Supporting Information

Data Collection and Reduction

X-ray total scattering data were collected on an amorphous silicon area detector (Varex 4343CT) at beamline 6-ID-D at the Advanced Photon Source. Data for liquid water were acquired at 300K using 0.153394 Å x-rays at an approximate sample to detector distance of 270 mm. Data for molten sulfur were acquired at 453K using 0.123696 Å x-rays at an approximate sample to detector distance of 300 mm. Appropriate background data were acquired for each sample under complementary conditions. CeO2 (NIST standard 676a) was used to calibrate the sample to detector distance, beam center, detector tilt and rotation in GSAS-II. Masking of bad pixels and integration of the diffraction images were performed in GSAS-II.

Equations from Main Text

1. $f_\alpha(Q) = f_\alpha^0(Q)\left[1 - \frac{a_O}{z_\alpha} e^{\left(-Q^2/2\delta^2\right)}\right]$
2. $S(Q) - 1 = \frac{I(Q) - \sum_\alpha c_\alpha f_\alpha^2(Q) - C(Q)}{\left[\sum_\alpha c_\alpha f_\alpha(Q)\right]^2}$
3. $G(r) - 1 = \frac{1}{2\pi^2\rho}\int_0^{Qmax} Q[S(Q) - 1]\frac{sin(Qr)}{r} M(Q)dQ$
4. $D(r) = 4\pi\rho r[G(r) - 1]$
5. $S(Q = 0) - 1 = \frac{\rho\chi k_B T\left[\sum_\alpha c_\alpha f_\alpha(0)\right]^2 - \sum_\alpha c_\alpha f_\alpha^2(0)}{\left[\sum_\alpha c_\alpha f_\alpha(0)\right]^2}$
6. $n_{\alpha\beta} = \int_{r1}^{r2} 4\pi\rho r^2 c_\beta g_{\alpha\beta}(r)dr$
7. $G(r) = \frac{2}{\pi}\int_0^{Qmax} Q[S(Q) - 1]sin(Qr)M(Q)dQ$ (Billinge/Egami formulation)
8. $F(Q) = Q[S(Q) - 1]$
9. $n = \frac{Q_{maxinst} r_{poly}}{\pi}$

GudrunX Extended Discussion

GudrunX was developed as an x-ray version of an earlier software, Gudrun, which was originally created for obtaining structure factors and PDFs from neutron scattering data. GudrunX is primarily designed for analysis of scattering data from laboratory x-ray sources, so it provides many additional parameters specific to those sources, but the program is also applicable for synchrotron x-ray data [1]. Prior to calculation of structure factors and pair distribution functions with GudrunX, 2θ-binned I(Q) data were manually corrected for detector attenuation and oblique incidence [2], since these corrections are not available in GudrunX. Then in GudrunX, the workflow is managed through a GUI divided into tabs for Instrument, Beam, Normalisation, Background, and Sample. Under the Instrument tab, the Q range was set as 0.45-20 Å^{-1} for both water and sulfur. The Beam tab is used to define the X-ray source. Sample geometry was set to cylindrical, and the beam edges were defined corresponding to a 0.5 X 0.2 mm beam centered on the sample. Parameters for bremsstrahlung, pertinent for a lab-based X-ray source, were not used. In the Normalisation tab, the azimuthal angle of the detector was set to 0, normalization was set to <F>^2, a Breit-Dirac factor 2 was used, and Krogh-Moe & Norman normalization was enabled with an overlap factor of 0. For the Background tab, background scaling was set to 1. On the Sample tab, mass densities were set to 1.0 and 1.8 g/cm^3 for water and sulfur, respectively. Factors to modify the multiple and Compton scattering were both set to 1. Fluorescence corrections were applied by entering the K-edge energies for either hydrogen and oxygen (water), or for sulfur. The overall fluorescence level, which represents how much fluorescence contributes to I(Q), was manually adjusted to optimize the fit of I(Q) to the self-scattering. Incident beam polarization was set to -1, which, along with the detector azimuthal angle of 0, results in no fluorescence correction. (Fluorescence was already corrected in earlier steps with Fit2D).

In the Sample tab, GudrunX also offers parameters for the so-called "top hat" convolution [3], a method used to remove any residual long-wavelength background in S(Q), which may be present due to insufficient or inaccurate data corrections (e.g., large multiple scattering effects for highly x-ray absorbing samples). The top hat convolution uses two parameters: (1) r_{min}, which is set to be smaller than the first atomic pair correlation in real-space, and (2) Q_T, which is the threshold for which frequencies are to be suppressed in S(Q) to remove the residual background. GudrunX provides output files before and after the top hat convolution is applied in the *.soq and *.int01 files, respectively. Ideally, the top hat convolution is not necessary, and the S(Q) obtained before and after the top hat will be nearly identical. However, if residual long-wavelength background cannot be removed by tweaking the other corrections (e.g., Compton scattering, multiple scattering, or fluorescence), the top hat is a final option adjust S(Q) so its high-Q baseline is flat. It is absolutely essential that the software user compare the S(Q) before and after the top hat function to ensure it has been applied appropriately and with reasonable parameters (as defined in the manual), otherwise the top hat may distort the data. For water, Q_T was set at 5.0 $Å^{-1}$ and r_{min} was 0.7 Å. For sulfur, Q_T was set at 2.5 $Å^{-1}$ and r_{min} was 1.4 Å.

After the top hat settings, two parameters are used to define the revised Lorch function that can optionally be applied during the Fourier transform to obtain the pair distribution function. (The original Lorch function is not provided as an option in GudrunX.) The width of broadening was set to 0.1 Å, and the broadening power was 0. These parameter values were chosen so that the revised Lorch function produced a pair distribution function similar to that obtained from a Fourier transform with a conventional Lorch function, which was calculated in a separate program. The Fourier transforms of the *.soq and *.int01 files are provided in the *.gr1 and *.gofr files, respectively. For the analysis of water and sulfur, the *.int01 and *.gofr files (after top hat correction) were used.

It is important to note that the GudrunX outputs for structure factors (*.soq and *.int01) are actually S(Q) – 1, and the pair distribution functions (*.gr1 and *.gofr) are actually G(r) – 1. For comparison to other PDF extraction packages, *D(r)* was back-calculated from the *.gofr files by rearranging [eq 4].

Extracting g(r) from GSAS-II

Minimal code required for export of g(r) data from GSAS-II, which is not currently implemented in the GUI as of writing this review.

```
import os,sys,glob,re #load these modules for all my code
import numpy,scipy #need numpy for sure, loaded scipy just in case
sys.path.insert(0,"C:\\[path-redacted]\\gsas2full\\GSASII") # needed
to "find" GSAS-II modules, should be actual path to GSAS-II python
files
import GSASIIscriptable as G2sc #need this for GSAS-II functions
```

```
from PIL import Image #need this to read in images and save them
from matplotlib import pyplot as plt #plotting utility
from matplotlib.path import Path
import csv

#def __reset__(): get_ipython().magic('reset -sf') #not called in
function, used to reset in interactive python session

datadir = '[path-redacted-2]'
os.chdir(datadir) #set working directory to current directory
dataoutdir = os.getcwd() ##use current working directory

gpx = G2sc.G2Project(gpxfile=os.path.join(datadir,"project.gpx"))

pdflist=gpx.pdfs()

r_GSASII = pdflist[0].data['PDF Controls']['g(r)'][1][0]

g_r_GSASII = pdflist[0].data['PDF Controls']['g(r)'][1][1]

# fig, ax = plt.subplots()
# plt.plot(r_GSASII,g_r_GSASII)
# plt.xlim([0, 20])
# plt.show()

with open('samplename.gofr', 'w') as f:
    writer = csv.writer(f, delimiter=',')
    writer.writerows(zip(r_GSASII, g_r_GSASII))
```

Scripting pair distribution function extraction in PDFgetX2 via IDL and Python

A significant barrier to use of PDFgetX2 is the inability to natively batch process large datasets. Per the PDFgetX2 manual, PDFs can be batch extracted using IDL; however, this functionality is not well documented. Below are some minimal examples of IDL and python code that leverages the IDL/Python bridge module. The full (PRO) version of IDL is required for this functionality. Python can be used to copy an existing PDFgetX2 history file to a new file and update the relevant filenames and scale factors. This can effectively allow for a sequential optimization and extraction of PDFs.

1. IDL code

```
installdir = 'PDFgetX2_installation_directory'
restore, installdir + 'pdfgetx2.sav'

workingdir = 'directory with files'

cd, workingdir

getxpdf = Obj_New('GetXPDFData')
```

The pdfgetx2 commands below have been successfully run in IDL, where *element* is the name of a given history file:

```
      getxpdf->loadhistory, element
      getxpdf->resetalldata
      getxpdf->readdetfile
      getxpdf->getiq
      getxpdf->calcsqcorrections
      getxpdf->getsq
      getxpdf->optimizesq
      getxpdf->getsq
      getxpdf->getgr
      getxpdf->setupvisualization, create_struct('datatype', 'GrData')
      getxpdf->plotdata
      WAIT, 5
      getxpdf->savedata, SQFILE=STRING(element)+'.sq',
GRFILE=STRING(element)+'.gr'
      getxpdf->savehistory, element
```

These can be embedded in a loop in IDL:

```
foreach element, elementlist do begin
    IDL command
    Another IDL command
endforeach
```

2. Python/IDL bridge

```
from idlpy import *

IDL.run("print, 'some string'", stdout=1)  #print 'some string' to
standard output
IDL.variablename=variablename #pass variable from python to IDL

IDL.run("installdir = ' 'PDFgetX2_installation_directory'",
stdout=1) #directory where PDFgetx2 is installed
IDL.run("restore, installdir + 'pdfgetx2.sav'", stdout=1) # load
pdfgetx2 code into IDL

#IDL.run("pdfgetx2", stdout=1) #launch PDFgetx2 directly

IDL.run("workingdir = 'directory with files'", stdout=1) #set
working directory to directory with files

IDL.run("cd, workingdir", stdout=1) #move to file directory

IDL.run("getxpdf = Obj_New('GetXPDFData')", stdout=1) #create a new
object with pdfgetx2 functions, procedures, structures, etc.
```

```
IDL.element=element #pass history file name to IDL
IDL.run("getxpdf->loadhistory, element", stdout=1) #load history
file

    IDL.run("getxpdf->resetalldata", stdout=1) #reset data button in
pdfgetx2
    IDL.run("getxpdf->readdetfile", stdout=1) #read in data
    IDL.run("getxpdf->getiq", stdout=1) #calculate I(Q)
    IDL.run("getxpdf->calcsqcorrections", stdout=1) #calculate S(Q)
corrections
    IDL.run("getxpdf->getsq", stdout=1) #get S(Q)
    IDL.run("getxpdf->optimizesq", stdout=1) #optimize S(Q)
    IDL.run("getxpdf->getsq", stdout=1) #get S(Q) again
    IDL.run("getxpdf->getgr", stdout=1) #get G(r)--
crystallographers' G(r), glass community's D(r)
    IDL.run("getxpdf->setupvisualization, create_struct('datatype',
'GrData')", stdout=1) #prepare to plot G(r)
    IDL.run("getxpdf->plotdata", stdout=1) #plot G(r)
    IDL.run("WAIT, 5", stdout=1) #pause 5 seconds so the plot can be
viewed
    IDL.run("getxpdf->savedata, SQFILE=STRING(element)+'.sq',
GRFILE=STRING(element)+'.gr'", stdout=1) #export S(Q) and G(r)
    IDL.run("getxpdf->savehistory, element", stdout=1) #save history
file
```

References

1. Soper, A.K., *GudrunN and GudrunX.* Programs for correcting raw neutron and x-ray diffraction data to differential scattering cross section, 2010.
2. Skinner, L.B., C.J. Benmore, and J.B. Parise, *Area detector corrections for high quality synchrotron X-ray structure factor measurements.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2012. **662**(1): p. 61-70.
3. Soper, A.K., *Inelasticity corrections for time-of-flight and fixed wavelength neutron diffraction experiments.* Molecular Physics, 2009. **107**(16): p. 1667-1684.