



JOURNAL OF  
APPLIED  
CRYSTALLOGRAPHY

**Volume 51 (2018)**

**Supporting information for article:**

**Differential evolution and Markov chain Monte Carlo analyses of layer disorder in nanosheet ensembles using total scattering**

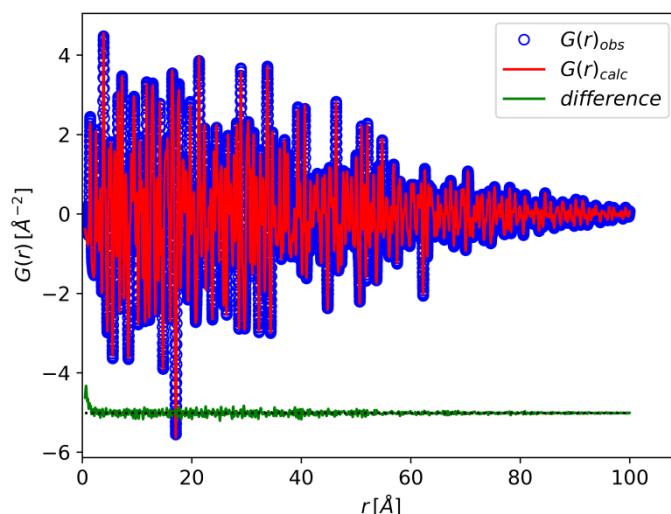
**Peter C. Metz, Robert Koch and Scott T. Misture**

## S1. Instrumental damping

The instrumental damping of the computed PDF was refined from *DIFFaX* data computed with no stacking disorder or size broadening. All data were convolved with a Gaussian instrumental broadening function of constant width in Q-space, with  $\sigma = 0.02 \text{ \AA}^{-1}$ .

**Table S1** Refined instrumental damping coefficient.

$q_{damp}$ [ $\text{\AA}^2$ ]	<i>scale</i> [-]	<i>a</i> [ $\text{\AA}$ ]	<i>c</i> [ $\text{\AA}$ ]
0.011 5 (1)	0.228 9 (7)	2.500 007 (6)	3.000 01 (7)



**Figure S1** Refinement of the instrumental damping envelope from the standard, non-disordered computed data ( $R_{\text{wp}} = 4.5\%$ ).

## S2. DIFFaX input files

### S2.1. Disordered

*DIFFaX* input file used to generate elastic X-ray scattering data.

```

INSTRUMENTAL
X-RAY {rad type}
0.2114 {rad wvl}
GAUSSIAN 0.04 {empirical broadening}
STRUCTURAL
2.5000 2.5000 3.0000 120.0000
-1 {lowest Laue group}
1 {nlayers}
infinite {lateral}
LAYER 1
None {assumed layer symmetry (?)}
{type # x y z Biso occ}
C 1 0.3333 0.6666 0.5000 0.5000 1.000
C 2 0.6666 0.3333 0.5000 0.5000 1.000
STACKING
recursive
20 {mean column length}
TRANSITIONS
{alpij Rxj Ryj Rzj (clm)}
0.25 0.25 1.0 (0.5 0.5 0.0 0.0 0.0 0.0) {1-1}

```

### S2.2. Ordered

Identical, except:

```

TRANSITIONS
{alpij Rxj Ryj Rzj (clm)}
0.0 0.0 1.0 (0.0 0.0 0.0 0.0 0.0 0.0) {1-1}

```

## S3. Generation of sample vectors

*Python* code used to generate sample vectors on a set of concentric rings.

```

import numpy as np

def points(r, nr, ntheta, center):
    """
    get ntheta * nr evenly spaced concentric points
    - r (float): radius of space to sample from
    - nr (integer): number of radial slices
    - ntheta (integer): number of angular slices
    - center (tuple): fractional coordinates (Rx(frac.), Ry(frac.)) as tuple
    """
    rv = []

    # theta array
    t = [2 * i * np.pi / ntheta for i in range(1, ntheta + 1)]

```

```

# for angle in theta array at radius intervals dr
for i in range(nr):
    R = r / nr * i
    l = np.array((R * np.cos(t), R * np.sin(t))).T # get points

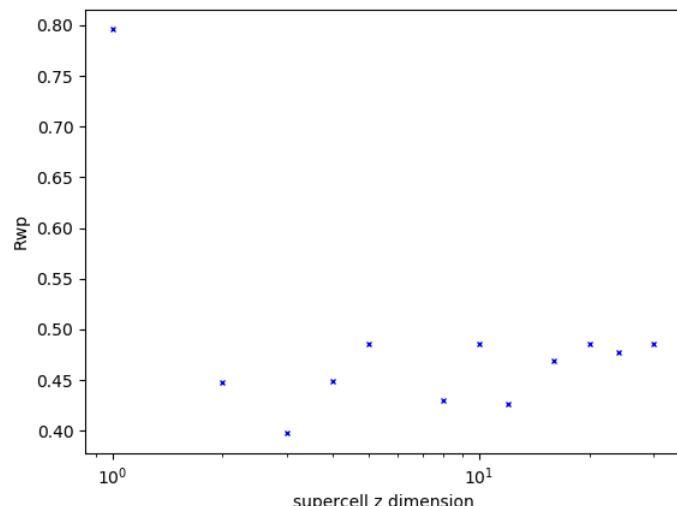
    # throw out duplicates
    t1 = lambda l: l == l[0, :]
    t2 = lambda l: l == 0

    if t1(l).all() or t2(l).all() is True: # if all identical, keep only first
        l = l[0]
        rv.append(l.tolist())
    else:
        rv.extend(l.tolist())

# return nicely reshaped array
rv = np.array(rv)
return rv + center

```

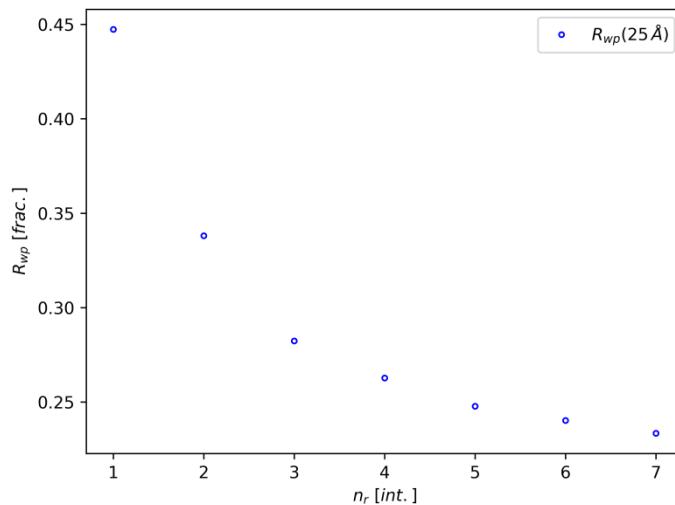
#### S4. Influence of supercell dimension



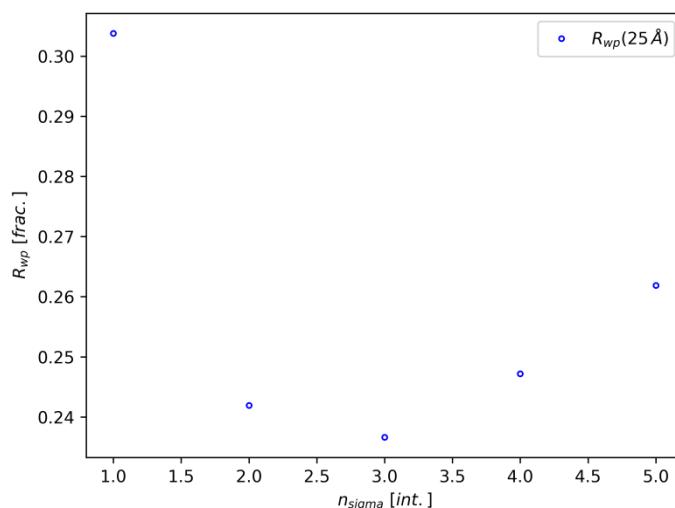
**Figure S2**  $R_{wp}(O)$  shows minima on supercells of dimension  $M \times N \times O$  which introduce no boundary errors.

### S5. Influence of model size (sampling)

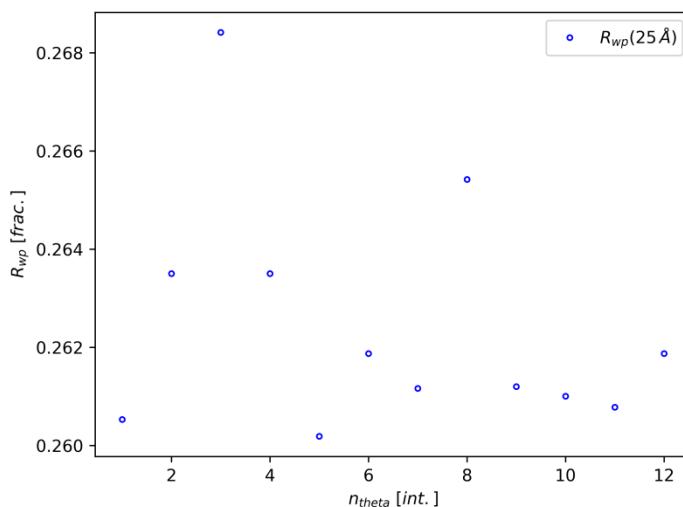
Each sampled configuration is weighted by a 2D standard normal distribution. In these examples  $\sigma_y = \sigma_x = \sigma$ , and  $\rho = 0$ . Supercell dimensions are 1x1x24 for all following figures.



**Figure S3** Influence of radial sampling ( $n_{\text{theta}} = 6$ ,  $\sigma = 0.03$  [/],  $r = 0.1$  [/])



**Figure S4** Influence of radius of sampling in integer multiples of the standard deviation ( $\sigma = 0.03$  [/],  $n_r = 6$ ,  $n_{\text{theta}} = 6$ )



**Figure S5** Influence of angular sampling ( $r = 0.1$  [/],  $n_r = 2$ ,  $\sigma = 0.03$  [/])

## S6. Computational resources

The differential evolution refinements described in this paper were performed on an ASUS PC equipped with an Intel Core i5-3550 CPU @ 3.30 GHz.

MCMC sampling, which requires computation of 10's to 100's of thousands of PDF, was parallelized using *MPI4Py* (Dalcín *et al.*, 2005; M. Price-Whelan & Foreman-Mackey, 2017). Generations of 120 – 160 population members were sampled over hundreds of iterations, where PDF calculations in each generation were distributed over a matching number of threads.

In the case of the constrained 31 layer-type model fit over 75 Å, individual PDF computations required of the order of 5 minutes. Generation of the corresponding MCMC chain required approximately  $10^4$  cpu-hours.

## References

- Dalcín, L., Paz, R., & Storti, M. (2005). *J. Parallel Distrib. Comput.* 65, 1108–1115.  
 M. Price-Whelan, A. & Foreman-Mackey, D. (2017). *J. Open Source Softw.* 2, 10–11.