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

P. Juhas (presented by S.J.L. Billinge)

Department of Applied Physics and Applied Mathematics

Columbia University,

CMPMS, Brookhaven National Laboratory

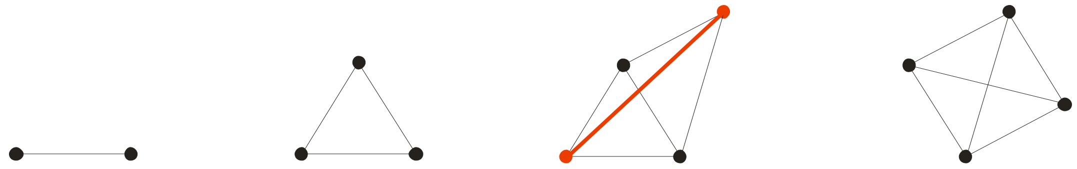
Structure solution from PDF

Input: PDF

Output: nanoparticle structure

Structure solution from Powder data: Liga Algorithm 2006

- square-distances = [4×1, 2×sqrt(2)]

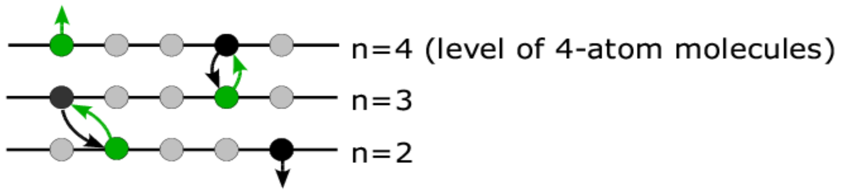


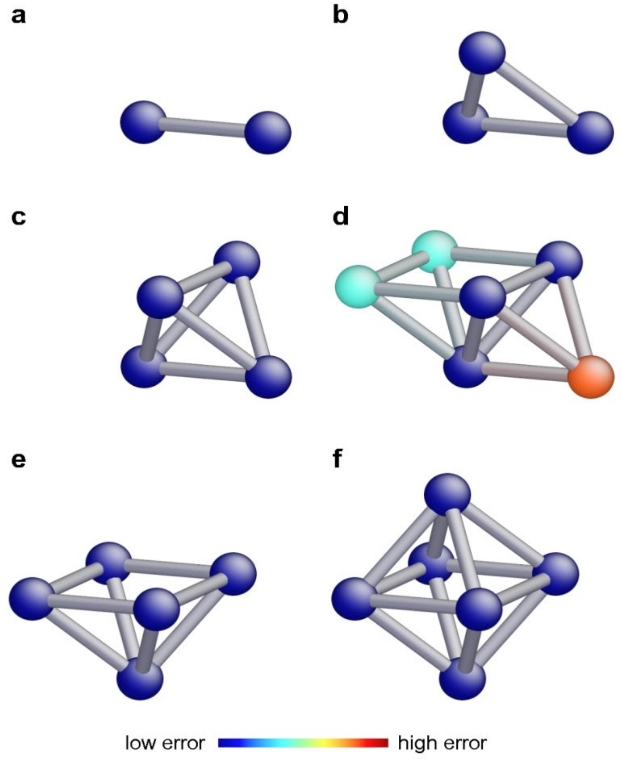
- octahedron-distances = [12×1, 3×sqrt(2)]

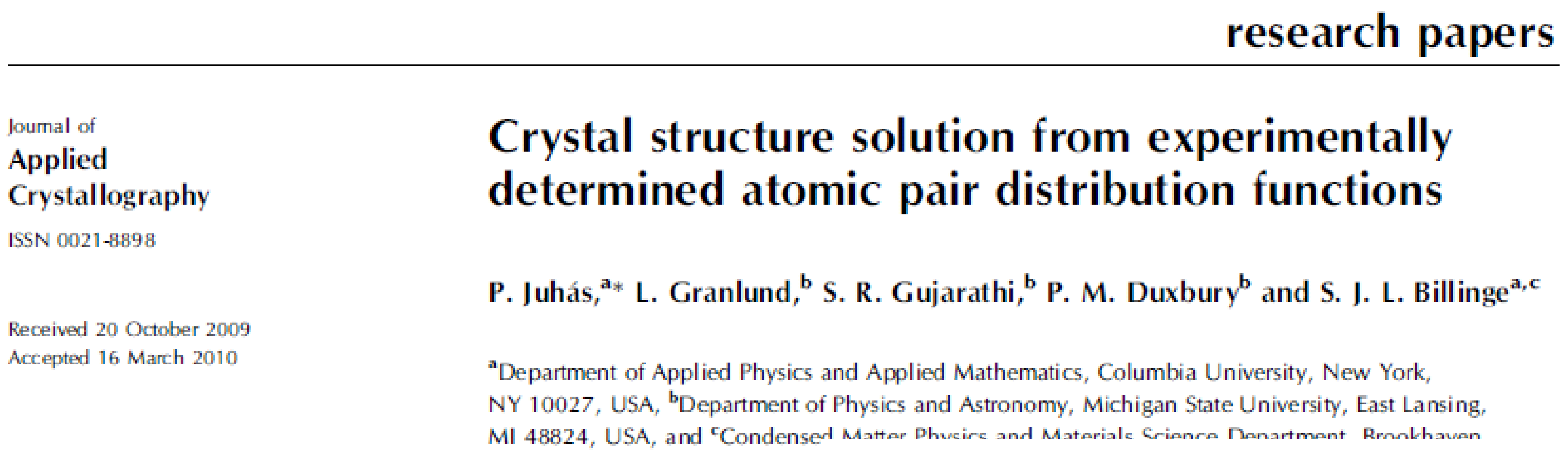
- minimized cost function:

$$\mathrm{var}(d) = \frac{1}{P}\sum_{k=1}^{P}[d_k - t_{l(k)}]^2$$

Juhas, SJB et al., Nature 2006







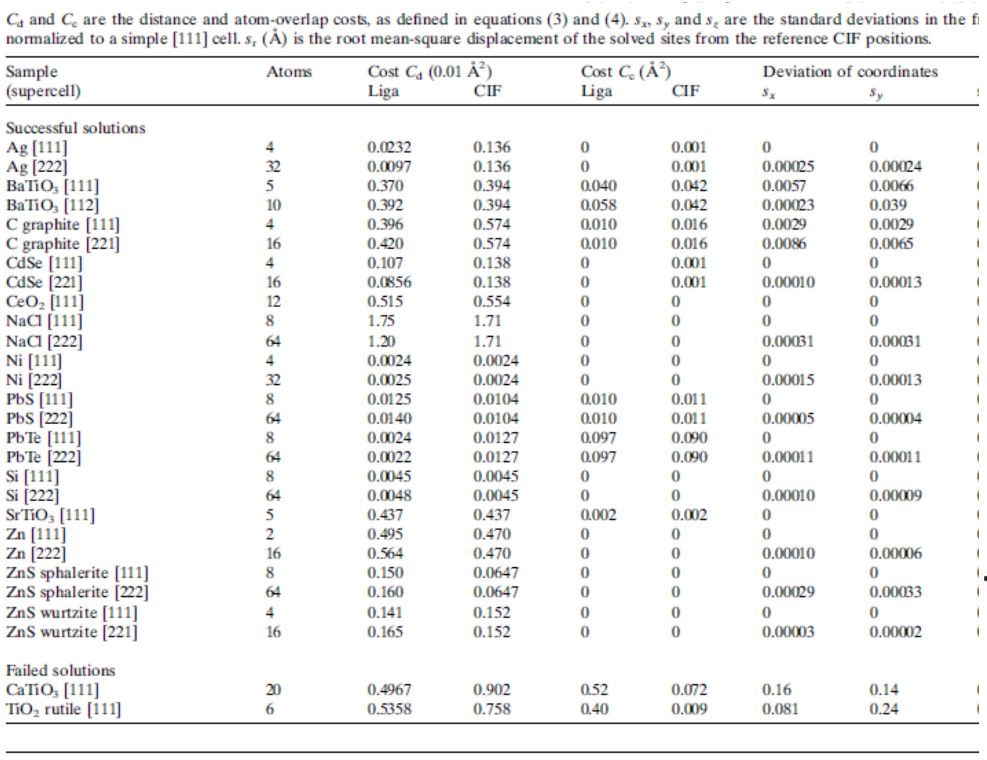
Journal of
Applied
Crystallography
ISSN 0021-8898

Received 20 October 2009
Accepted 16 March 2010

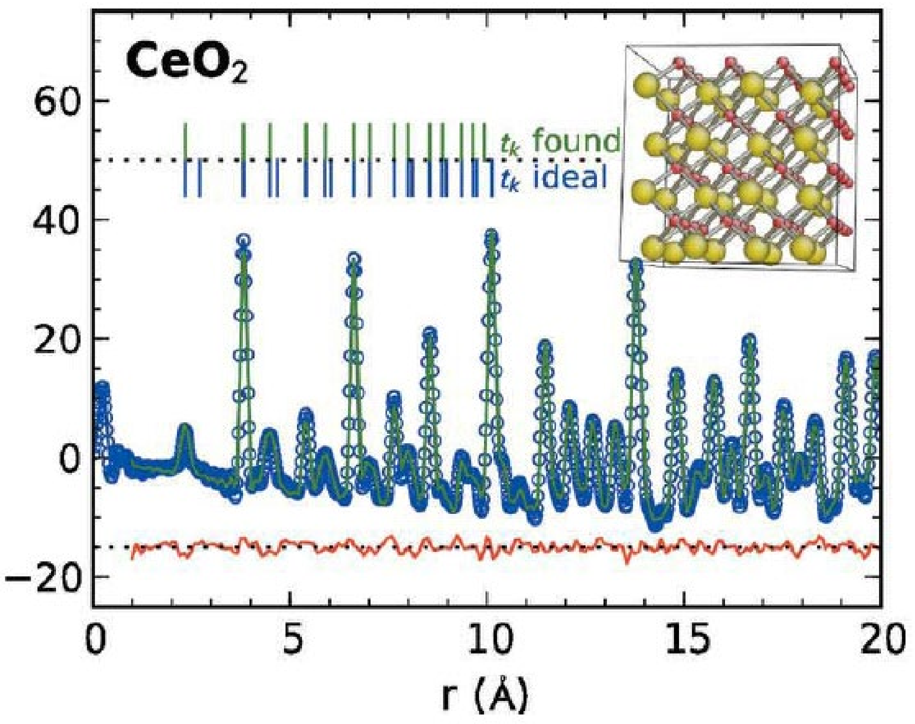
Crystal structure solution from experimentally determined atomic pair distribution functions

P. Juhás,[a]* L. Granlund,[b] S. R. Gujarathi,[b] P. M. Duxbury[b] and S. J. L. Billinge[a,c]

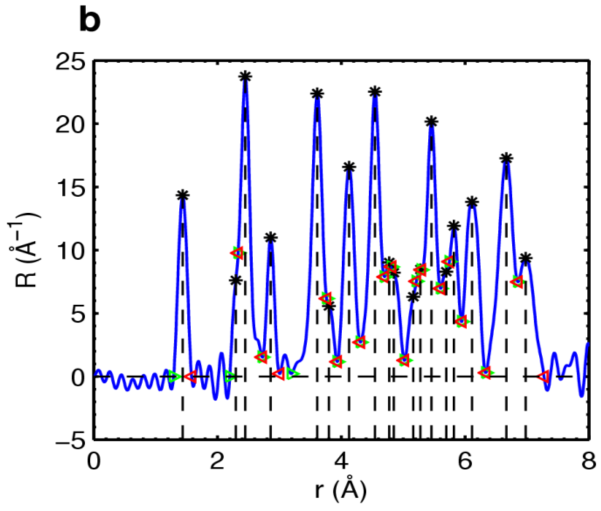
[a]Department of Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027, USA, [b]Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA, and [c]Condensed Matter Physics and Materials Science Department, Brookhaven

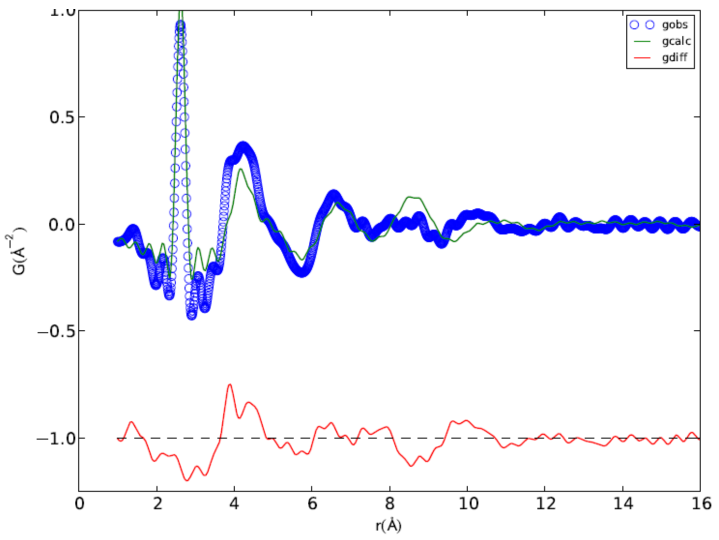

C_d and C_c are the distance and atom-overlap costs, as defined in equations (3) and (4). s_x, s_y and s_z are the standard deviations in the f
normalized to a simple [111] cell. s_r (Å) is the root mean-square displacement of the solved sites from the reference CIF positions.

Sample (supercell)	Atoms	Cost C_d (0.01 Å^2) Liga	CIF	Cost C_c (Å^2) Liga	CIF	Deviation of coordinates s_x	s_y
Successful solutions							
Ag [111]	4	0.0232	0.136	0	0.001	0	0
Ag [222]	32	0.0097	0.136	0	0.001	0.00025	0.00024
$BaTiO_3$ [111]	5	0.370	0.394	0.040	0.042	0.0057	0.0066
$BaTiO_3$ [112]	10	0.392	0.394	0.058	0.042	0.00023	0.039
C graphite [111]	4	0.396	0.574	0.010	0.016	0.0029	0.0029
C graphite [221]	16	0.420	0.574	0.010	0.016	0.0086	0.0065
CdSe [111]	4	0.107	0.138	0	0.001	0	0
CdSe [221]	16	0.0856	0.138	0	0.001	0.00010	0.00013
CeO_2 [111]	12	0.515	0.554	0	0	0	0
NaCl [111]	8	1.75	1.71	0	0	0	0
NaCl [222]	64	1.20	1.71	0	0	0.00031	0.00031
Ni [111]	4	0.0024	0.0024	0	0	0	0
Ni [222]	32	0.0025	0.0024	0	0	0.00015	0.00013
PbS [111]	8	0.0125	0.0104	0.010	0.011	0	0
PbS [222]	64	0.0140	0.0104	0.010	0.011	0.00005	0.00004
PbTe [111]	8	0.0024	0.0127	0.097	0.090	0	0
PbTe [222]	64	0.0022	0.0127	0.097	0.090	0.00011	0.00011
Si [111]	8	0.0045	0.0045	0	0	0	0
Si [222]	64	0.0048	0.0045	0	0	0.00010	0.00009
$SrTiO_3$ [111]	5	0.437	0.437	0.002	0.002	0	0
Zn [111]	2	0.495	0.470	0	0	0	0
Zn [222]	16	0.564	0.470	0	0	0.00010	0.00006
ZnS sphalerite [111]	8	0.150	0.0647	0	0	0	0
ZnS sphalerite [222]	64	0.160	0.0647	0	0	0.00029	0.00033
ZnS wurtzite [111]	4	0.141	0.152	0	0	0	0
ZnS wurtzite [221]	16	0.165	0.152	0	0	0.00003	0.00002
Failed solutions							
$CaTiO_3$ [111]	20	0.4967	0.902	0.52	0.072	0.16	0.14
TiO_2 rutile [111]	6	0.5358	0.758	0.40	0.009	0.081	0.24



Liga works! But it isn't a general solution to the Nanostructure inverse problem. Why not?





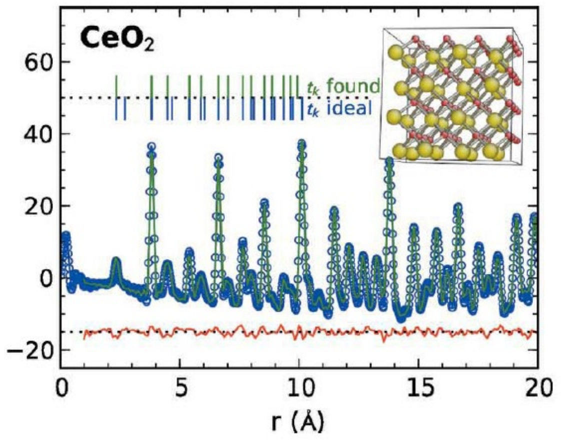
60 atoms

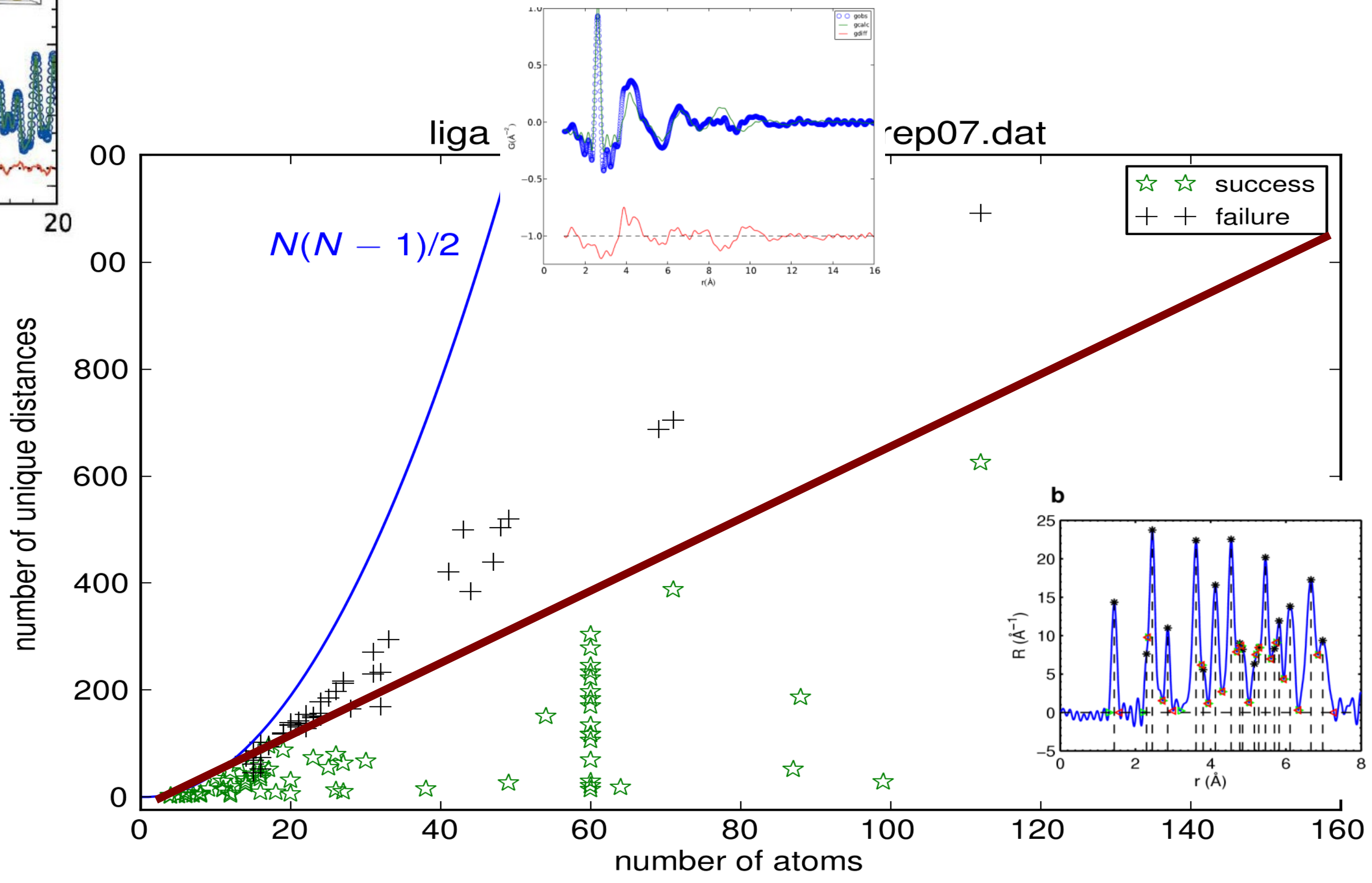
~64 atoms

C_{60}

Ultra-small CdSe NPs

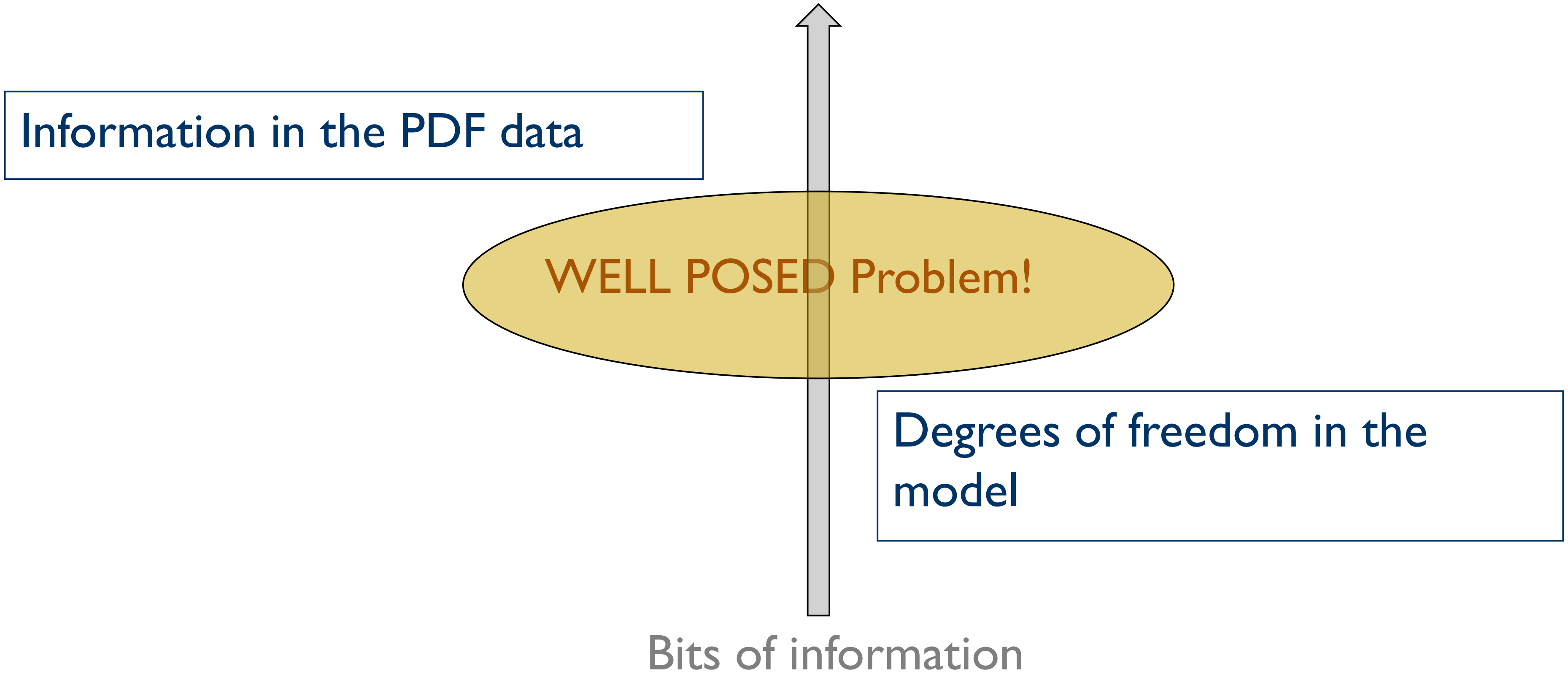
Successology





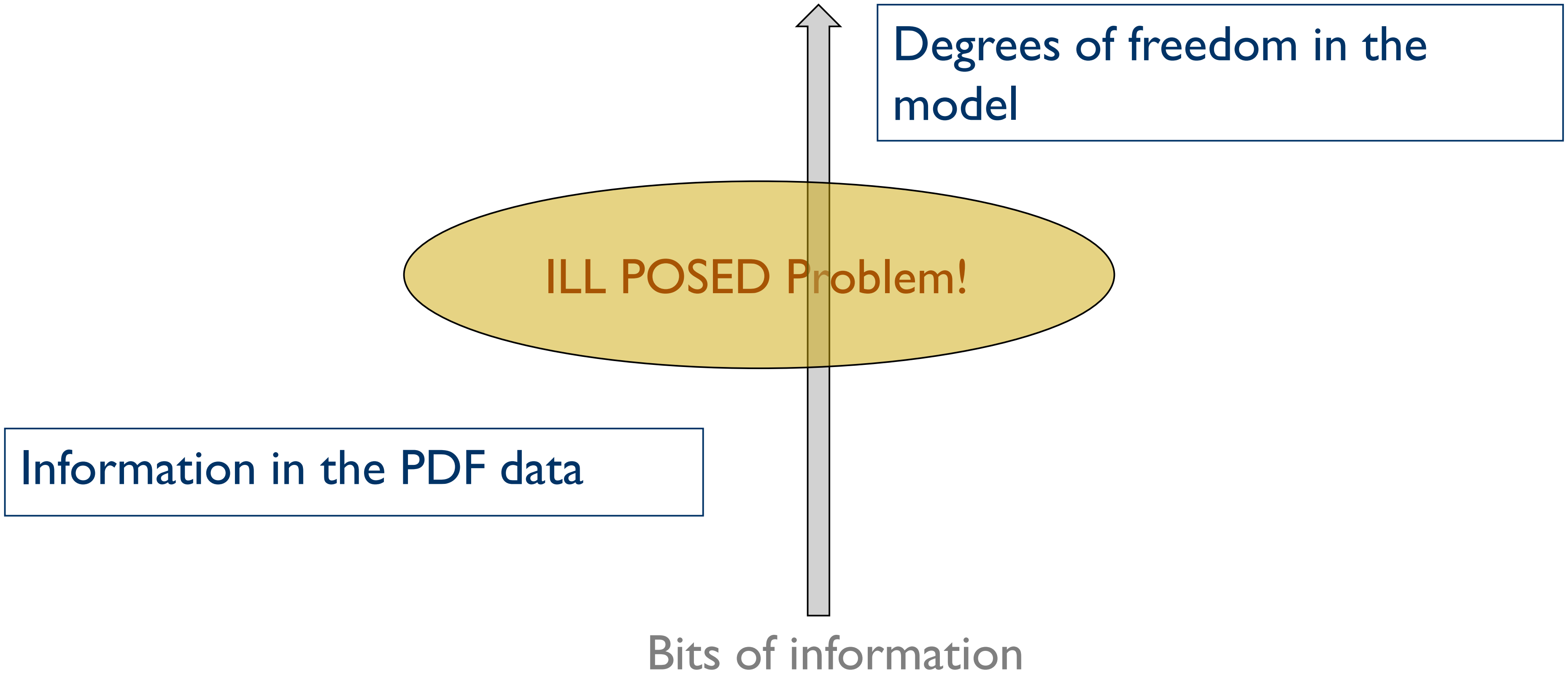
Problem

Well posed problem:



Problem

Ill posed problem:



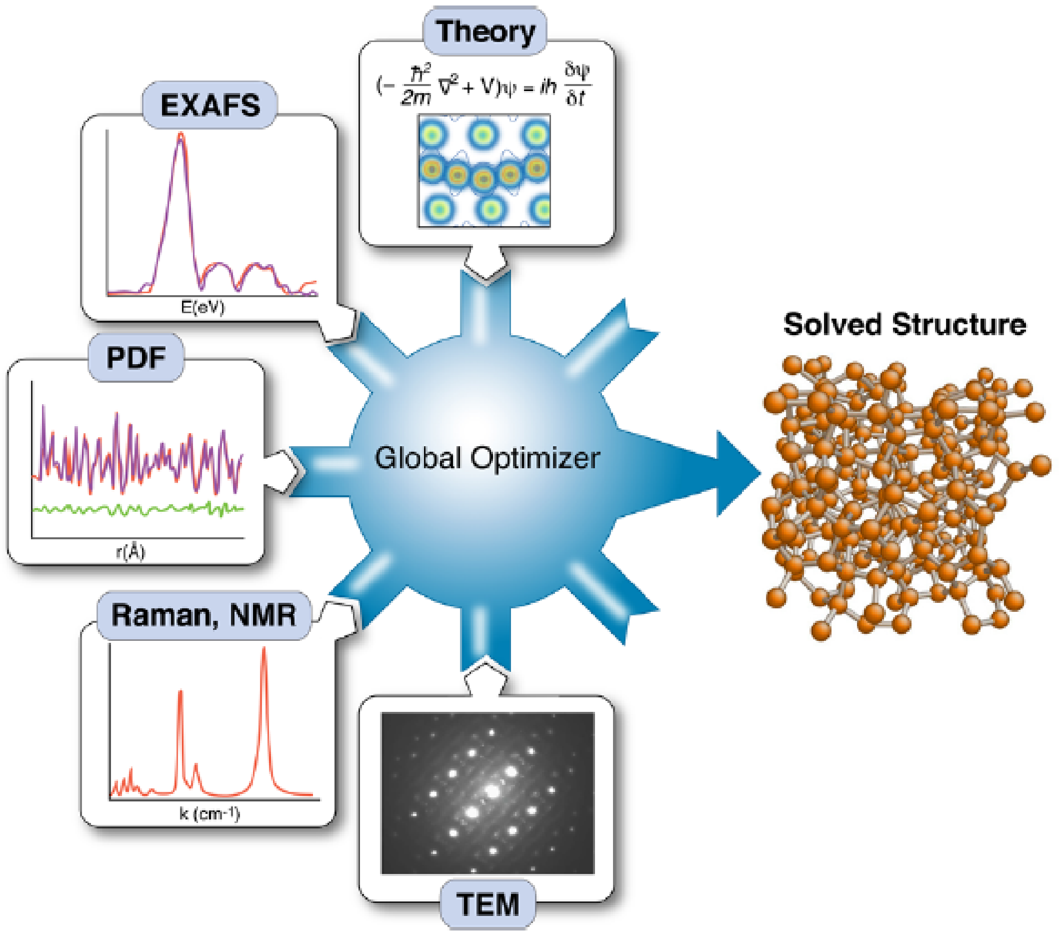
Solution

1. Increase the information content from experiments

2. Decrease the degrees of freedom in the model

=> Complex modeling

Complex modeling



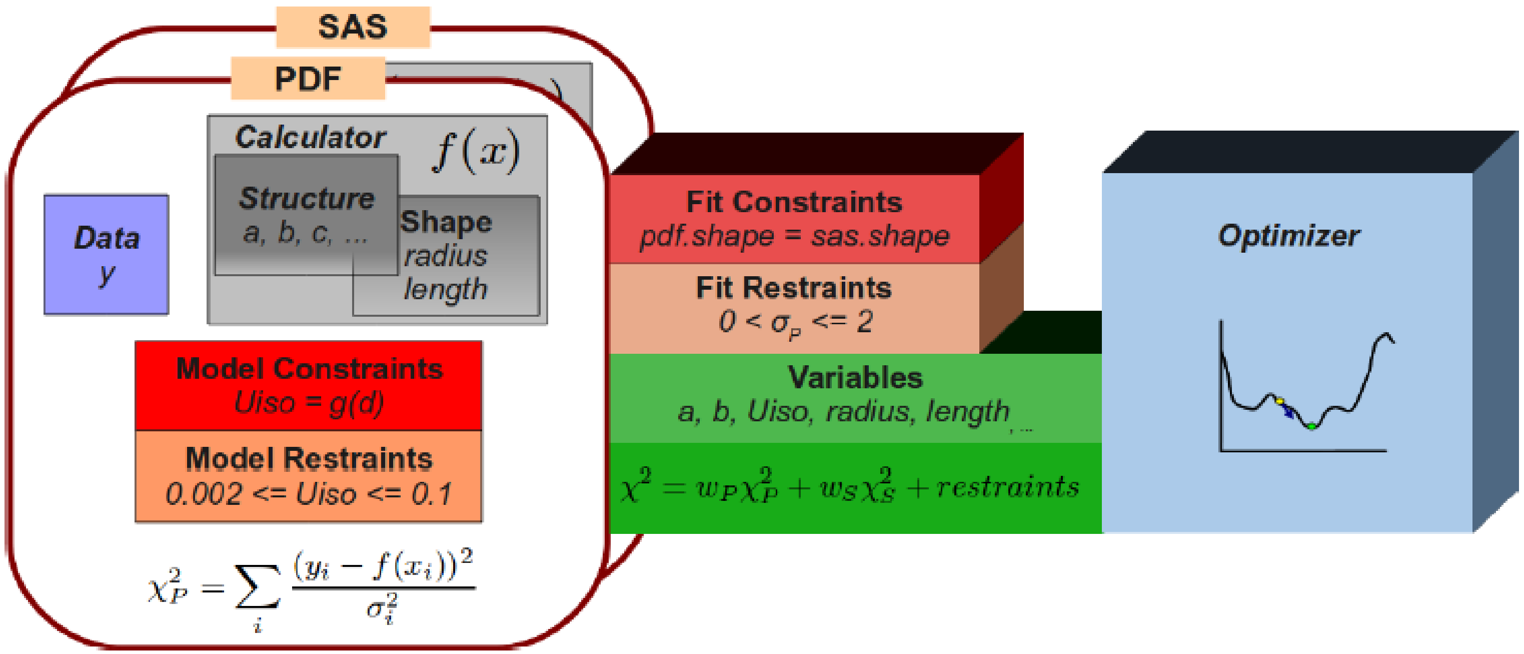
Problem

- not enough information in the available experimental data

Remedy

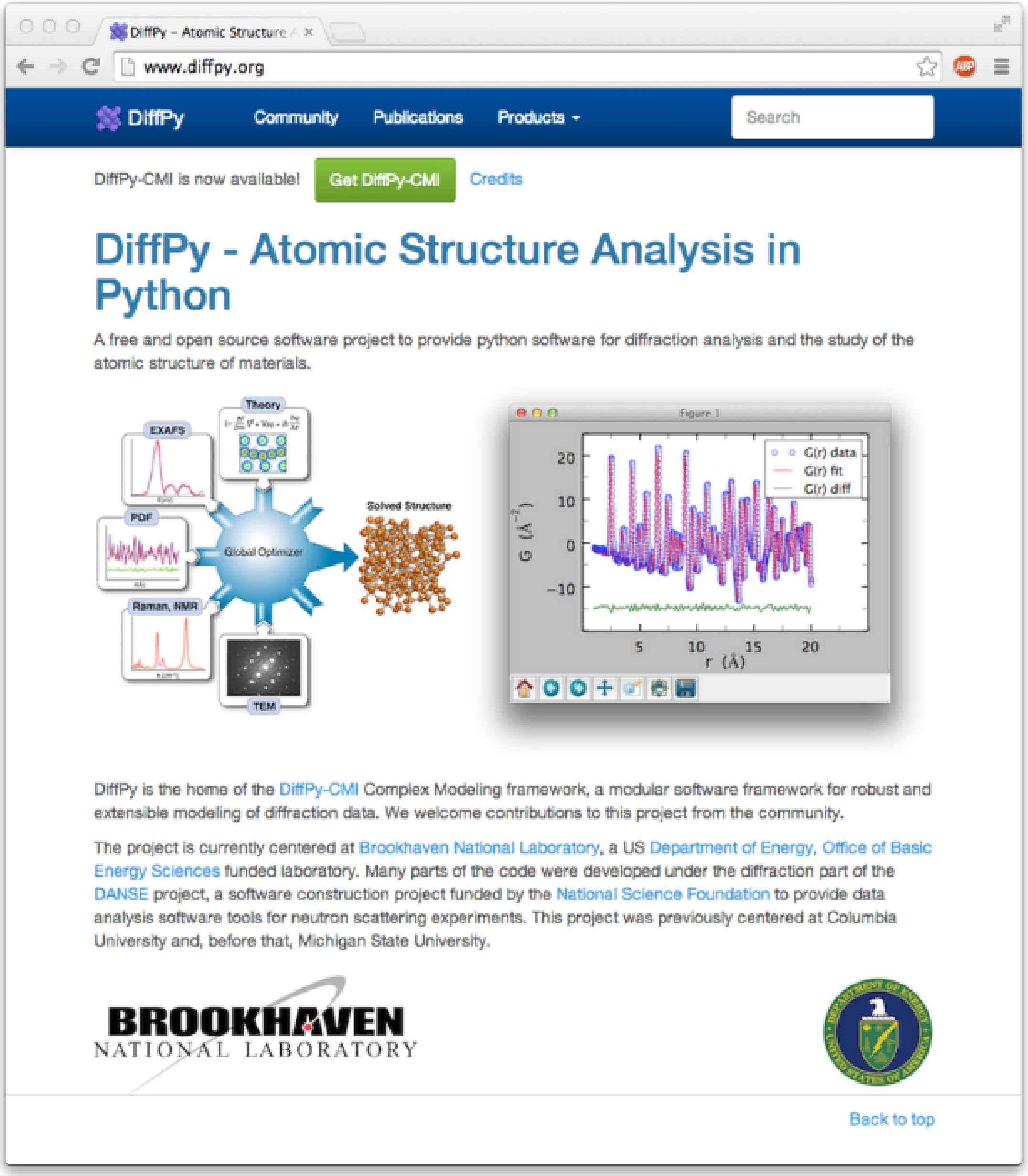
- collect data from multiple experimental techniques
- use additional knowledge about the studied material - chemical constraints, rigid units, bond-valence sums, energy calculation
- combine all experimental and theoretical inputs about the structure in one optimization scheme
- requires flexible software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

Software infrastructure for doing that



Diffpy-CMI

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.

- 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

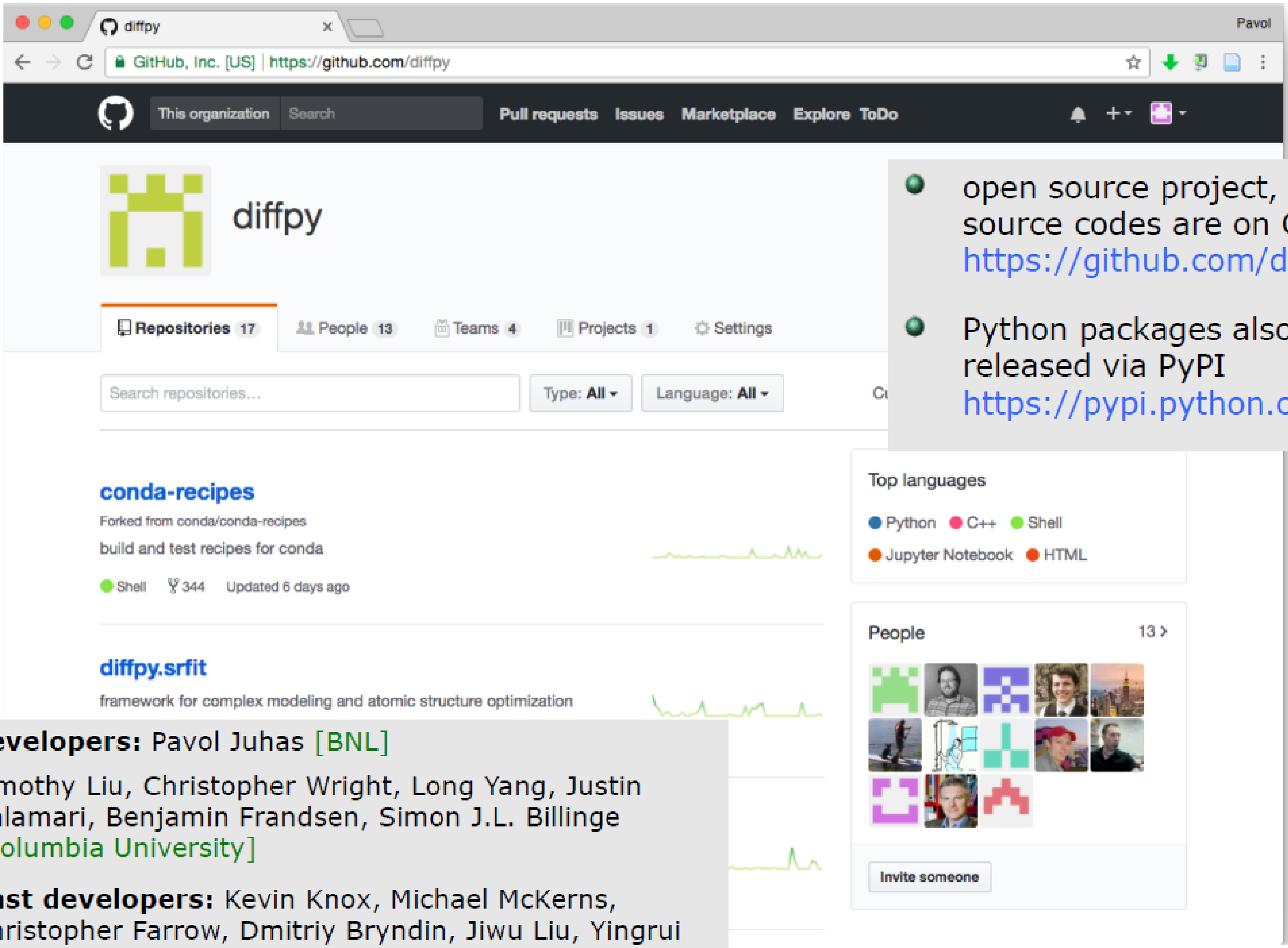
upgrade release March 2019

- added support for Python 3

Anaconda Python installation

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

DiffPy-CMI – open source project



diffpy

https://github.com/diffpy

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

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

- 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

BROOKHAVEN
NATIONAL LABORATORY

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].

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}
```

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
```

PDFCalculator

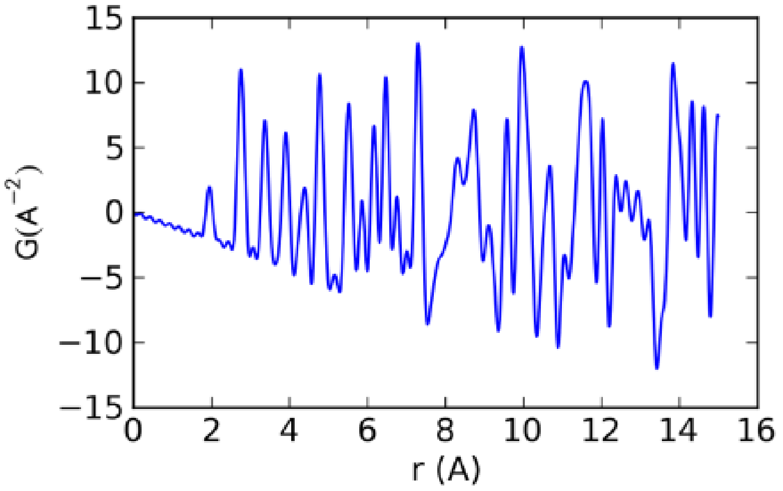
- PDF calculation in real-space
  - suitable for periodic systems
  - one structure per calculator → mixed-phase PDFs obtained by summing single-phase PDFs

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

- 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*)

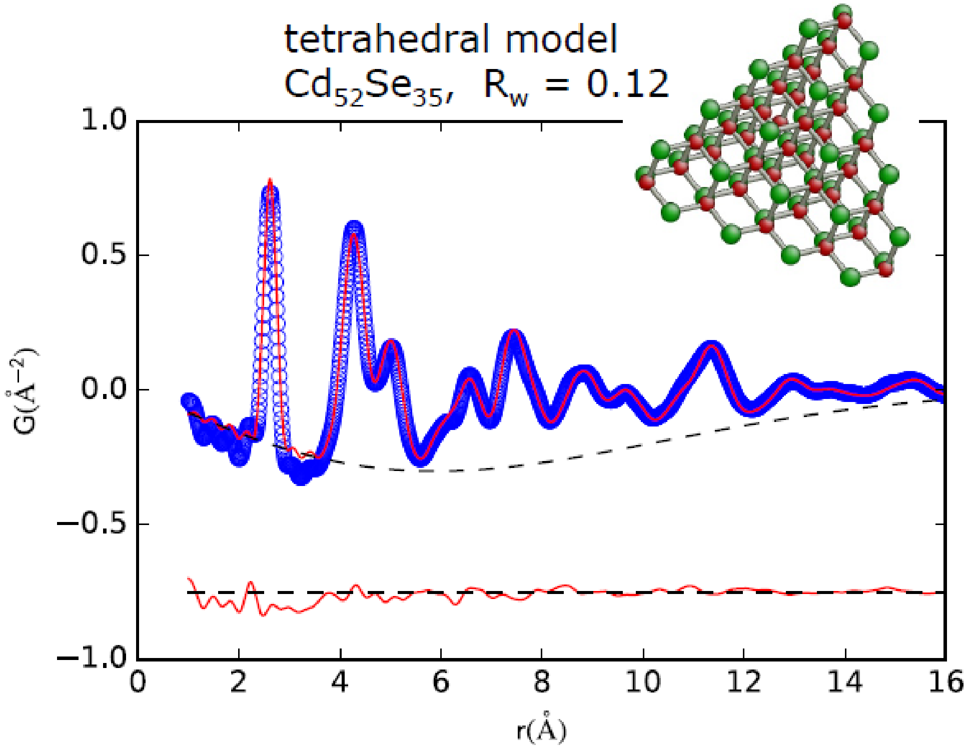
$$F(Q) = \frac{1}{N\langle f(Q)\rangle^2} \sum_{i,j} f_i(Q) f_j(Q) \frac{\sin Q r_{ij}}{r_{ij}} \exp\left[-\frac{1}{2}\,\sigma_{ij}^2\, Q^2\right] \qquad G(r) = \frac{2}{\pi} \int_{Q_{\min}}^{Q_{\max}} F(Q) \sin Qr \, \mathrm{d}Q$$

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

example:

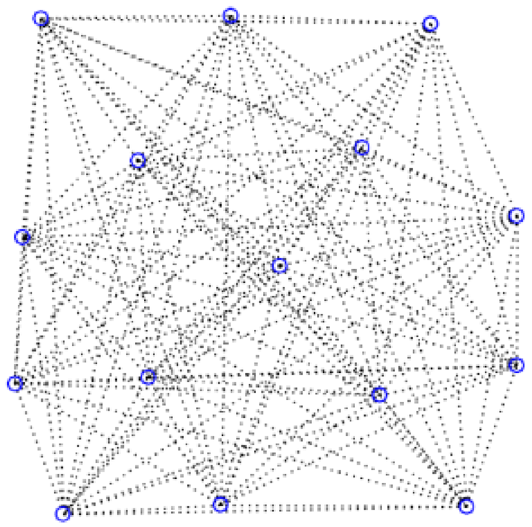
- structure refinement of CdSe quantum dots
- simulated PDF reflects particle shape in amplitude dampening and baseline shape

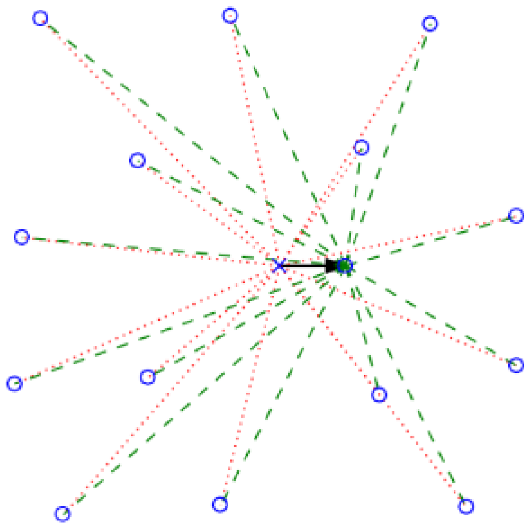
A. Beecher, J. Am. Chem. Soc., 2014, 136 (30), 10645–10653



Optimized PDF evaluation

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

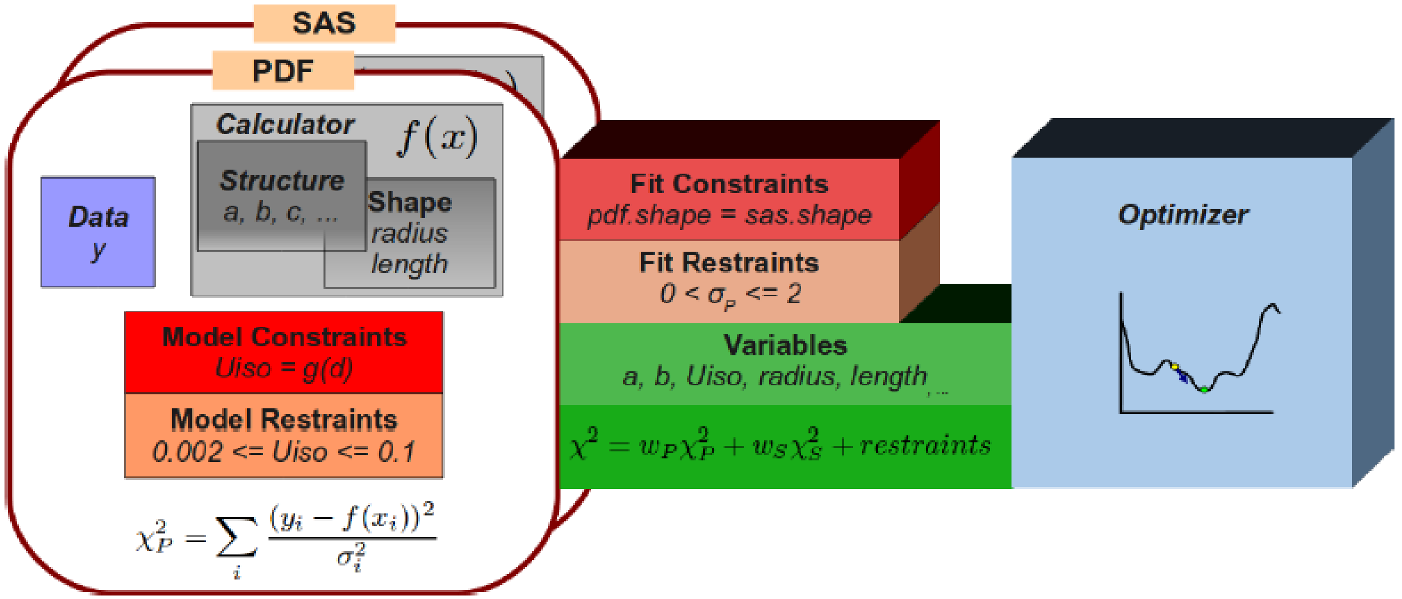




- real-space and Debye PDF calculators support optimized PDF calculation.
- when few atoms change, their old contributions are subtracted and new ones added to the PDF. Computational cost is reduced from N^2 → N.
- optimized PDF calculation is about 1000 faster for one-atom updates in 10,000 atom structure

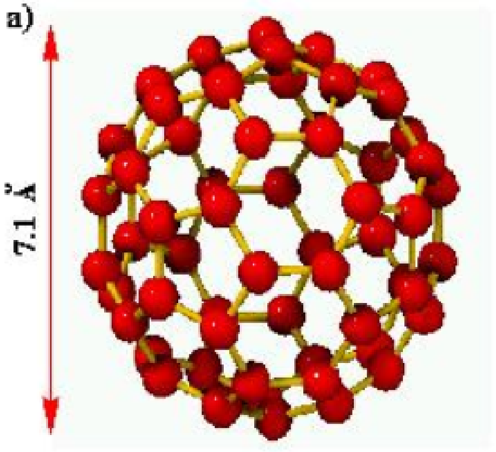
SrFit – multi-component fit manager

- 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.

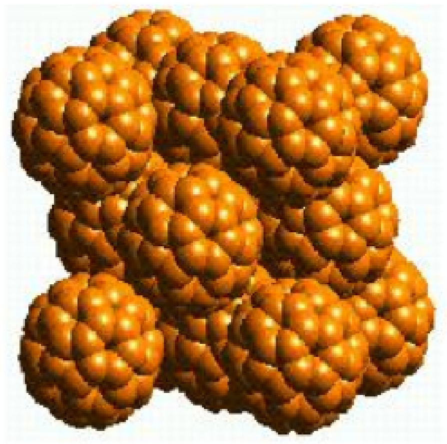


PDF modeling of fcc-C_{60}

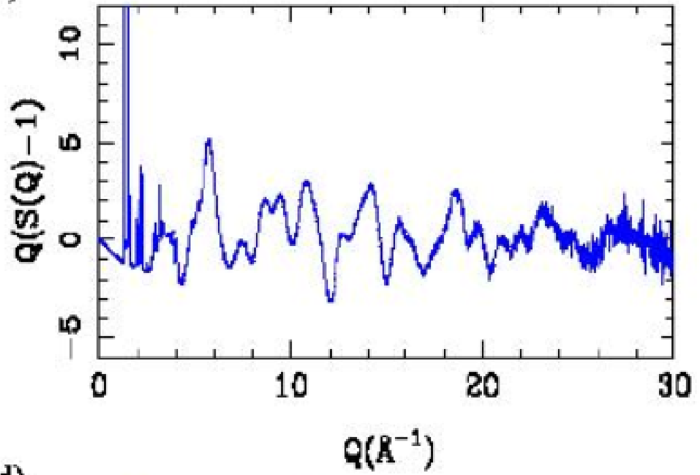
a)



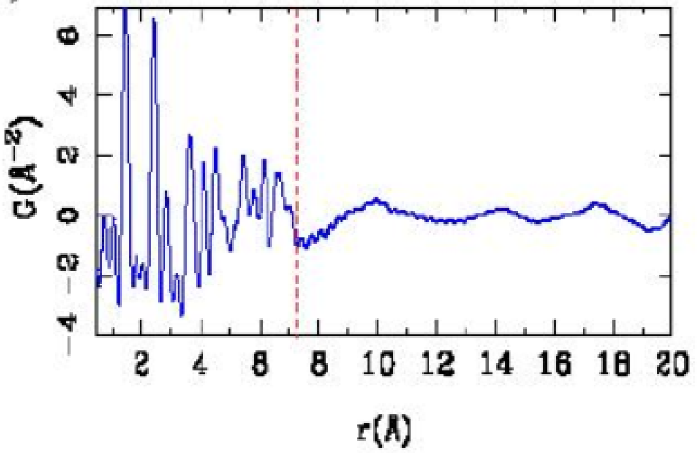
b)



(c)



(d)

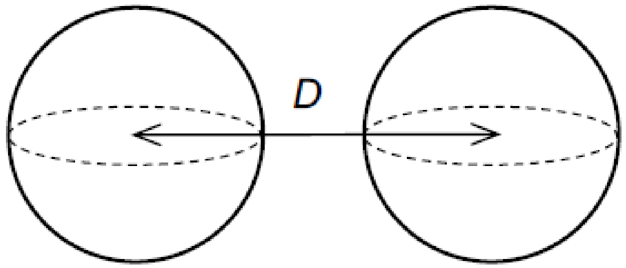


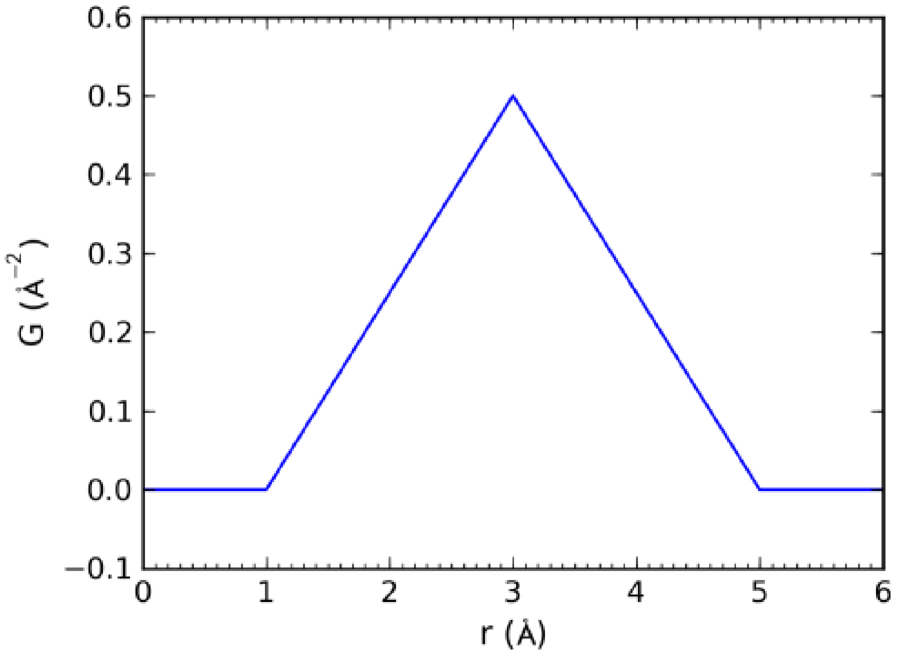
- neutron PDF measured on C_{60} fcc structure [GLAD IPNS, E. Bozin]
- low-*r* sharp peaks – correlations within C_{60}
- high-*r* broad peaks – correlations between randomly oriented balls

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

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

  triangular profile centered at spheres' distance *D*

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

Custom PDF peak profile

profile defined in C++

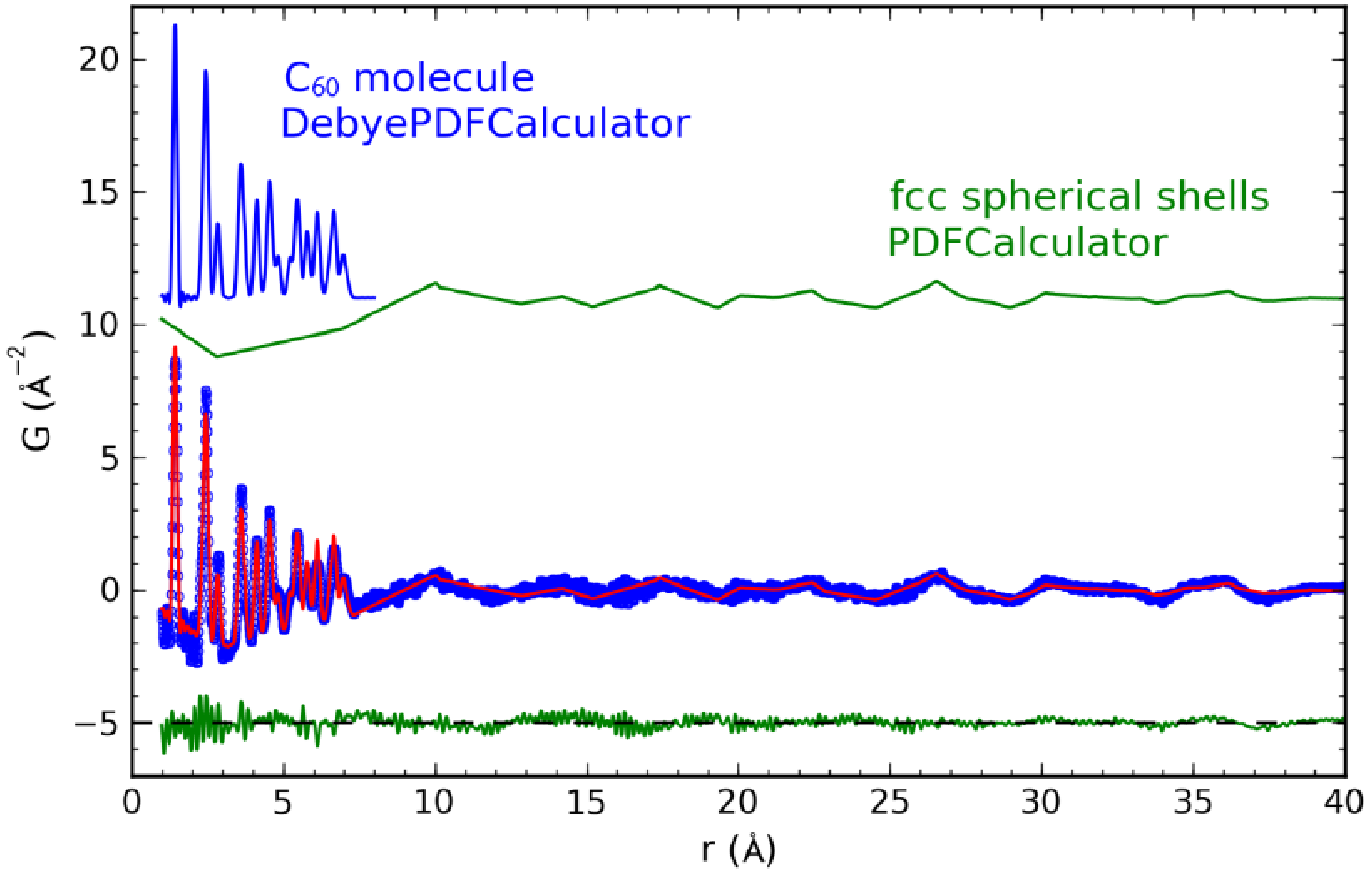
```
#include <cmath>
#include <diffpy/srreal/PeakProfile.hpp>

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

PDF refinement of fcc C_{60}

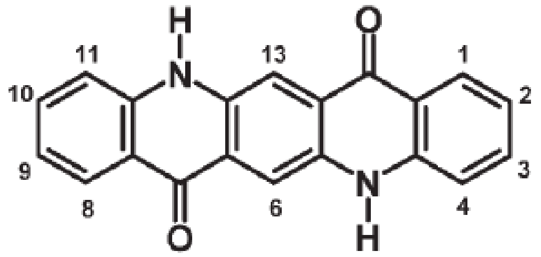


fit residuum R_w = 0.26 | scale ratio = 60.0(1) | U_{iso} = 0.00323(4)

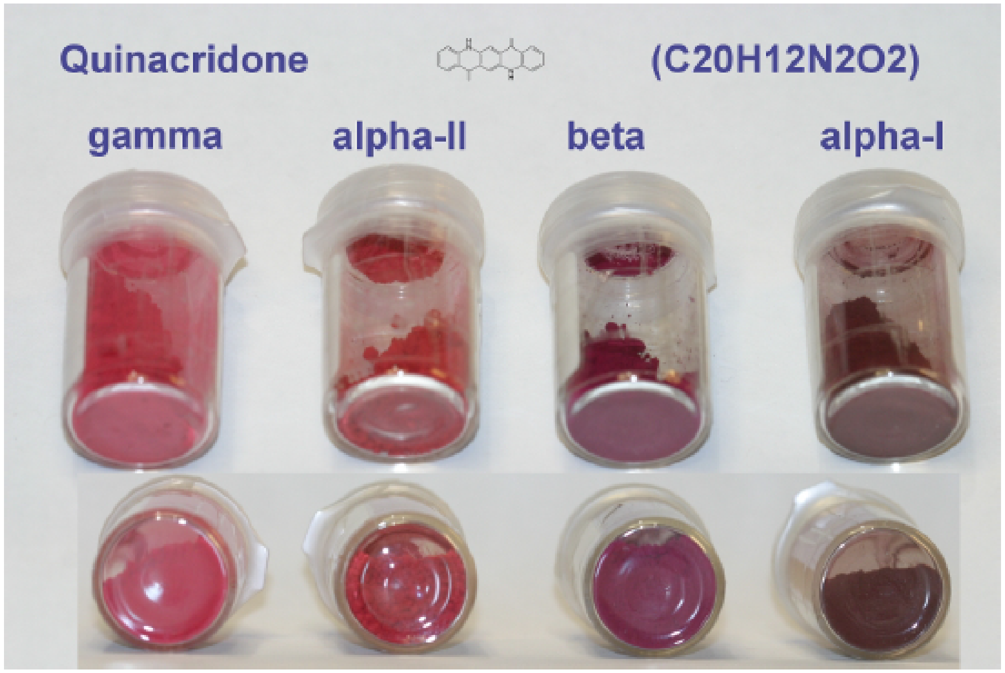
molecule diameter = 7.113(2) | shell diameter = 7.22(4)

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

PDF analysis of organic crystals



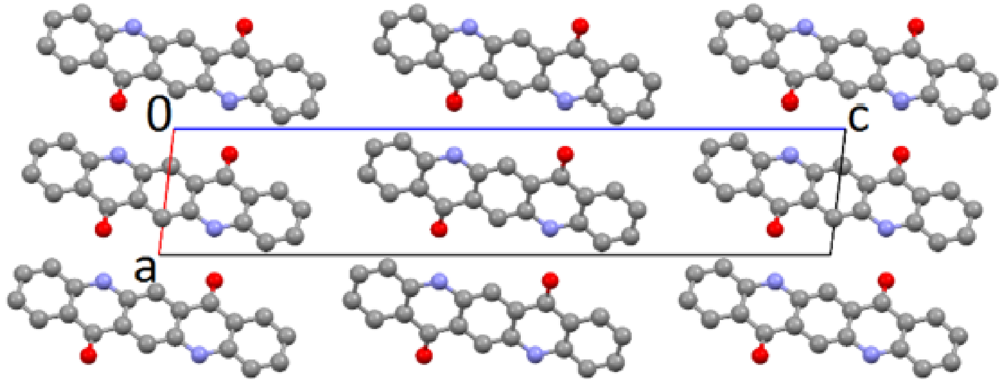
quinacridone – $C_{20}H_{12}N_2O_2$

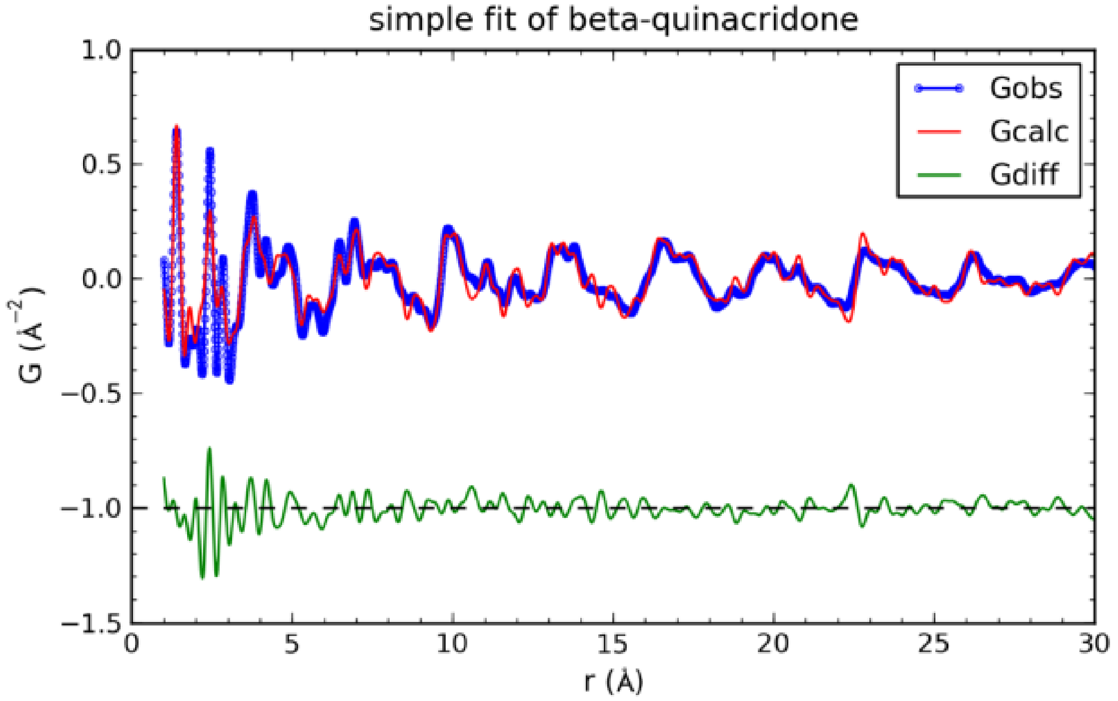


- industrially important pigments, red and violet paints
- can form many phases, some (alpha-II) do not crystallize and have unknown structure
- experimental PDFs measured at APS ANL beamline 11-ID-B and NSLS BNL beamline X17A
- standard refinement with PDFgui is of poor quality even for the known β-phase

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

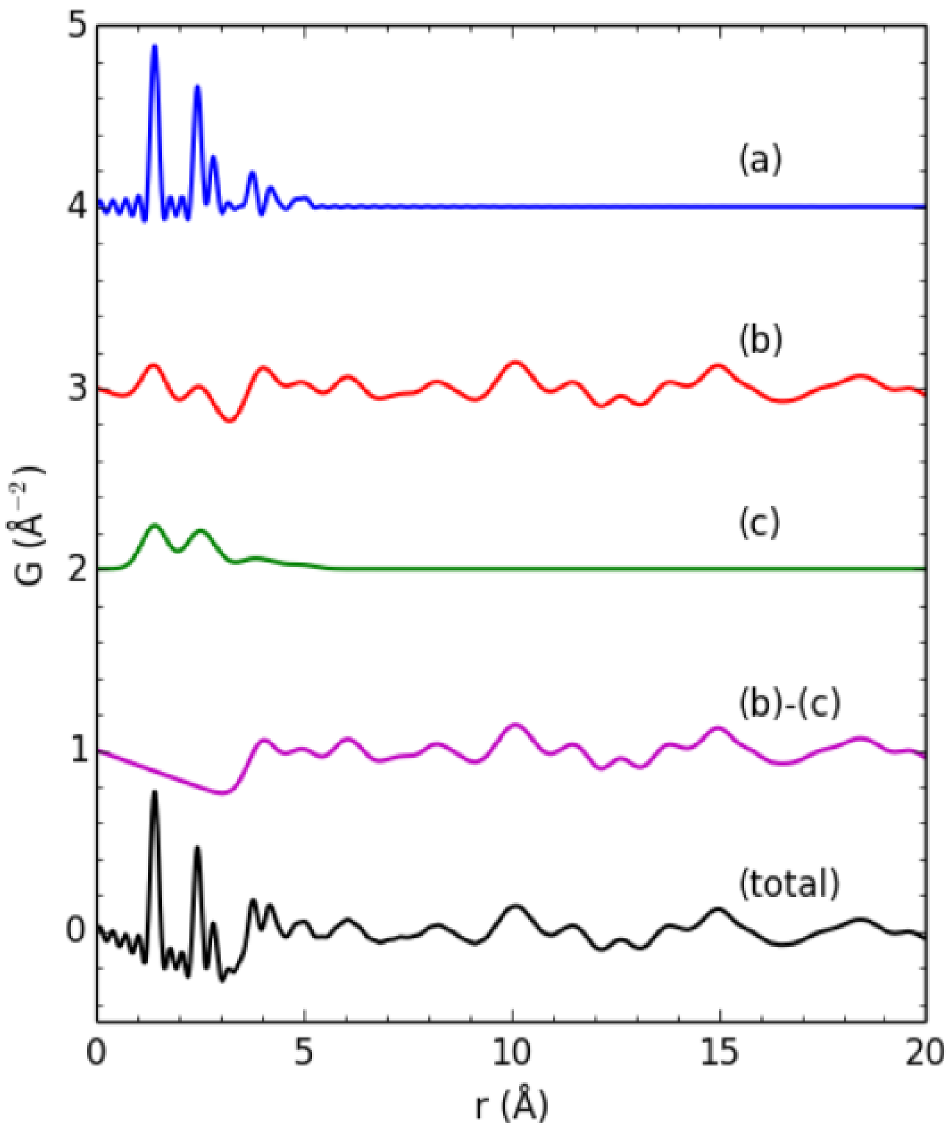
PDF modeling of β-quinacridone





- monoclinic $P2_1/c$, 2 molecules per unit cell
- refinement in PDFgui gives poor ***R_w* = 0.41**
  - low-*r* peaks too wide
  - high-*r* peaks too sharp
- simple model assumes independent 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

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}

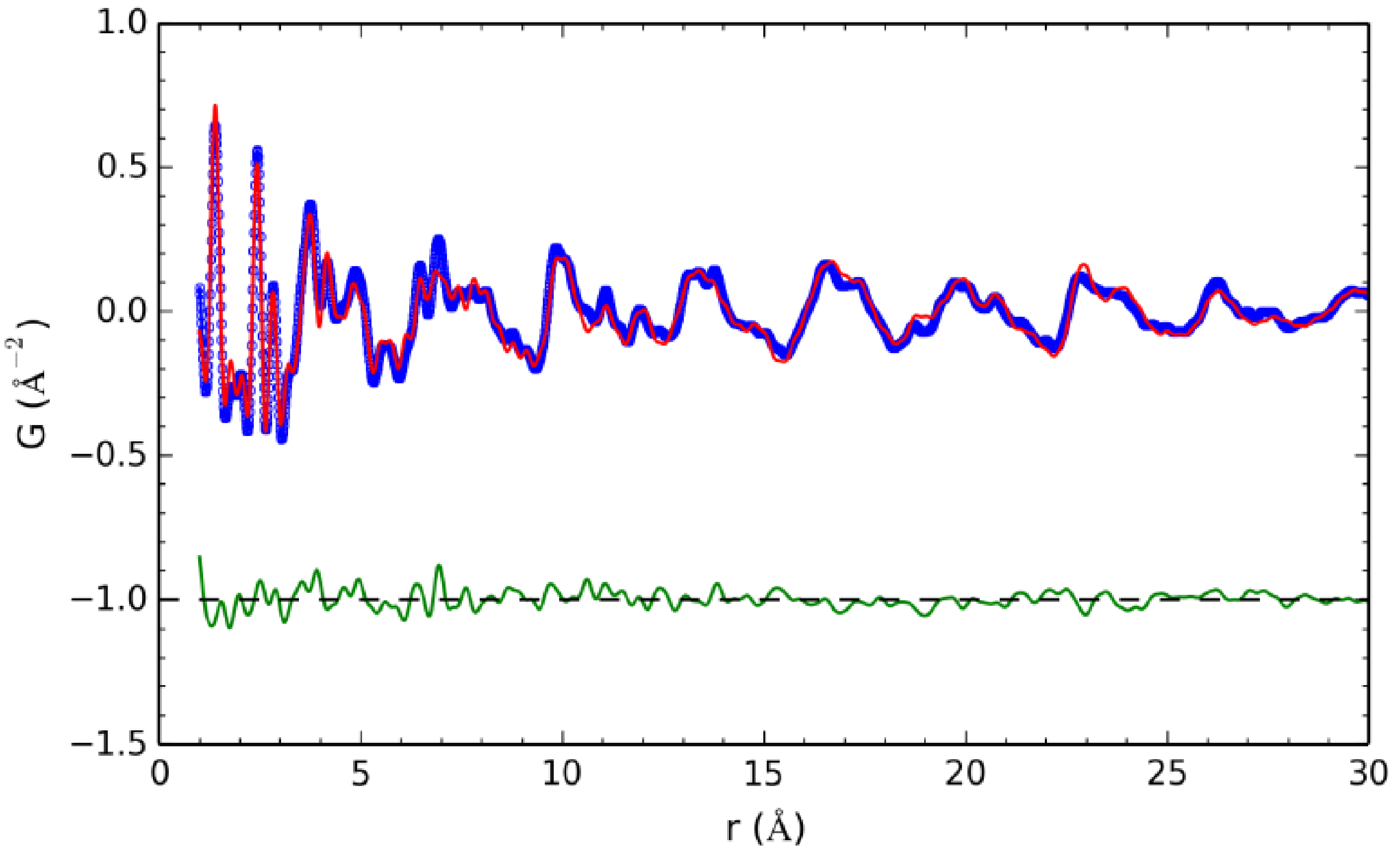
(b) PDF from a crystal with large displacements U_{inter}

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

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

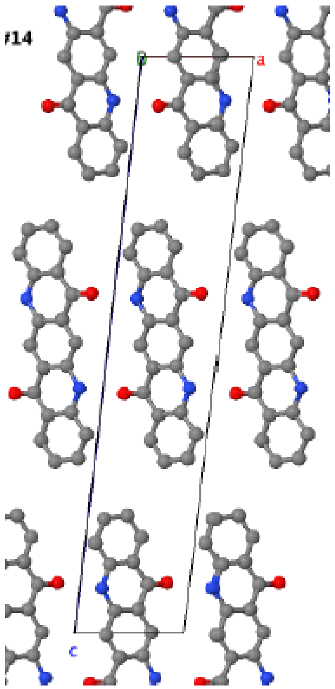
(a) + (b) – (c) total PDF reflecting both displacements U_{intra}, U_{inter}

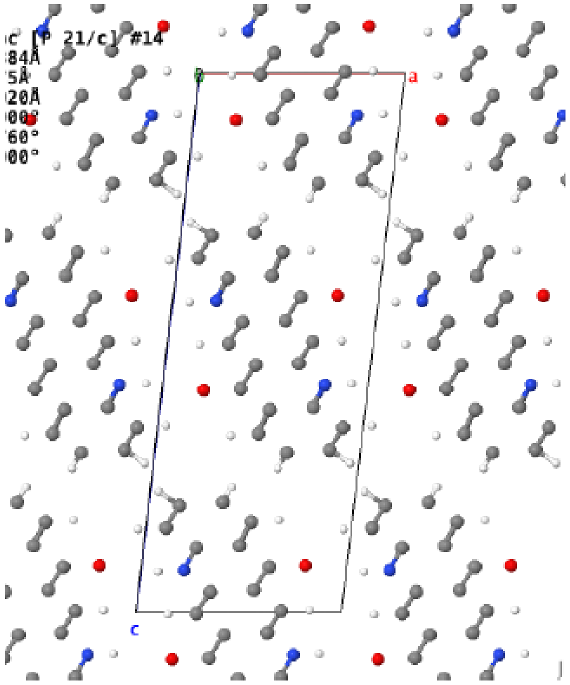
PDF modeling of β-quinacridone

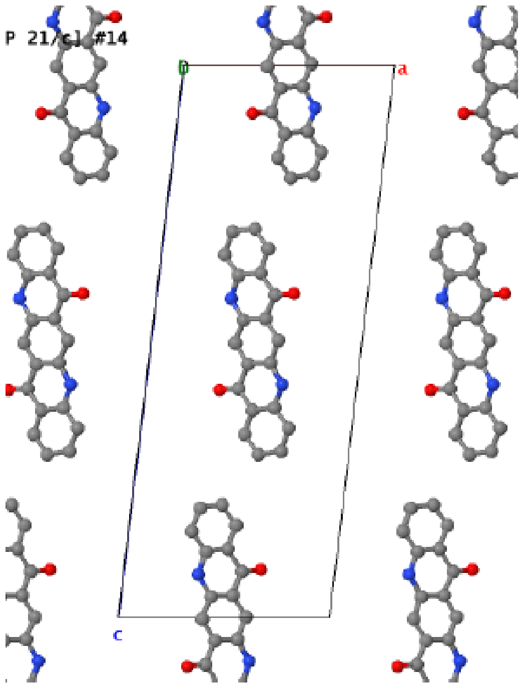


- refined unit cell, data scale and displacement factors
  U_{inter} = 0.0014(2) Å^2 U_{intra} = 0.023(2) Å^2
- significant fit improvement R_w = 0.41 → $\boldsymbol{R_w}$ **= 0.28**
- remnant fit difference due to anisotropic molecule displacements, displacement anisotropy can be studied with improved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

Structure representation with rigid molecules



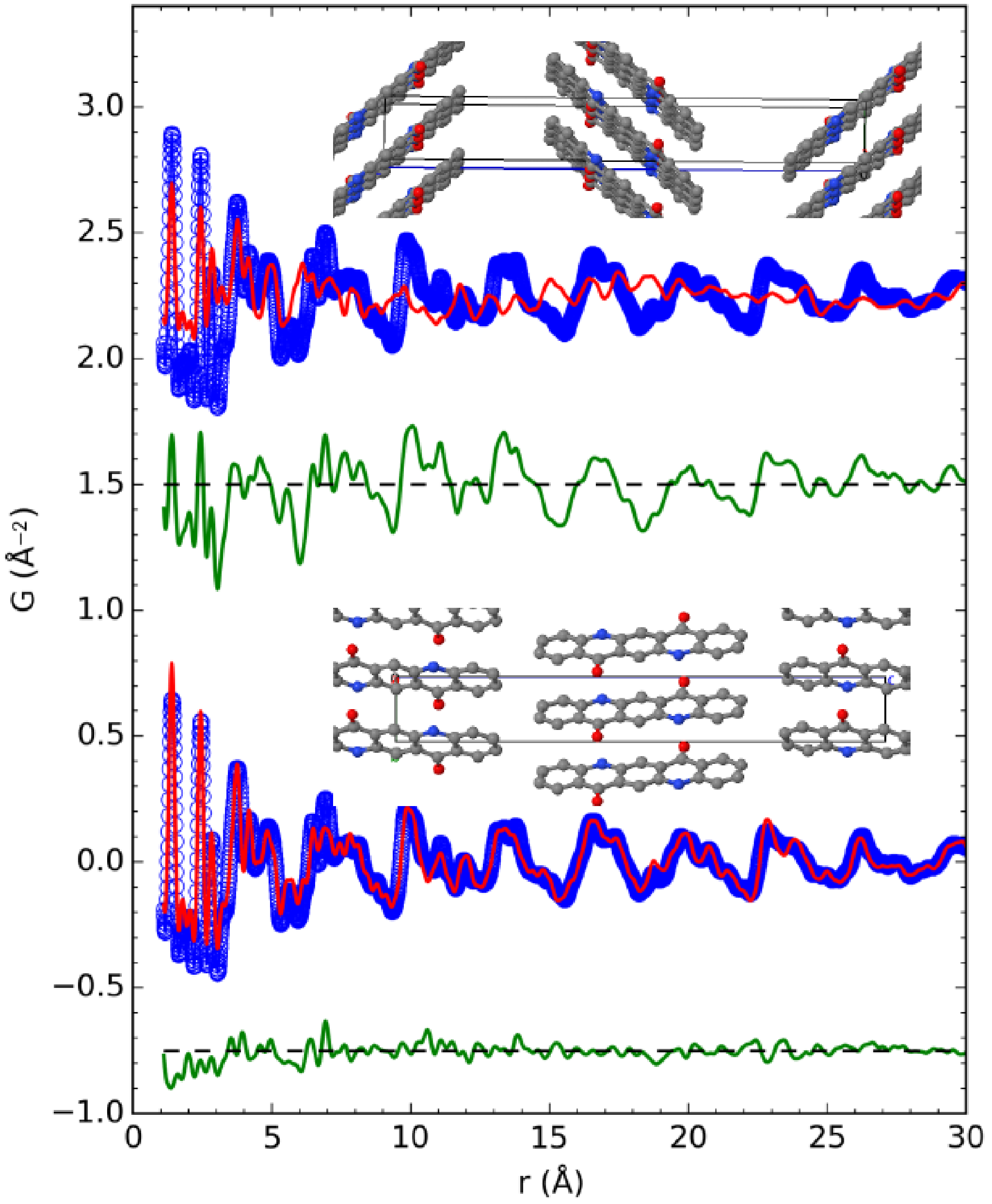




- structure representation needs to handle molecules as a rigid unit within crystal lattice → pyobjcryst in DiffPy-CMI, an interface to ObjCryst++
- molecule retains its shape when cell parameters change
- molecule placement is defined by its center-of-mass and orientation quaternion

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

Solution of molecule orientation in β-quinacridone

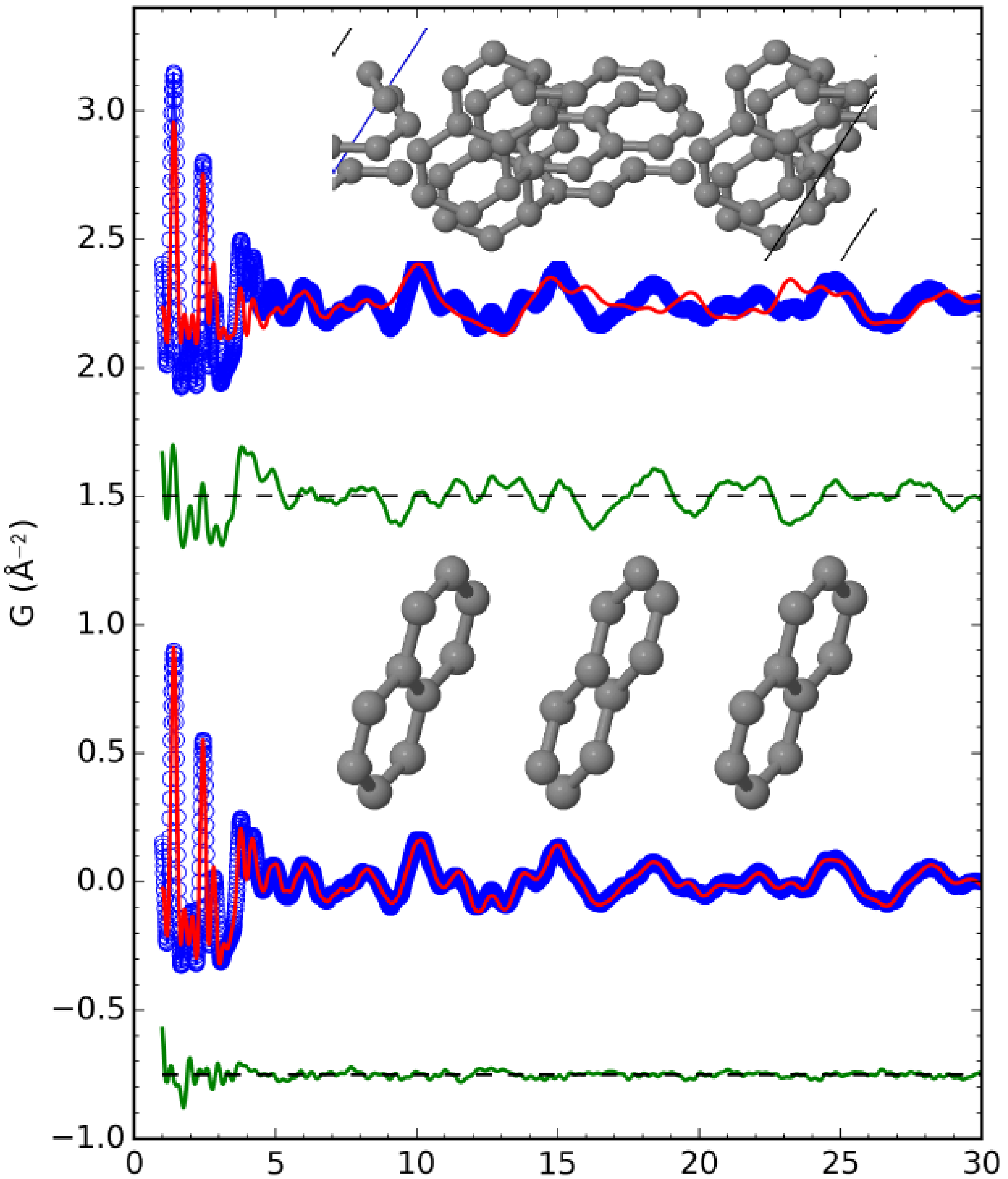


- apply random rotation on the quinacridone molecule in asymmetric unit
- preserve $P2_1/c$ symmetry
  - molecule position fixed at inversion center
  - use symmetric rotation for the second molecule in the UC
- refine molecule orientation, cell parameters, ADPs: $R_w = 0.83 \rightarrow R_w = 0.27$
- 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 → R_w = 0.16
- optimized structure converged to $P2_1/a$ naphthalene

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

BROOKHAVEN

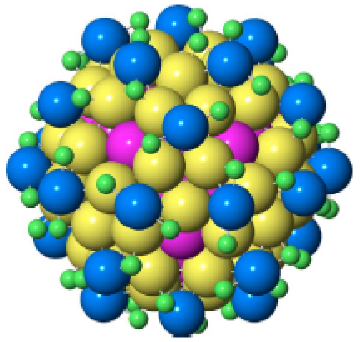
Structure solutions of molecular crystals

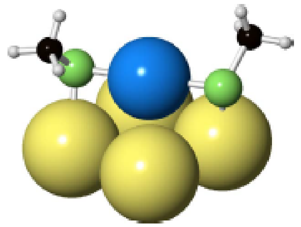
sample	symmetry	molecules	DOF	success rate
β-quinacridone	$P2_1/c$	1	3	6.4% (14/220)
naphthalene	$P2_1/a$	1	3	33% (79/240)
naphthalene	P1	2	9	2.5% (7/280)
allopurinol	$P2_1/c$	1	3	3.6% (12/329)
allopurinol	P1	4	21	0.4% (2/534)

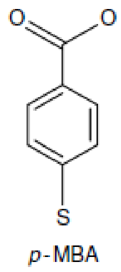
- structure determination of molecular crystals
  - start from random initial placement of molecule(s) in the unit cell
  - fit the PDF, optimize rotation, position, cell parameters, intra- and inter-molecular ADPs
  - repeat to search for best convergence
- correct solutions were found also at lowered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

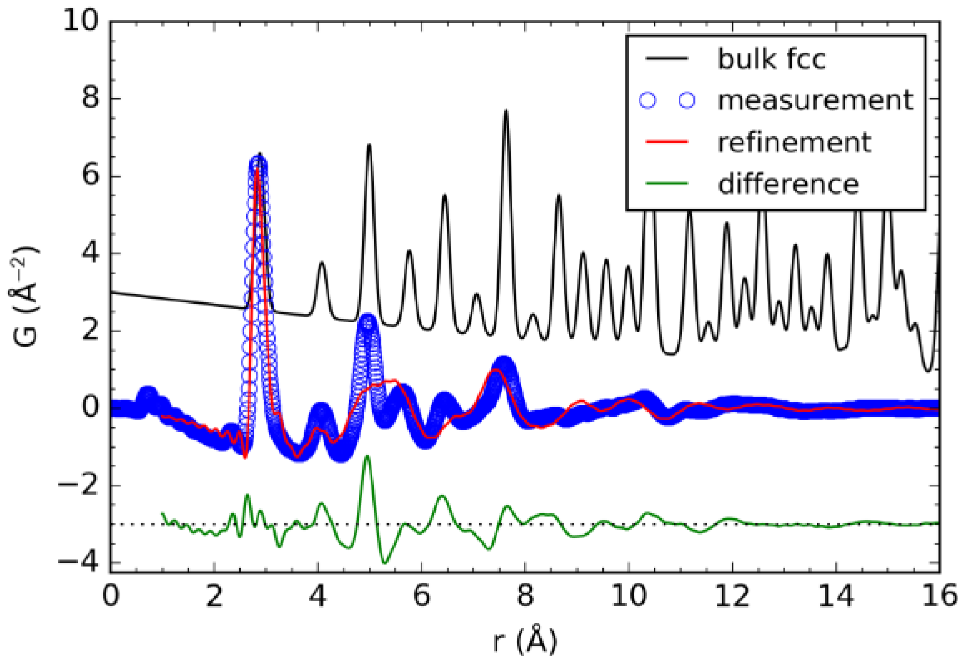
BROOKHAVEN NATIONAL LABORATORY

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







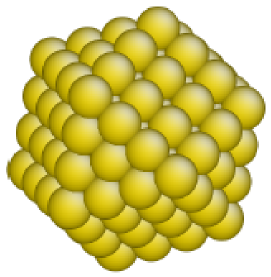


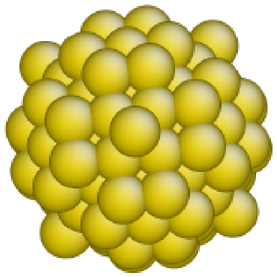
- $Au_{144}(SR)_{60}$ is ultra-stable (magic size), easy to prepare nanocluster system
- does not crystallize, published NMR+DFT studies claim icosahedral structure [Bahena *et al.*, J. Phys. Chem. Lett., 4, 975-981 (2013)
- $Au_{144}(p\text{-}MBA)_{60}$ produced by Prof. Ackerson group, Colorado State 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.

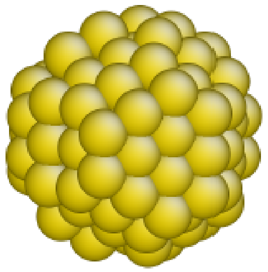
K. Jensen and P. Juhas *et al*,
Nat. Commun., 7:11859 (2016)

BROOKHAVEN
NATIONAL LABORATORY

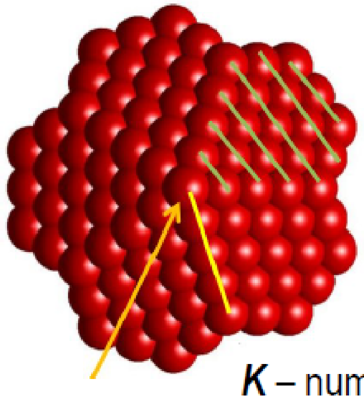
Polymorphism in $Au_{144}(SR)_{60}$


Au_{147} **cuboctahedron**


Au_{141} ***fcc***


Au_{147} ***hcp***

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

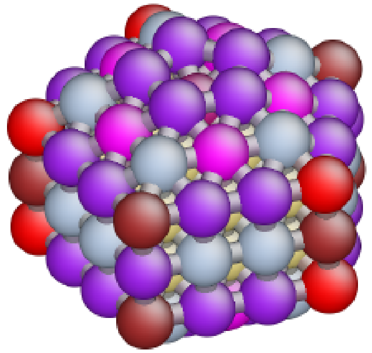

M – number of {002} shells

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

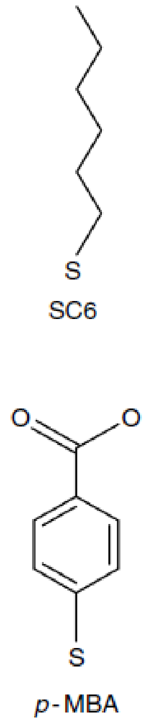
K – number of columns along the twin boundary

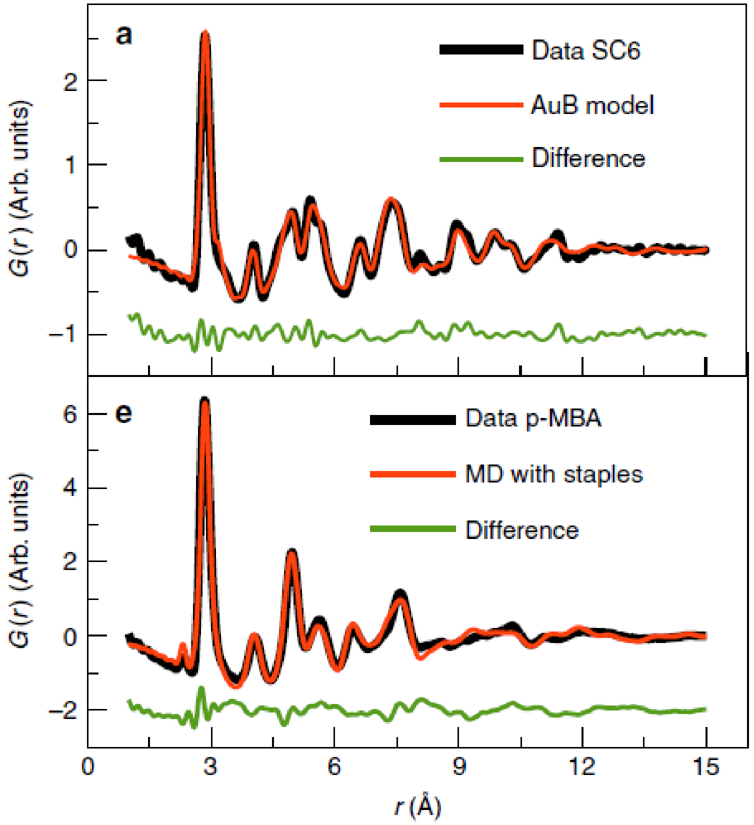
N – atoms in the central column

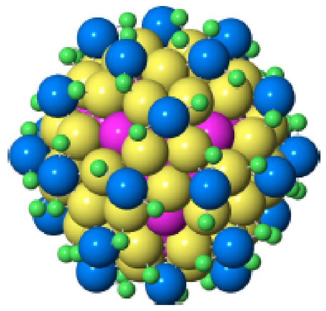
- DiffPy-CMI used for PDF refinement of gold clusters (expansion ratio, isotropic displacements for core and surface atoms)
- PDF fit evaluated for simple closed packed structures and a series of generated Marks decahedra (MD)
- very good PDF fit for 144-atom MD6441. Close PDF fit also after stripping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

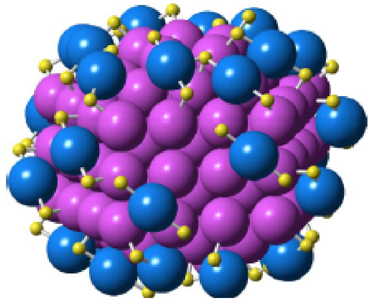


Polymorphism in $Au_{144}(SR)_{60}$



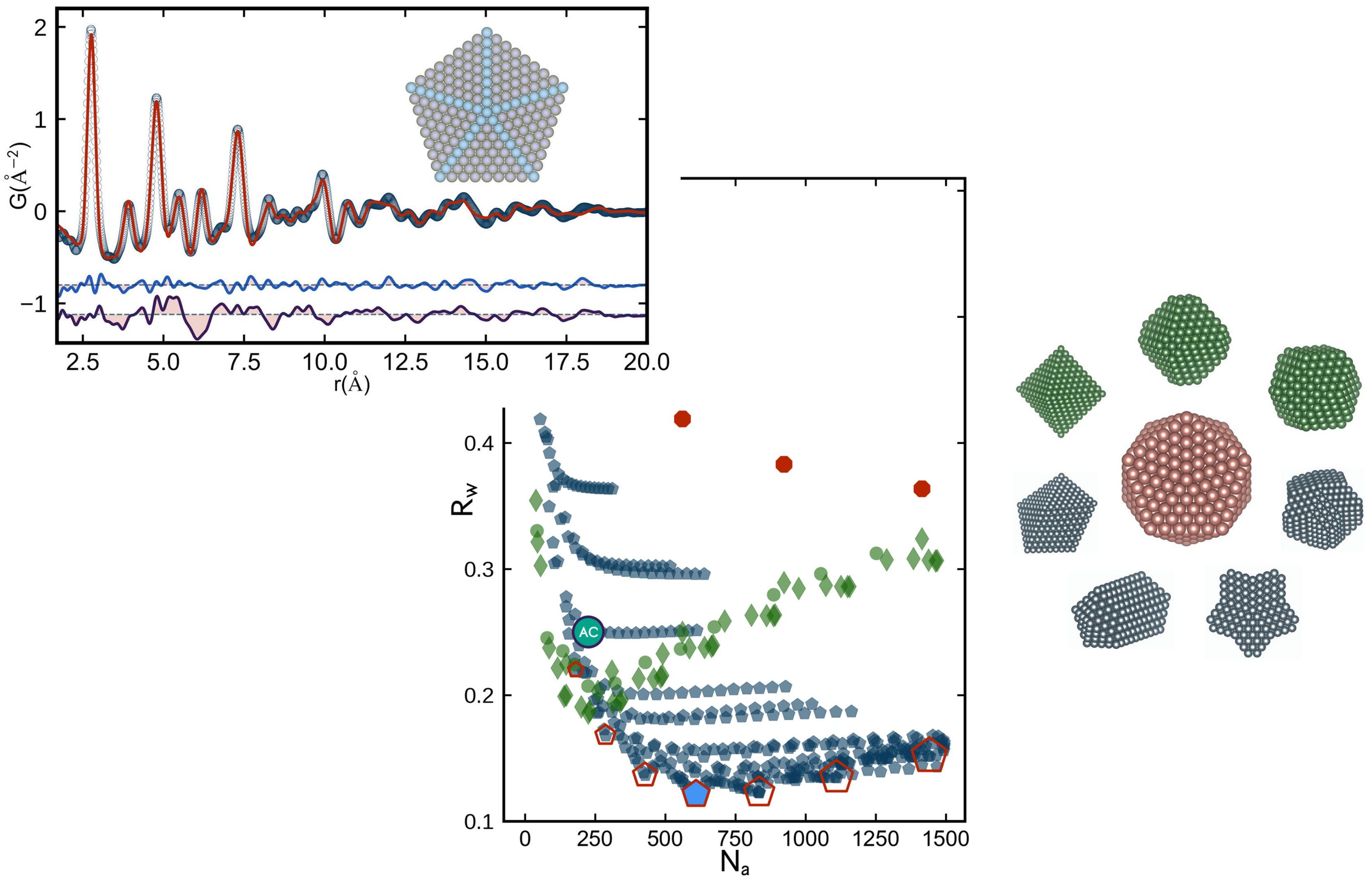




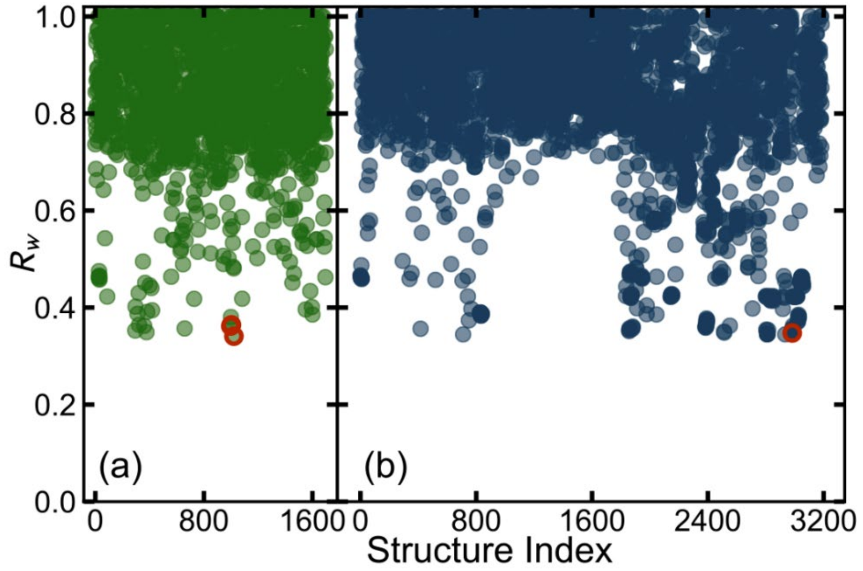


- PDF has not enough resolution to determine surface structure, i.e., staples-ligand placement. The 114-atom MD core is required for good fit.
- follow-up PDF measurements on $Au_{144}(SR)_{60}$ samples produced with different ligands showed the previous icosahedral phase and a mixture of icosahedral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

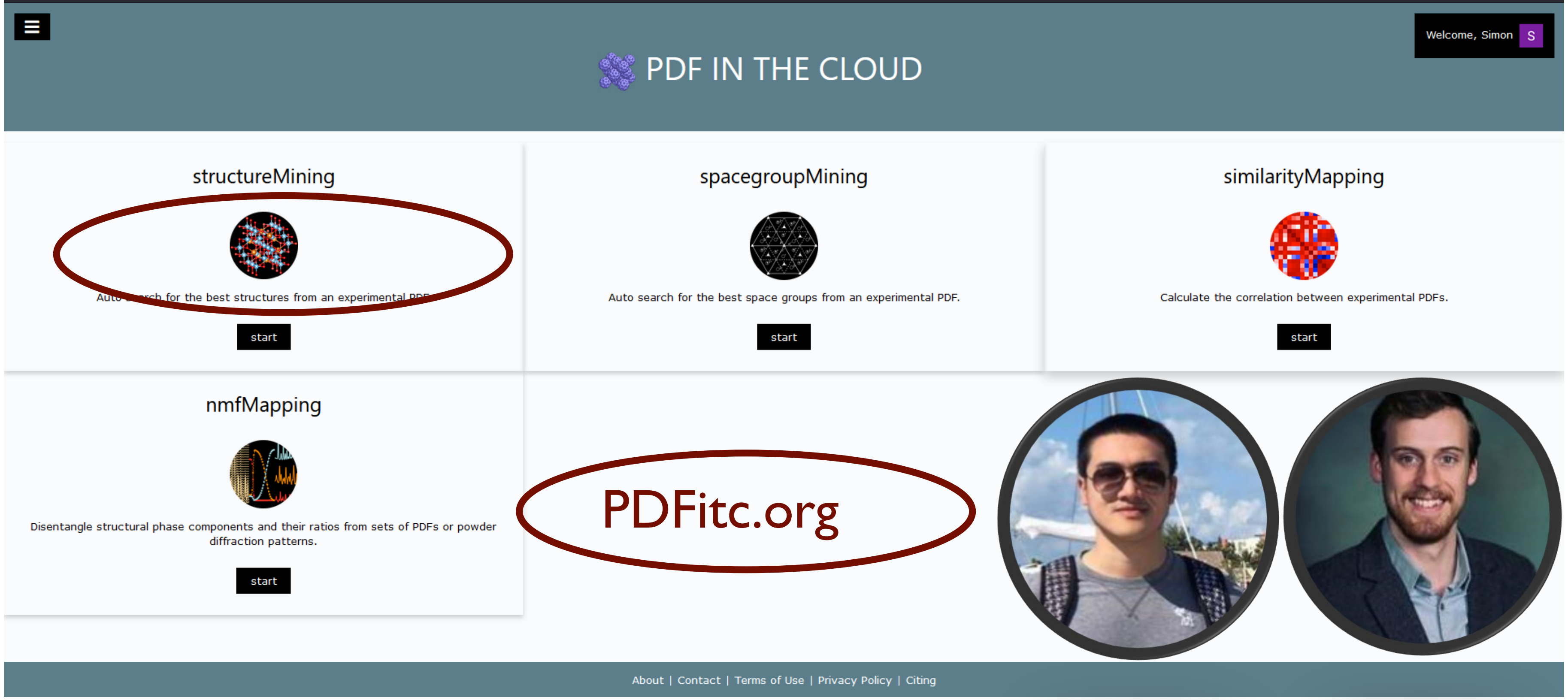
K. Jensen and P. Juhas *et al*, Nat. Commun., 7:11859 (2016)


G(Å^{-2})
2
1
0
−1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
r(Å)
R_w
0.4
0.3
0.2
0.1
AC
0
250
500
750
1000
1250
1500
N_a

With databases you can directly do data mining: structureMining



- Upload a PDF and some basic compositional information and search for close matches, e.g., $NaFeSi_2O_6$ nanowire date
  - Structure-mining found the same model as in prior work, MPD No. 1003 ($NaFeSi_2O_6$) and COD No. 2983 ($NaFeSi_2O_6$), s.g.: C 2/c.
  - It also returns some structures with space group C 2, such as MPD No. 998 ($Na_{0.83}FeSi_2O_6$), which may be viewed as a very similar structure but with a lowered symmetry and deficient atoms at some sites
  - It also returns some structures substituting at Na or Fe sites by other elements. For example, MPD No. 1021 ($NaGaSi_2O_6$).


Welcome, Simon S

PDF IN THE CLOUD

structureMining

Auto search for the best structures from an experimental PDF.

start

spacegroupMining

Auto search for the best space groups from an experimental PDF.

start

similarityMapping

Calculate the correlation between experimental PDFs.

start

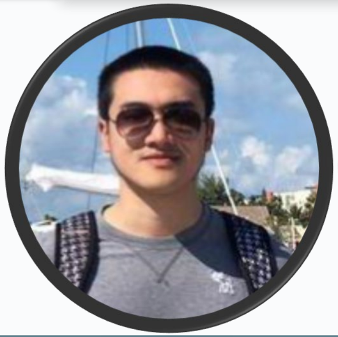
nmfMapping

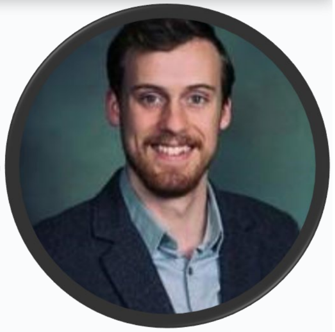
Disentangle structural phase components and their ratios from sets of PDFs or powder diffraction patterns.

start

About | Contact | Terms of Use | Privacy Policy | Citing

PDFitc.org

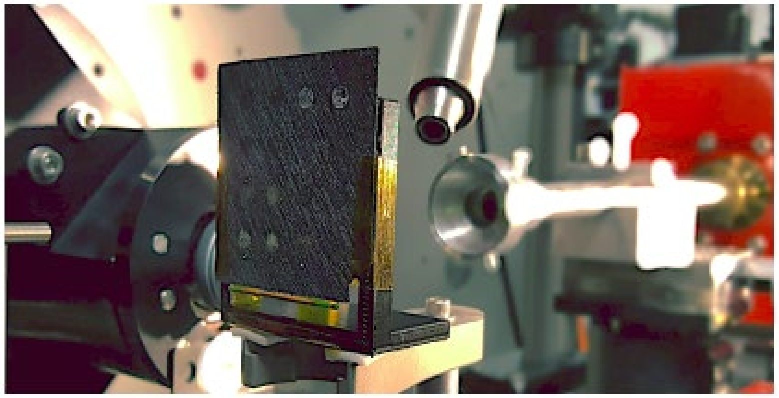


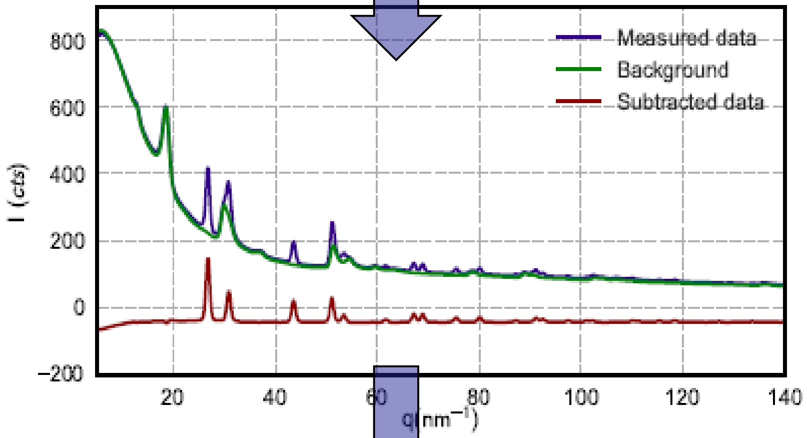


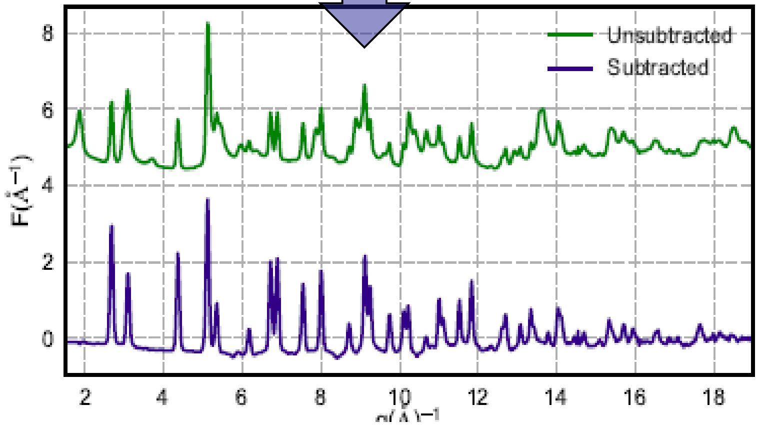
- Web platform developed as part of an NSF funded project, but nmfMapping App is a GENESIS product (Long Yang and Zach Thatcher)
- Upload a set of data (powder diffraction or PDF), get back the structure/space-group/NMF components and weights/Pearson matrix

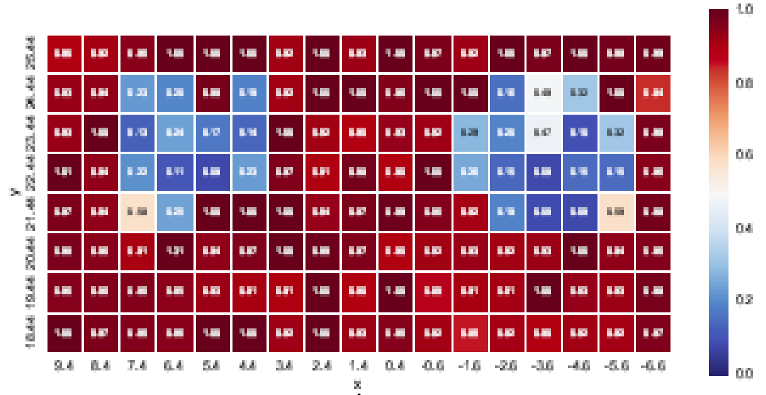
Spatially Resolved PDFs

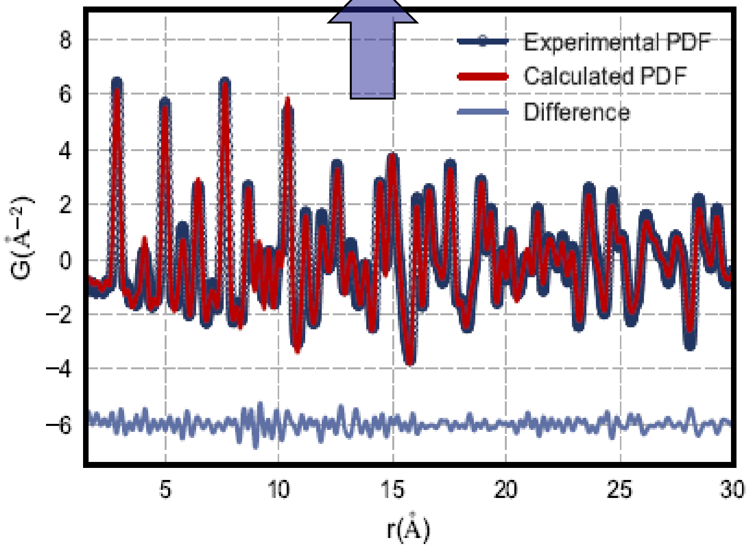
- Anton Kovyakh, Soham Banerjee, Chia Hao Liu, Tom Mallouk











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

Installation announcement

Summary

- PDF continues to grow in popularity
  - More and more materials of interest are nanostructured or amorphous
- PDF continues to become more powerful with faster measurements
  - In situ, spatially resolved etc.
- This raises issues with data handling and modeling. These are being addressed with high throughput analysis and modeling methods

Acknowledgements

- A special thank you to all my current and former students and post-docs
- Facility beamline and software teams
- Also my many wonderful collaborators, mentioned during the talk
- Facilities:
  - APS, CHESS, NSLSII (and people therein)
  - MLNSC, ISIS, SNS (and people therein)
- Funding: DOE-BES and NSF-DMR

Science in the Zoom times