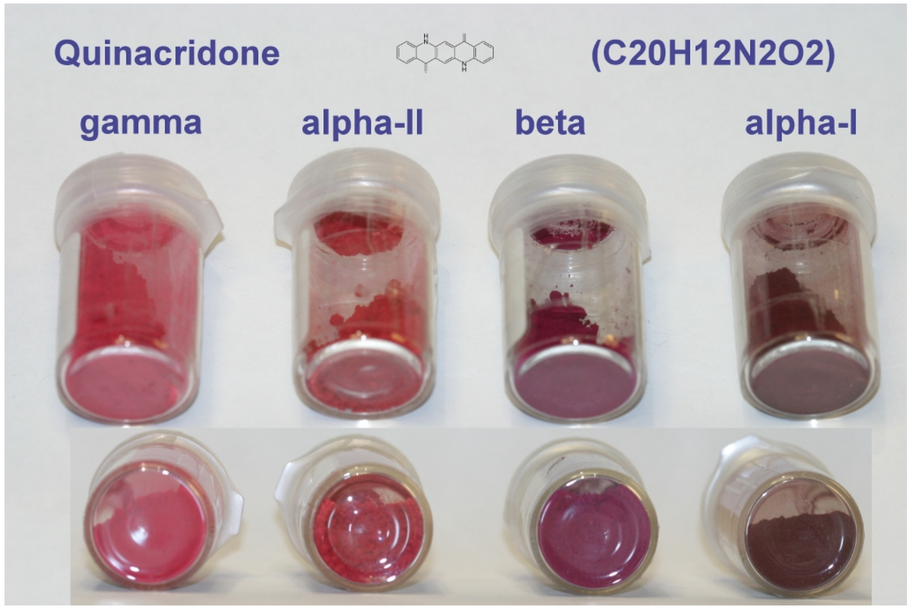
molecule diameter = 7.113(2) shell diameter = 7.22(4)

- PDF from fcc C_{60} can be refined on the full measured range accounting for both intra and inter-molecular correlations

BROOKHAVEN NATIONAL LABORATORY

PDF analysis of organic crystals



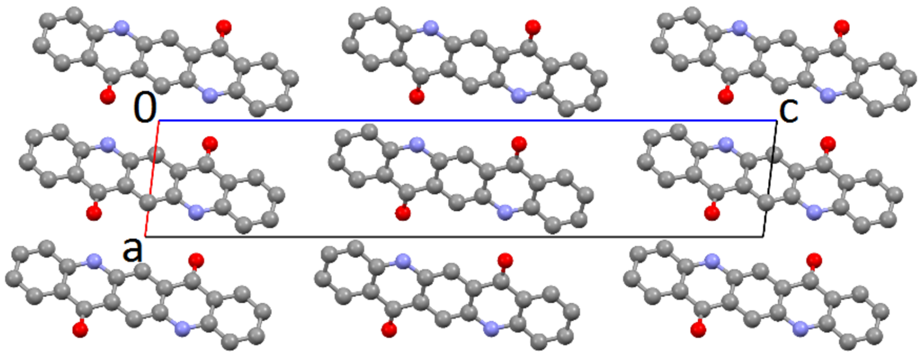


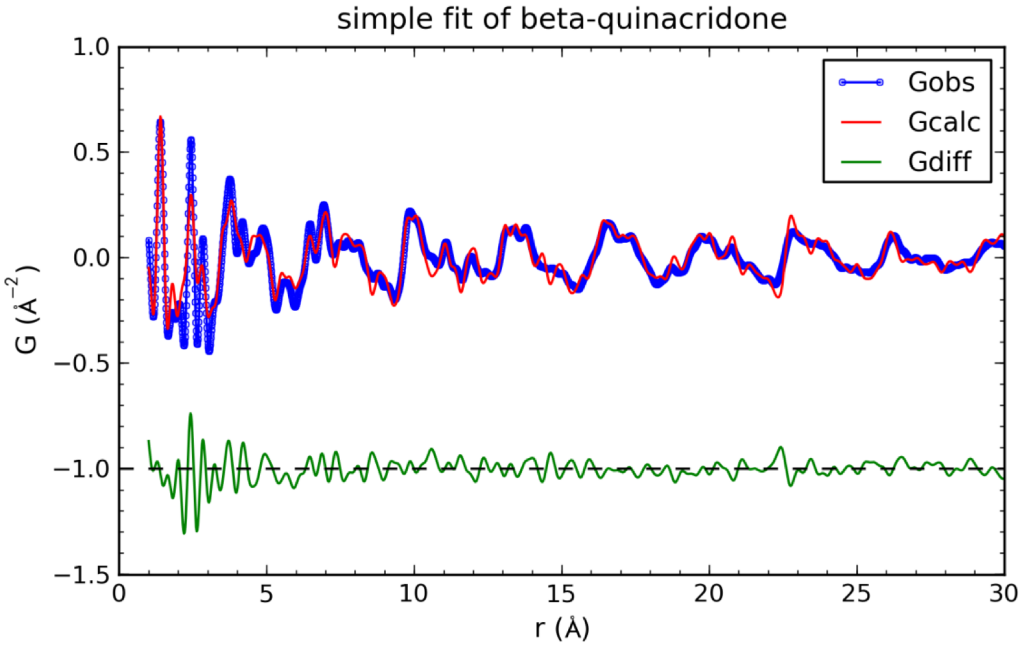
quinacridone – $C_{20}H_{12}N_2O_2$

- industrially important pigments, red and violet paints
- can form many phases, some (alpha-II) do not crystallize and have unknown structure
- experimental PDFs measured at APS ANL beamline 11-ID-B and NSLS BNL beamline X17A
- standard refinement with PDFgui is of poor quality even for the known β-phase

collaboration with Prof. Martin U. Schmidt and
Dr. Dragica Prill, Goethe Universität, Frankfurt am Main

PDF modeling of β-quinacridone



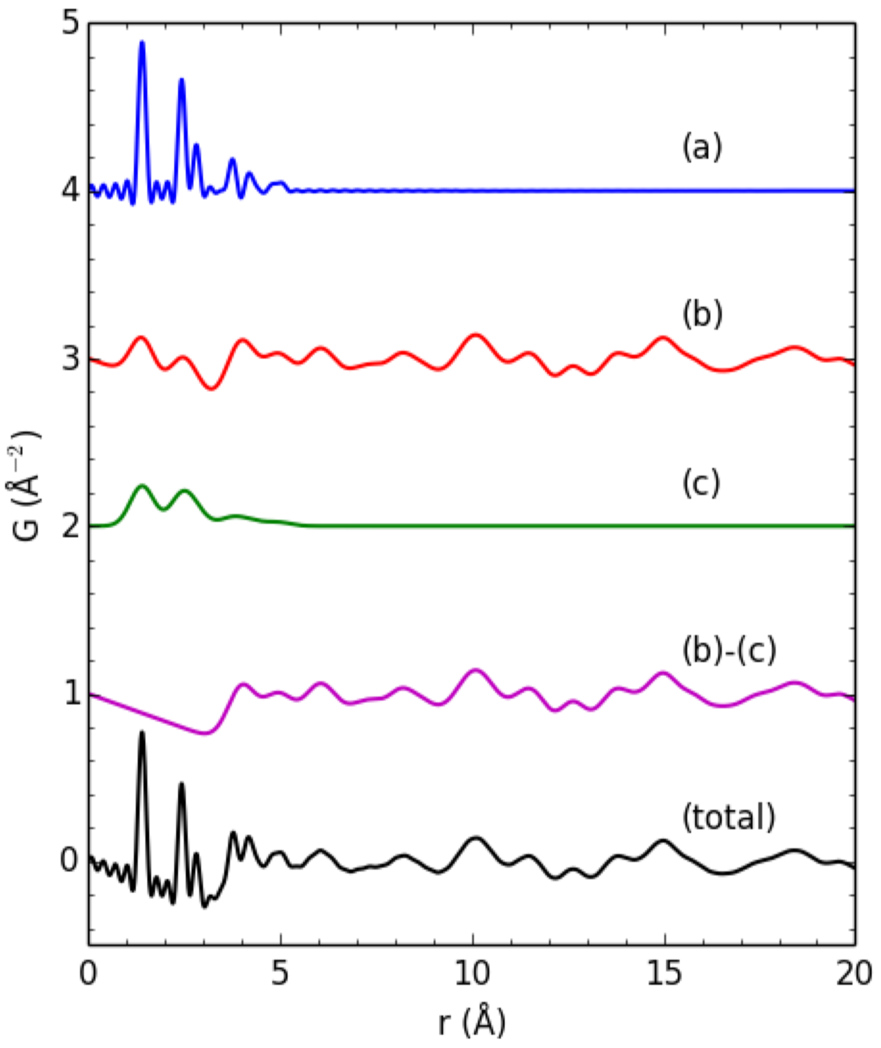


- monoclinic $P2_1/c$, 2 molecules per unit cell
- refinement in PDFgui gives poor $\boldsymbol{R_w = 0.41}$
  - low-*r* peaks too wide
  - high-*r* peaks too sharp
- simple model assumes indepen-dent isotropic thermal vibrations
- peak widths depend strongly on *r*
  - sharp peaks for intra-molecular atom pairs
  - broad peaks for inter-molecular correlations
- PDF model has to use different displacement factors for pairs in the same molecule and inter-molecular pairs
- molecule must not deform when cell parameters change

D. Prill, *J. Appl. Cryst.* **48**, 171-178 (2015).

BROOKHAVEN NATIONAL LABORATORY

PDF modeling of β-quinacridone



PDF calculation with separate intra- and inter-molecular contributions

(a) PDF from a single-molecule, small atom displacements U_{intra}

(b) PDF from a crystal with large displacements U_{inter}

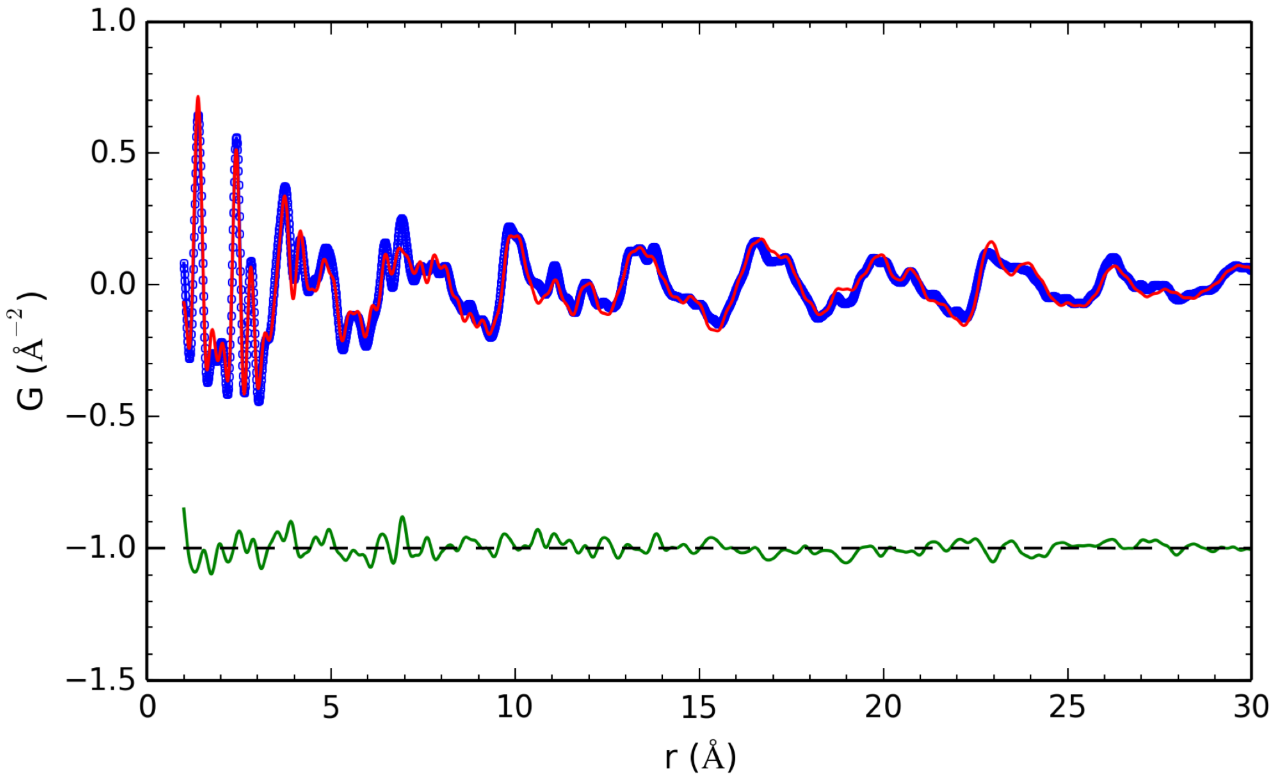
(c) PDF from a molecule with large displacements U_{inter}

(b) – (c) PDF from inter-molecular interactions only

(a) + (b) – (c) total PDF reflecting both displacements U_{intra}, U_{inter}

BROOKHAVEN NATIONAL LABORATORY

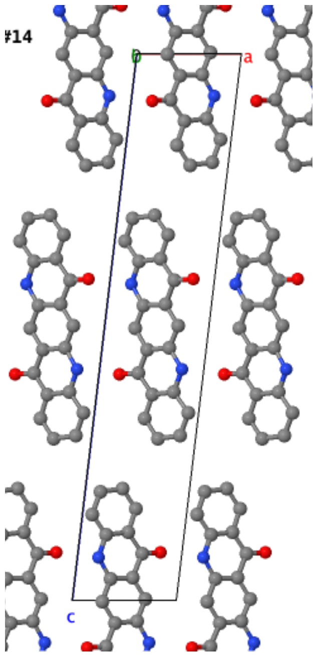
PDF modeling of β-quinacridone

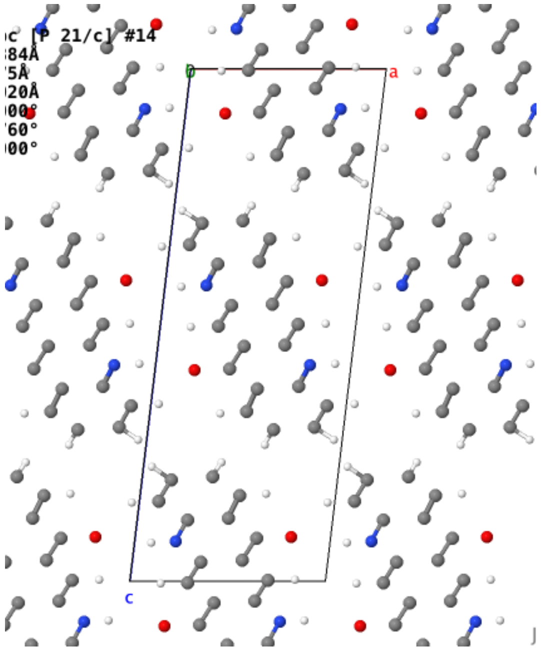


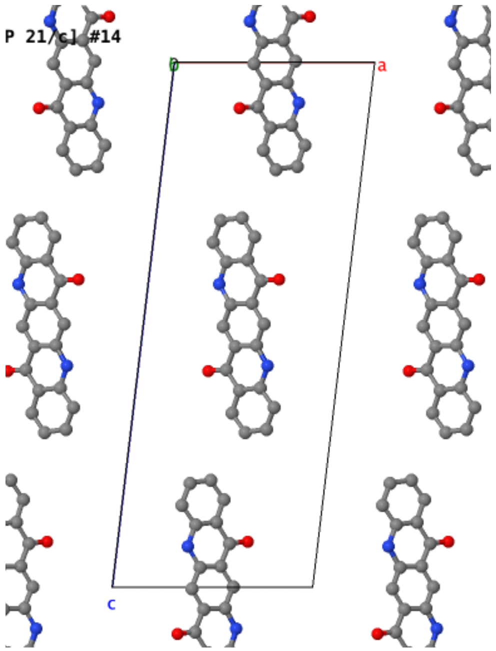
- refined unit cell, data scale and displacement factors
  U_{inter} = 0.0014(2) Å^2 U_{intra} = 0.023(2) Å^2
- significant fit improvement R_w = 0.41 → $\boldsymbol{R_w}$ **= 0.28**
- remnant fit difference due to anisotropic molecule displacements, displacement anisotropy can be studied with improved models

BROOKHAVEN
NATIONAL LABORATORY

Structure representation with rigid molecules



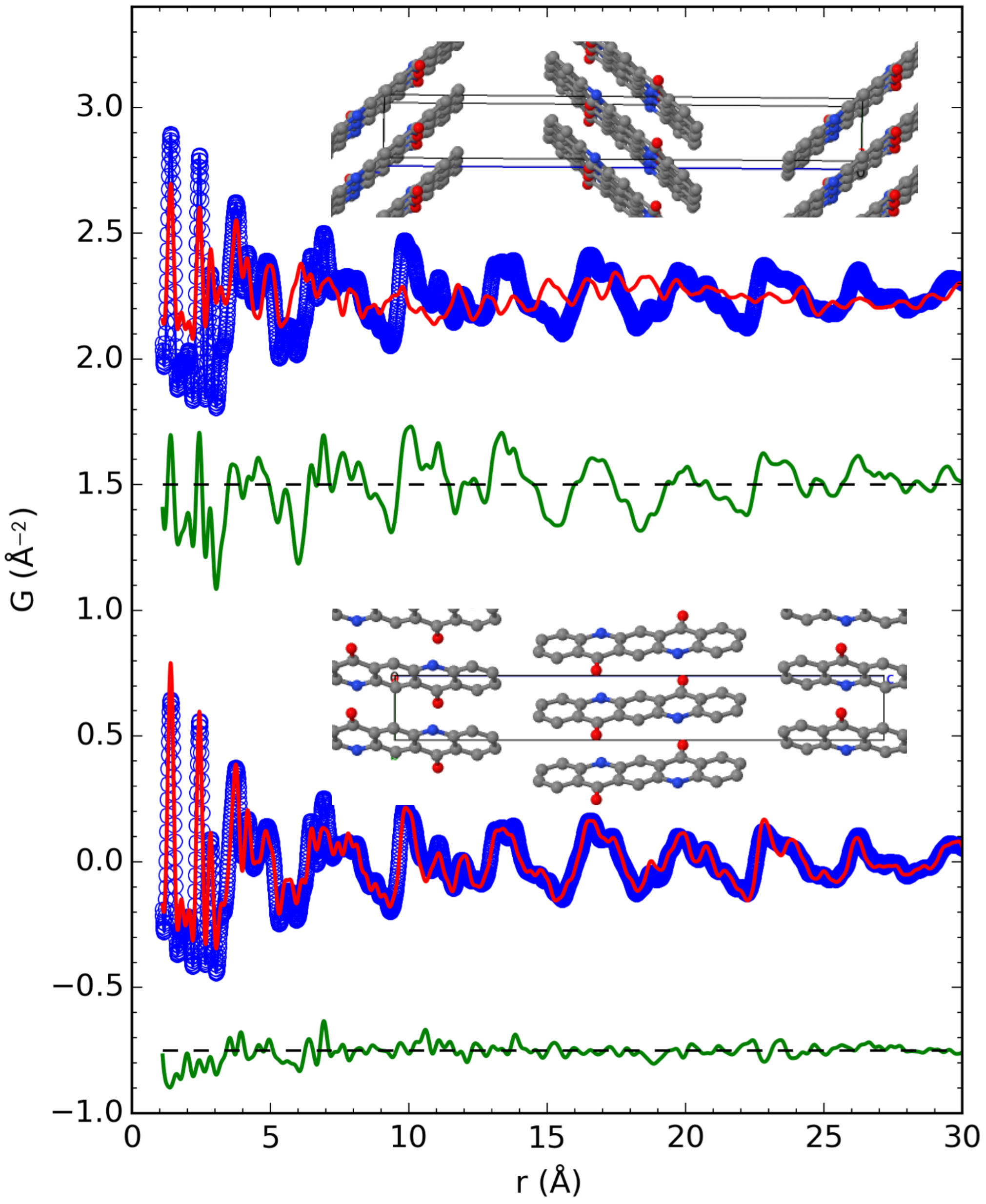




- structure representation needs to handle molecules as a rigid unit within crystal lattice → pyobjcryst in DiffPy-CMI, an interface to ObjCryst++
- molecule retains its shape when cell parameters change
- molecule placement is defined by its center-of-mass and orientation quaternion

V. Favre-Nicolin, J. Appl. Cryst. 35, 734-743 (2002)

Solution of molecule orientation in β-quinacridone

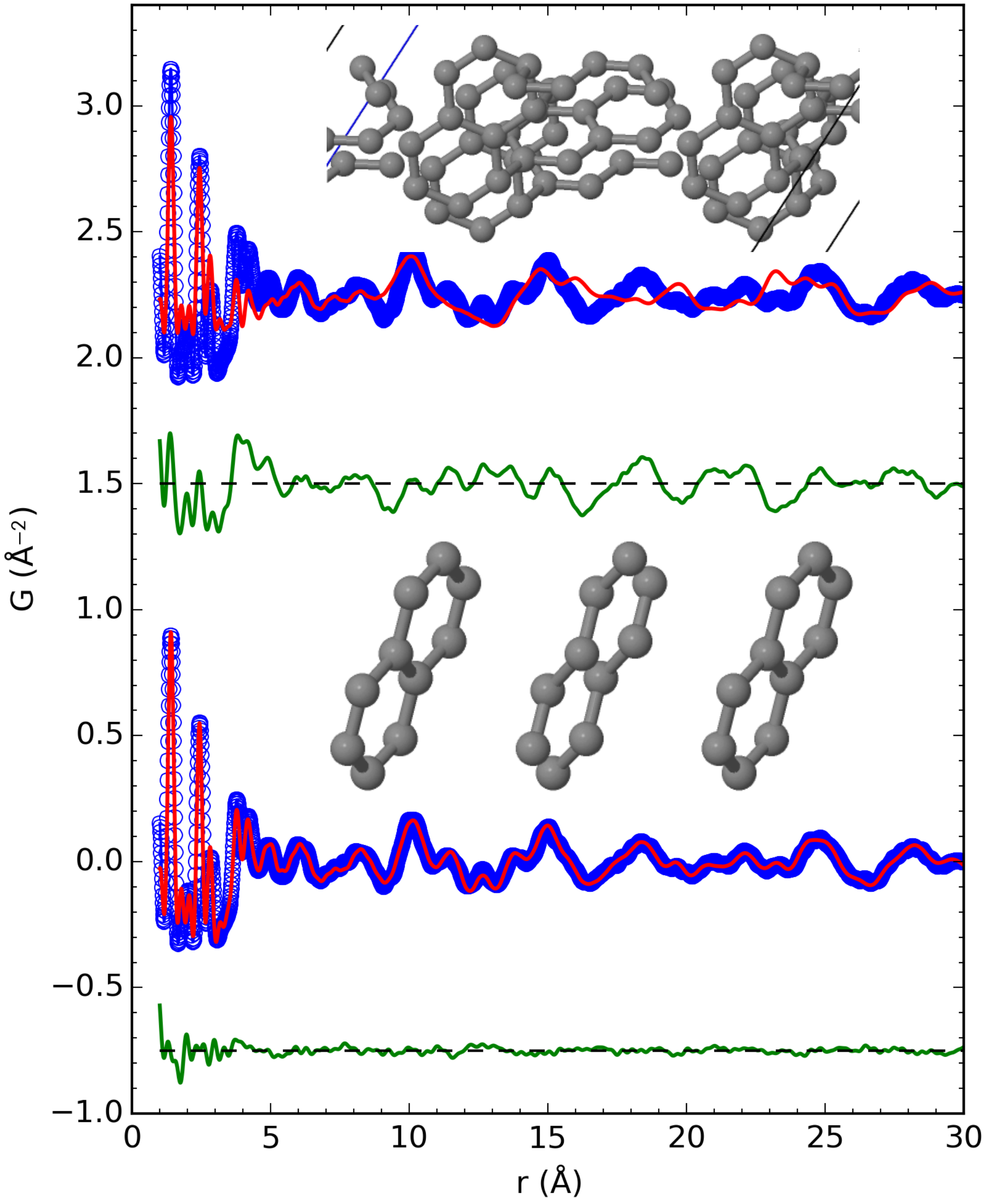


- apply random rotation on the quinacridone molecule in asymmetric unit
- preserve $P2_1/c$ symmetry
  - molecule position fixed at inversion center
  - use symmetric rotation for the second molecule in the UC
- refine molecule orientation, cell parameters, ADPs: $R_w = 0.83 \rightarrow R_w = 0.27$
- optimized structure converged to β-quinacridone

D. Prill *et al.*, Acta Crystallogr. A 72, 62-72 (2016)

BROOKHAVEN NATIONAL LABORATORY

Solving molecule placement in naphthalene



- structure solution in P1
- 2 molecules in the unit cell with independent positions and orientations, 9 DOF
- start from a random initial placement of the UC molecules
- refine orientations, position, 6 cell parameters, ADPs: $R_w = 0.61 \rightarrow R_w = 0.16$
- optimized structure converged to $P2_1/a$ naphthalene

D. Prill *et al.*, Acta Crystallogr. A 72, 62-72 (2016)

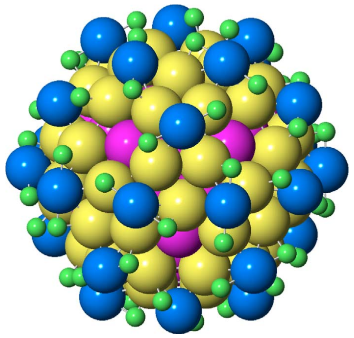
BROOKHAVEN NATIONAL LABORATORY

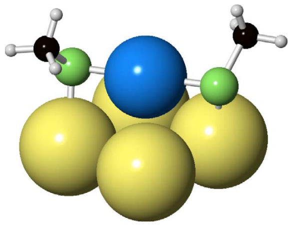
Structure solutions of molecular crystals

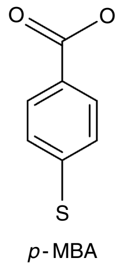
sample	symmetry	molecules	DOF	success rate
β-quinacridone	$P2_1/c$	1	3	6.4% (14/220)
naphthalene	$P2_1/a$	1	3	33% (79/240)
naphthalene	P1	2	9	2.5% (7/280)
allopurinol	$P2_1/c$	1	3	3.6% (12/329)
allopurinol	P1	4	21	0.4% (2/534)

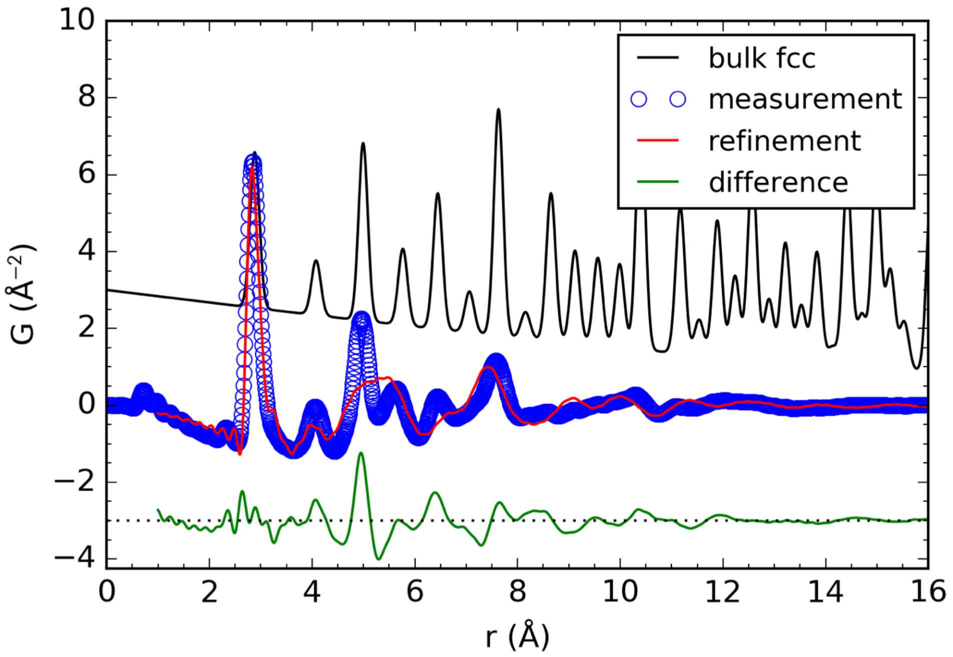
- structure determination of molecular crystals
  - start from random initial placement of molecule(s) in the unit cell
  - fit the PDF, optimize rotation, position, cell parameters, intra- and inter-molecular ADPs
  - repeat to search for best convergence
- correct solutions were found also at lowered symmetry and enlarged DOF for molecular placement

Polymorphism in $Au_{144}(SR)_{60}$







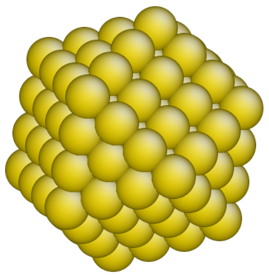


- $Au_{144}(SR)_{60}$ is ultra-stable (magic size), easy to prepare nanocluster system
- does not crystallize, published NMR+DFT studies claim icosahedral structure [Bahena *et al.*, J. Phys. Chem. Lett., 4, 975-981 (2013)
- $Au_{144}(p\text{-MBA})_{60}$ produced by Prof. Ackerson group, Colorado State Univ., Fort Collins.
- X-ray PDF from $Au_{144}(p\text{-MBA})_{60}$ is dramatically different and inconsistent with icosahedral model.
- measured peaks line-up with prominent peaks in bulk Au-*fcc* → actual structure should contain close-packed motifs.

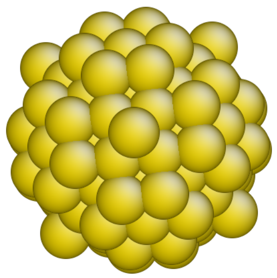
K. Jensen and P. Juhas *et al*, Nat. Commun., 7:11859 (2016)

BROOKHAVEN NATIONAL LABORATORY

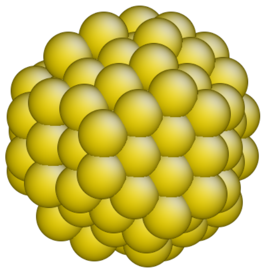
Polymorphism in $Au_{144}(SR)_{60}$



Au_{147} cuboctahedron

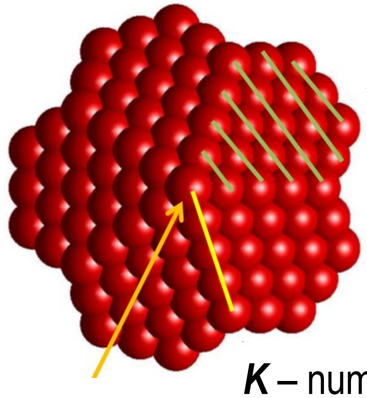


Au_{141} *fcc*



Au_{147} *hcp*

Marks decahedron (*N, M, K, T*)



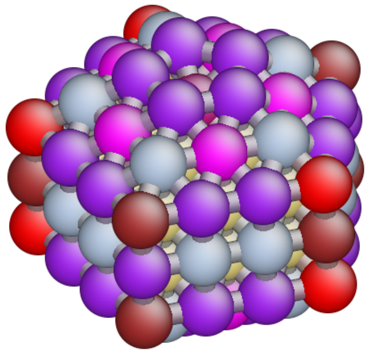
M – number of {002} shells

T – number of planes truncated from the top and bottom of decahedron

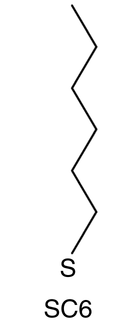
K – number of columns along the twin boundary

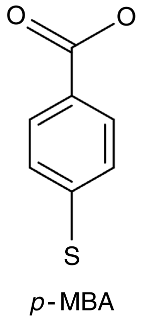
N – atoms in the central column

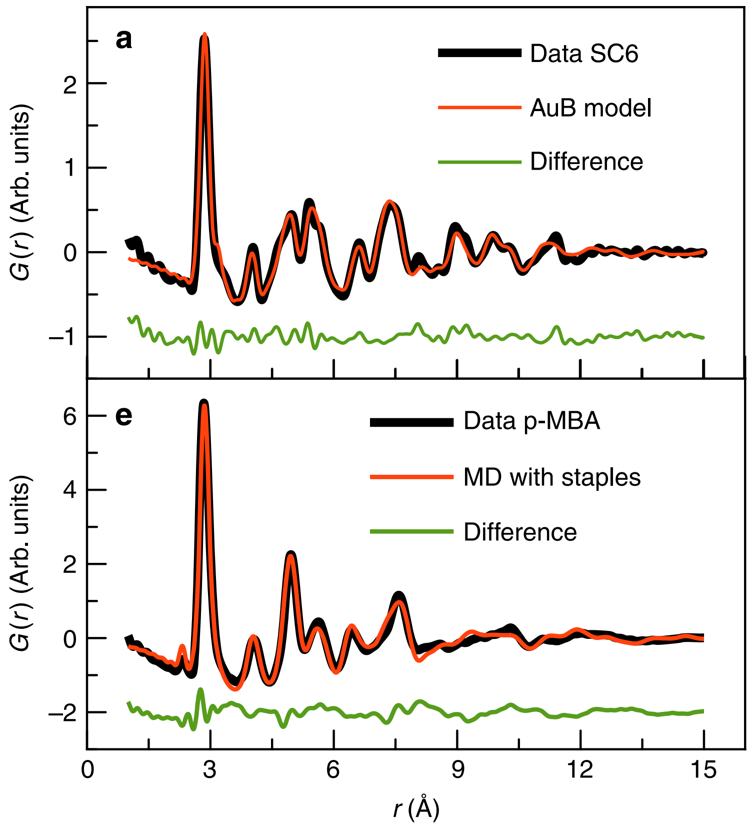
- DiffPy-CMI used for PDF refinement of gold clusters (expansion ratio, isotropic displacements for core and surface atoms)
- PDF fit evaluated for simple closed packed structures and a series of generated Marks decahedra (MD)
- very good PDF fit for 144-atom MD6441. Close PDF fit also after stripping to 114-atom MD6341, which has room for 30 S-Au-S surface “staples”.

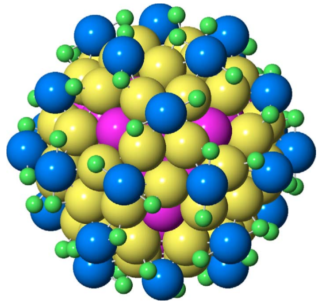


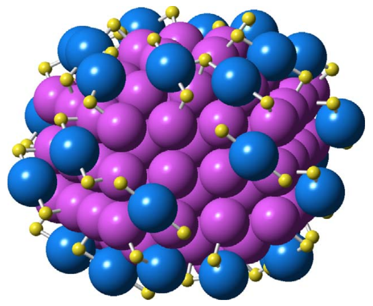
Polymorphism in $Au_{144}(SR)_{60}$











- PDF has not enough resolution to determine surface structure, i.e., staples-ligand placement. The 114-atom MD core is required for good fit.
- follow-up PDF measurements on $Au_{144}(SR)_{60}$ samples produced with different ligands showed the previous icosahedral phase and a mixture of icosahedral and MD phases → polymorphism happens also for nanoclusters.

K. Jensen and P. Juhas *et al*, Nat. Commun., 7:11859 (2016)

MPDF – magnetic Pair Distribution Function

- developed by Benjamin Frandsen, Columbia University (now Brigham Young University) [*Acta Crystallogr. A* **70**, 3-11 (2014)]
- **magpdf** – extension to DiffPy-CMI for simulation and refinement of magnetic PDFs
- magpdf can be obtained from **https://github.com/benfrandsen/diffpy.magpdf**

How can we analyze diffuse magnetic scattering?

$$\begin{tikzcd}
\text{Diffuse scattering from } \textbf{atoms} \text{ and } \textbf{nuclei} \arrow[r, "\text{Fourier transform}"] & \textbf{Atomic PDF} \\
\text{Diffuse scattering from } \textbf{magnetic moments} \arrow[r, "\text{Fourier transform}"] & \textbf{Magnetic PDF}
\end{tikzcd}$$

Expression for the magnetic PDF

Magnetic PDF is defined analogously to the atomic PDF:

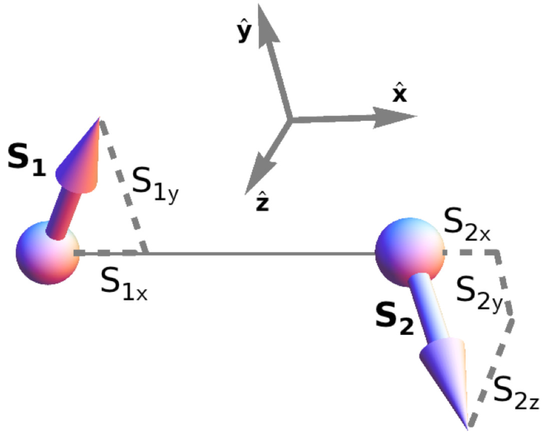
$$f(r) = \frac{2}{\pi}\int_0^\infty Q\left(S(Q) - 1\right)\sin Qr\,\mathrm{d}Q$$

where $S(Q)$ is the *magnetic* structure function, given by

$$S(Q) = 1 + \frac{1}{N}\frac{3}{2S(S+1)}\sum_{i\neq j}\left[A_{ij}\frac{\sin Qr_{ij}}{Qr_{ij}} + B_{ij}\left(\frac{\sin Qr_{ij}}{(Qr_{ij})^3} - \frac{\cos Qr_{ij}}{(Qr_{ij})^2}\right)\right]$$

$$A_{ij} = S_i^y S_j^y$$

$$B_{ij} = 2S_i^x S_j^x - S_i^y S_j^y$$



This results in the following expression for the mPDF:

$$f(r) = \frac{1}{N}\frac{3}{2S(S+1)}\sum_{i\neq j}\left(\frac{A_{ij}}{r}\delta(r - r_{ij}) + B_{ij}\frac{r}{r_{ij}^3}\Theta(r_{ij} - r)\right)$$

HAVEN
LABORATORY

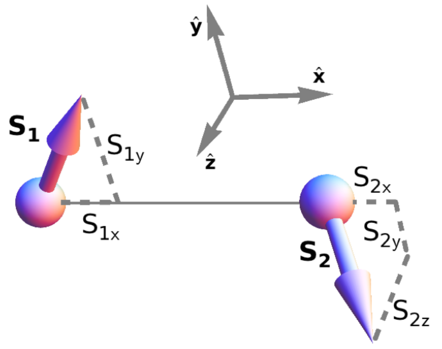
Simple examples of the mPDF

ATOMIC $$f(r) = \frac{1}{N\langle b\rangle^2}\sum_{i\neq j}\frac{b_i b_j}{r}\delta(r-r_{ij})$$

MAGNETIC $$f(r) = \frac{1}{N}\frac{3}{2S(S+1)}\sum_{i\neq j}\left(\frac{A_{ij}}{r}\delta(r-r_{ij}) + B_{ij}\frac{r}{r_{ij}^3}\Theta(r_{ij}-r)\right)$$

$$A_{ij} = S_i^y S_j^y$$

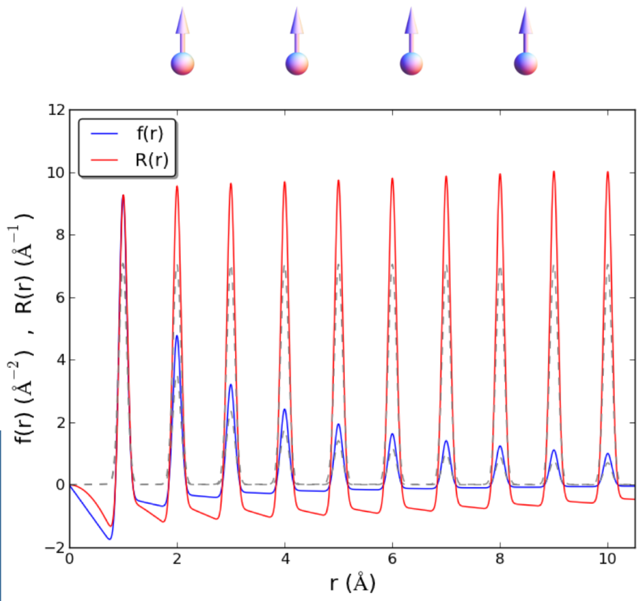
$$B_{ij} = 2S_i^x S_j^x - S_i^y S_j^y$$

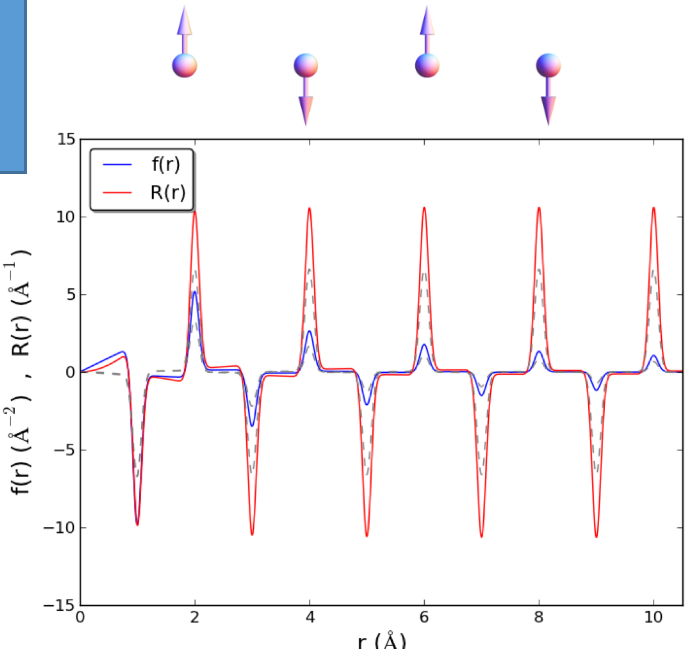


Positive peaks for *ferromagnetic* correlations

Negative peaks for *antiferromagnetic* correlations

Frandsen et al, *Acta A* **70**, 3-11 (2014)





What we get in a standard atomic PDF measurement

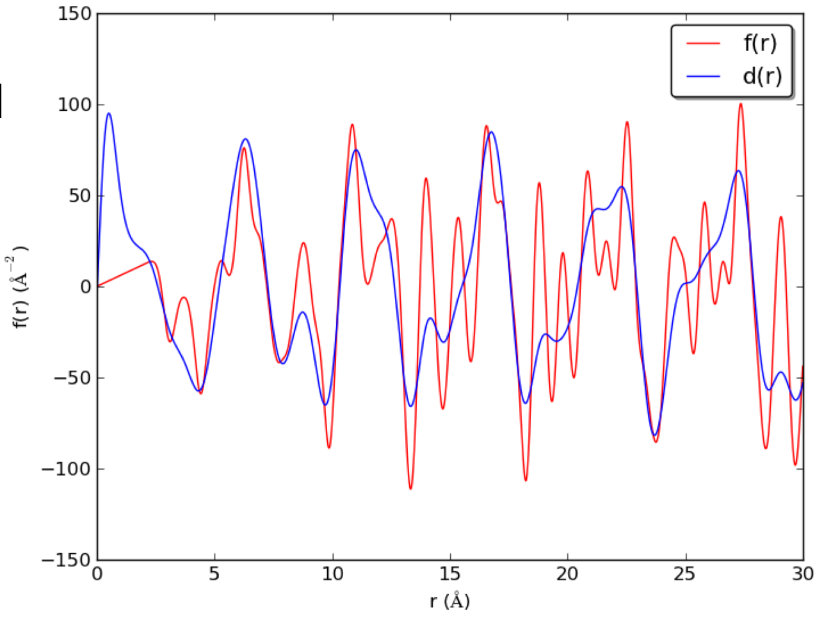
In a typical neutron PDF measurement, the **magnetic and nuclear scattering are mixed together.** Standard data reduction routines normalized by the average squared nuclear scattering length, but not the magnetic form factor. As a result, the **mPDF obtained from the standard procedures is a *broadened form* of the true expression:**

$$G_{\text{mag}} \approx f(r) * S(r)$$

Proper mPDF

Convolution

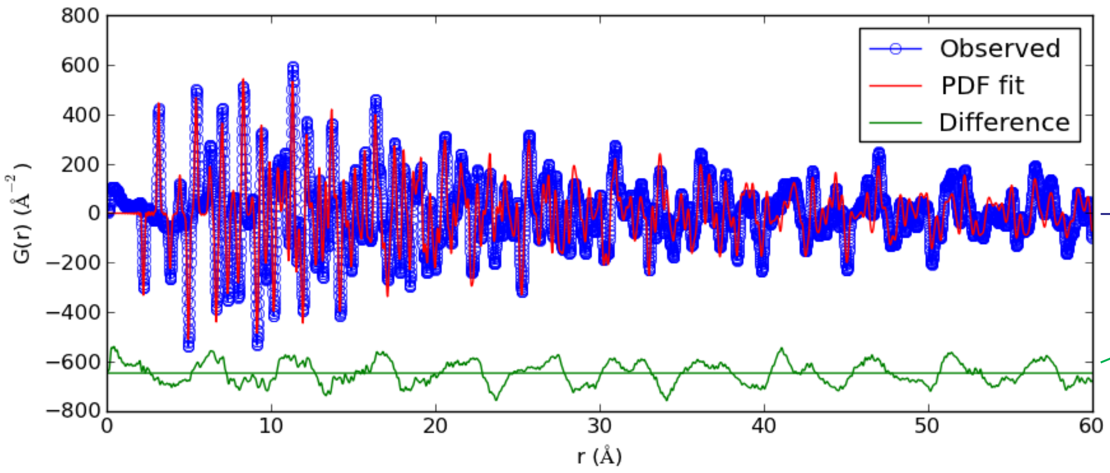
Approximately $\sqrt{2}\times$ the real-space extent of the magnetic moment



Red = proper mPDF
Blue = broadened mPDF

BROOKHAVEN NATIONAL LABORATORY

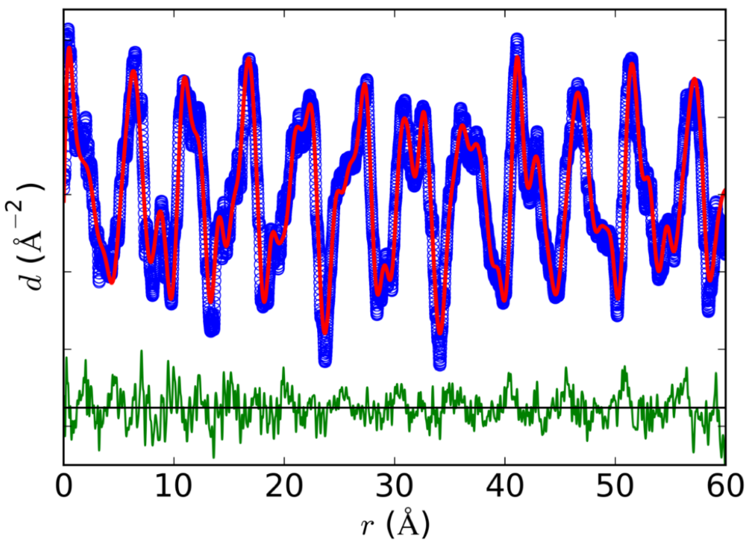
Example: Antiferromagnetic MnO



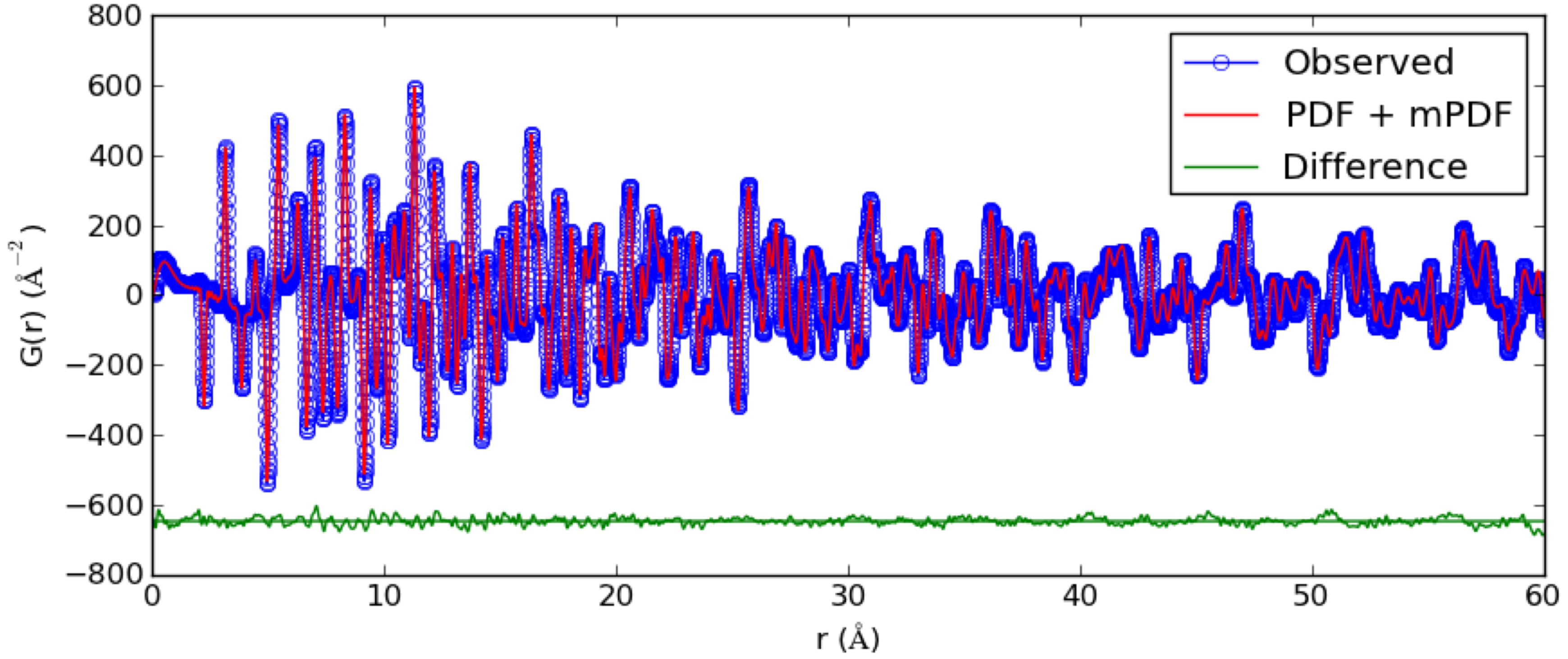
Fit includes only the atomic PDF.

Fit residual contains the mPDF.

Known antiferromagnetic structure of MnO provides excellent fit to the mPDF.

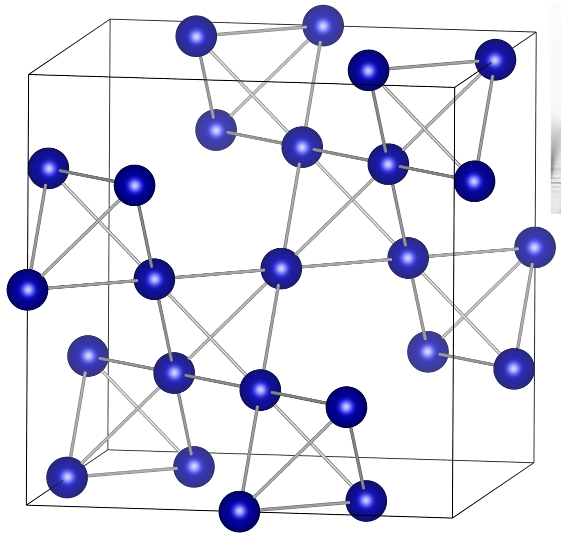


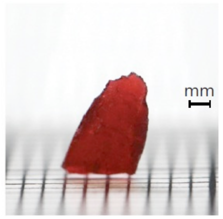
Example: Antiferromagnetic MnO

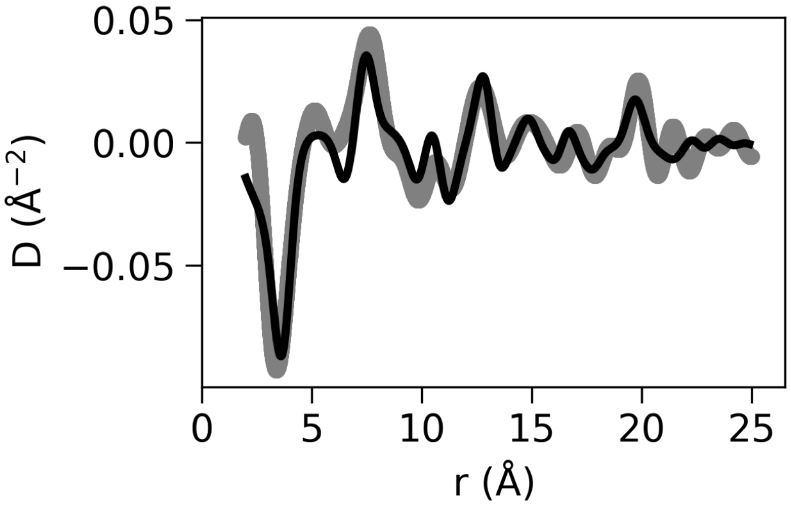


Total PDF fit (atomic + magnetic)

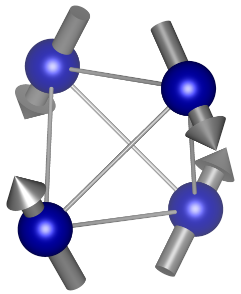
Exotic example: New frustrated pyrochlore system $NaCaCo_2F_7$







mPDF analysis reveals a non-trivial, non-collinear magnetic configuration with a correlation length of ≈2 nm.



Summary

- DiffPy-CMI – software framework for complex modeling, structure representations and manipulations, calculators for PDF, BVS, SAS, multi-component fit management, add-on for magnetic PDF analysis
- **resources:** https://www.diffpy.org – project webpage
  diffpy-users Google Group – users mailing list
- **installation on Linux or Mac**
  - download Anaconda Python – https://anaconda.com
  - install diffpy-cmi from "conda" package manager

```
conda install –c diffpy diffpy–cmi
```

- **installation on Windows** – requires Linux Virtual Machine

DiffPy-CMI tutorial

https://github.com/diffpy/add2019-diffpy-cmi

- **what to expect:**
  - IPython notebooks that proceed from basic functions to refinements not possible in PDFgui.
- **goals:**
  - learn about the main components of the framework and understand how they work together to define refinements.
  - perform structure manipulations and simple calculations
  - learn what structure analysis problems are suitable for DiffPy-CMI