JOURNAL OF SYNCHROTRON RADIATION
ISSN 1600-5775

pyXPCSviewer: an open-source interactive tool for X-ray photon correlation spectroscopy visualization and analysis

Miaoqi Chu,* Jeffrey Li, Qingteng Zhang, Zhang Jiang, Eric M. Dufresne, Alec Sandy, Suresh Narayanan and Nicholas Schwarz

X-ray Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA. *Correspondence e-mail: mqichu@anl.gov

Received 11 April 2022
Accepted 5 May 2022

Edited by V. Favre-Nicolin, ESRF and Université Grenoble Alpes, France

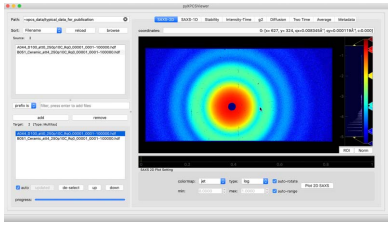
Keywords: X-ray photon correlation spectroscopy; synchrotron; visualization; GUI; Python.

pyXPCSviewer, a Python-based graphical user interface that is deployed at beamline 8-ID-I of the Advanced Photon Source for interactive visualization of XPCS results, is introduced. *pyXPCSviewer* parses rich X-ray photon correlation spectroscopy (XPCS) results into independent *PyQt* widgets that are both interactive and easy to maintain. *pyXPCSviewer* is open-source and is open to customization by the XPCS community for ingestion of diversified data structures and inclusion of novel XPCS techniques, both of which are growing demands particularly with the dawn of near-diffraction-limited synchrotron sources and their dedicated XPCS beamlines.

1. Introduction

X-ray photon correlation spectroscopy (XPCS), whose underlying principles are derived from photon correlation spectroscopy (PCS), is a coherent X-ray scattering technique enabled by high-brilliance undulator sources as it relies on the spatial coherence of the X-ray beam to generate far-field interference patterns ('speckles') in the scattered beam intensity. In addition to the structural information $I(Q)$ encoded in X-ray scattering (*e.g.* small-angle X-ray scattering, SAXS), where I is the scattering intensity and Q is the momentum transfer, XPCS records $I(Q, t)$ of intensity speckles over time t. As a result, XPCS provides direct measurement of system dynamics via the temporal intensity correlation function, *e.g.* $g_2(\tau, Q) = \langle I(t, Q) I(t + \tau)\rangle_t / \langle I^2(t, Q)\rangle_t$. For the simplest XPCS measurements, the beam on the sample is fixed and the statistics of g_2 can be greatly enhanced by binning detector pixels over Q and averaging over t assuming that the dynamics are stationary during intensity collection (Dierker *et al.*, 1995). For more details about XPCS theories and experimental setup, readers can refer to Grübel *et al.* (2008) and Shpyrko (2014).

The versatility of XPCS, combined with full compatibility with *in situ* sample environments and wide temporo-spatial coverage, has led to unique insight into the collective atomic motion in metallic glasses (Evenson *et al.*, 2015; Giordano & Ruta, 2016) and supercooled liquid alloys (Ruta *et al.*, 2020), arrested dynamics of nanoparticles in polymer matrices (Dierker *et al.*, 1995; Senses *et al.*, 2017; Yavitt *et al.*, 2021), liquid–liquid phase separation in protein (Begam *et al.*, 2021) and micelle suspensions (Sheyfer *et al.*, 2020), and the mesoscopic structure dynamics in both abrasive manufacturing such as ion-beam patterning (Myint *et al.*, 2021) and additive manufacturing such as 3-D printing (Lin *et al.*, 2021), where



the results common to all these XPCS applications are intensity correlation functions in the form of multi-dimensional spectra over time and reciprocal space. However, generating and interpreting XPCS results can be very challenging due to the multi-dimensional and statistical nature of the data that require efficient reduction and sophisticated visualization tool suites. This challenge will be further heightened by more sophisticated XPCS analysis including the higher-order (Perakis *et al.*, 2017; Dallari *et al.*, 2020) and higher-dimensional (Sutton *et al.*, 2021) correlation functions enabled by the orders-of-magnitude increases of coherent flux from emerging near-diffraction-limited synchrotron sources around the world such as APS-U (Shi *et al.*, 2017), ESRF-EBS (Chenevier & Joly, 2018), PETRA-IV (Schroer *et al.*, 2018) and MAX-IV (Martensson & Eriksson, 2018). A suite of expandable, customizable and user-friendly XPCS software tools is therefore essential for advancing the scientific impact of XPCS.

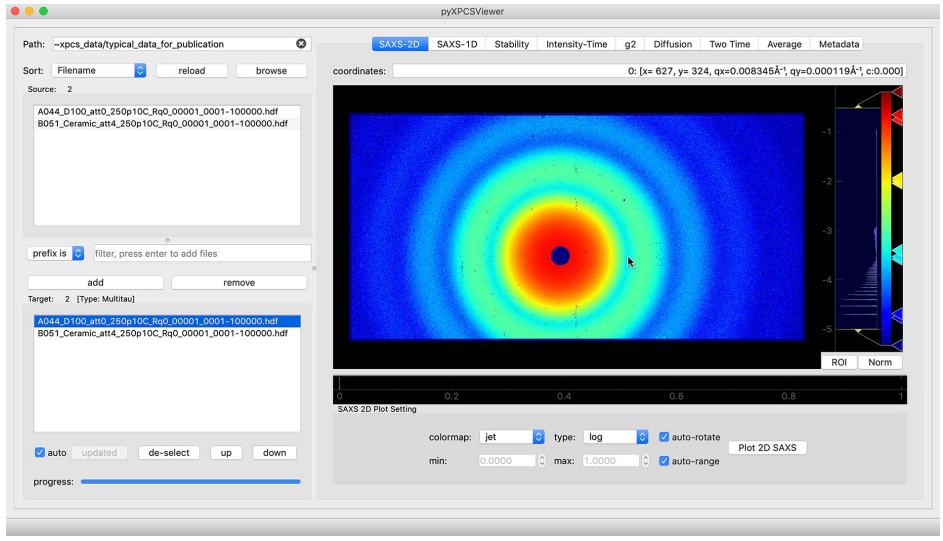


Figure 1
Layout and modules for *pyXPCSviewer*. In the left panel, users can set or reload the working directory. The data files will be displayed in the source box with the option to be sorted and filtered. Files can be added, removed and sorted in the target box that lists the files for visualization. The right panel contains the visualization modules, organized in tabs — a 2-D scattering pattern is displayed in this figure.

A Matlab-based graphical user interface (GUI) for visualizing XPCS results, *XPCSGUI* (Sikorski *et al.*, 2011), is currently deployed at APS 8-ID. However, due to the limitation of the Matlab proprietary platform, a growing amount of software at synchrotron X-ray facilities is being developed using open-source tools (Rio & Rebuffi, 2019), with the existing Matlab GUIs either transitioning to Python (Rogers *et al.*, 2019) or having supplementary Python versions (Lou *et al.*, 2021; Schirmer & Althaus, 2020; Zhang *et al.*, 2014). A Python-based GUI for XPCS will not only benefit significantly from the foundation already laid by the community but will, we hope, also become an organic component for the ongoing development of an open-source Python-based ecosystem at multiple synchrotron facilities, including the Python-based beamline control system (Bluesky) at NSLS-II (Arkilic *et al.*, 2017) and the Data Management workflow at APS (Veseli *et al.*, 2018).

2. Package structure

pyXPCSviewer follows the standard Model-View-Controller (MVC) pattern (Gamma *et al.*, 1995) for quick development and easy maintenance. The GUI takes users' controls (mouse click/movement) and inputs and displays the images and plots, while *pyXPCSviewer*'s kernel (Model) fetches and processes the data for display.

pyXPCSviewer is written in Python so it can run easily on major platforms, including Windows, MacOS and Linux. The GUI functions are enabled by a combination of *PyQt* (Riverbank Computing, 2022), *PyQtGraph* (Campagnola, 2022) and *Matplotlib* (Hunter, 2007) that provide performance, interactivity and ready customization. The compute-intensive operations of *pyXPCSviewer* are performed with *NumPy* (Harris *et al.*, 2020) and *SciPy* (Virtanen *et al.*, 2020). The code and installation guide are available on GitHub (Chu, 2021)

Following standard practices optimized over many years of operations at 8-ID-I at the APS, the workflow of *pyXPCSviewer* is divided into the following six groups: (i) dataset selection and loading, (ii) inspection of scattering patterns, (iii) assessment of sample beam damage and beamline stability, (iv) visualization and modeling of multi-tau correlations, (v) visualization of two-time correlations, and (vi) helper functions. The functions in each group are organized as separate tabs in the GUI, as shown in Fig. 1, so that the information is centrally organized. Comprehensive help also accompanies each of the workflow steps.

2.1. Dataset loading

Currently, *pyXPCSviewer* supports the APS Data Exchange format based on HDF5 file structures (De Carlo *et al.*, 2014). The files are generated from XPCS-Eigen (Khan *et al.*, 2018), which computes the correlation of the raw data and combines all the metadata (such as X-ray energy, Q-map definition and beam position) as one file. Other data formats can also be loaded using an adaptor layer to map the data to the format used by *pyXPCSviewer*.

In the GUI shown in Fig. 1, users can specify the working directory interactively via the load button or as a command line argument when launching the viewer. The example data set in this manuscript was captured using XSPA-500k, a 1024 × 512 pixel array detector featuring gapless field-of-view and

a maximum continuous frame rate of 52 kHz (Nakaye *et al.*, 2021). After a valid working directory is set, the source panel lists the available files. To locate and select datasets quickly, users can sort the source files using the filename, the scan index (the first four digits of the filename at the APS) and time of creation. In addition, two filter types are included to narrow down the possibilities, namely a prefix filter and a substring filter. Once added to the target box, the datasets are loaded into the computer's RAM to accelerate subsequent operations. *pyXPCSviewer* supports loading multiple datasets allowing users to overlay the correlation curves to compare the differences in detail. This is only possible if the detector dimensions match and the Q-partition maps are the same.

2.2. Inspection of scattering patterns

The time-average scattering pattern from an XPCS dataset is obtained by averaging all frames in the time series. It contains information about the structure of the sample. *pyXPCSviewer* uses the ImageView module of *PyQtGraph* to display the 2-D scattering patterns. As shown in Fig. 1, it provides customizable colormaps, easy control over the data range and quick response during zoom in/out. By examining the displayed scattering pattern, many anomalies can be easily spotted, such as no beam, scattering streaks (parasitic scattering from slits or sample holders) or a misaligned beamstop. Such datasets can be excluded from further analysis to avoid artifacts in the correlation results or users can define a mask that excludes the abnormal regions.

In the small-angle geometry, if the sample is isotropic, it is convenient to perform azimuthal averaging of the scattering signal. Once collapsed to 1-D $I(Q)$, users can change axis types (log or linear) and apply a normalization method if needed to enhance features or contrast of the profile. *pyXPCSviewer* provides two drawing methods as shown in Fig. 2 that allow users to draw lines interactively with the mouse. The horizontal line method allows users to draw horizontal lines between peaks/valleys to estimate the particle's size, that is calculated as $\Delta_x = 2\pi/\Delta_Q$, where Δ_Q is the length of the horizontal line. The other method is slope drawing that allows users to estimate the power-law parameter of a in the asymptotic relationship $I \simeq Q^a$. More advanced modeling of SAXS data is not provided but users can export the data to a dedicated SAXS package (Ilavsky & Jemian, 2009; Petoukhov *et al.*, 2012) for in-depth analysis.

2.3. Sample and beamline stability

XPCS is a technique that probes the dynamics of materials by correlating the scattering fluctuations. In order to investigate the inherent properties of the dynamic material system, it is necessary to exclude X-ray introduced artifacts (also known as X-ray beam damage) or fluctuations from the beamline. *pyXPCSviewer* enables quick inspection of the data so users can optimize data collection protocols at the beamline to identify and eliminate such artifacts.

The sample stability is characterized by dividing a time series into several continuous segments and computing the

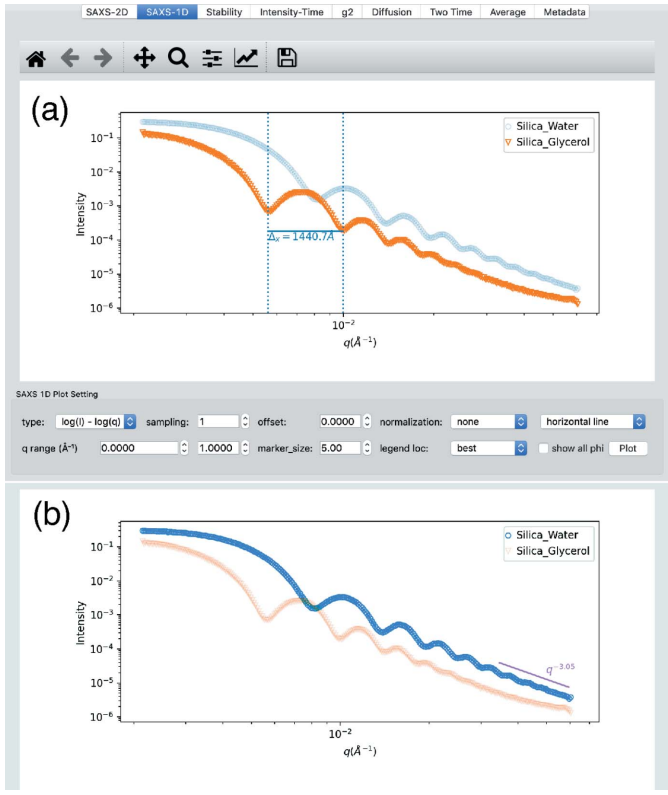


Figure 2
1-D small-angle scattering visualization. Users can draw horizontal lines (vertical dashed lines help users locate peaks/valleys) to estimate the structure size as shown in (*a*) or draw slopes to estimate the power-law decay constant as shown in (*b*).

1-D SAXS curve, $I(Q)$, in each segment, as shown in Fig. 3(*a*). The curves should overlap with each other if the measurement does not suffer from artifacts. If the curves are different, users might have to attenuate the beam to reduce the radiation dose on the sample or use a different kind of sample. Inspection of the SAXS data for radiation-induced effects will become particularly critical with the marked increase of coherent flux from the next generation of X-ray sources.

We also need to make sure that the beamline is stable during the measurement. *pyXPCSviewer* can compute simple diagnostics to aid in identifying instabilities in the sample environment or measurement conditions. In Fig. 3(*b*), we plot the averaged photon counts on the detector as a function of frame index (or time) that can be used to identify frame drops that could arise due to detector malfunction or changes in the incident beam. Users can also choose a region to zoom in to explore the fine structure, as shown in Fig. 3(*d*). The Fourier transform of Fig. 3(*b*) is shown in Fig. 3(*c*) and this can help with identifying possible vibration modes in either the beam intensity or the sample setup.

2.4. Multi-tau correlation visualization and modeling

Multi-tau correlation approximates g_2 and characterizes the dynamics of materials in equilibrium. As the structure of a material fluctuates, the speckles fluctuate correspondingly.

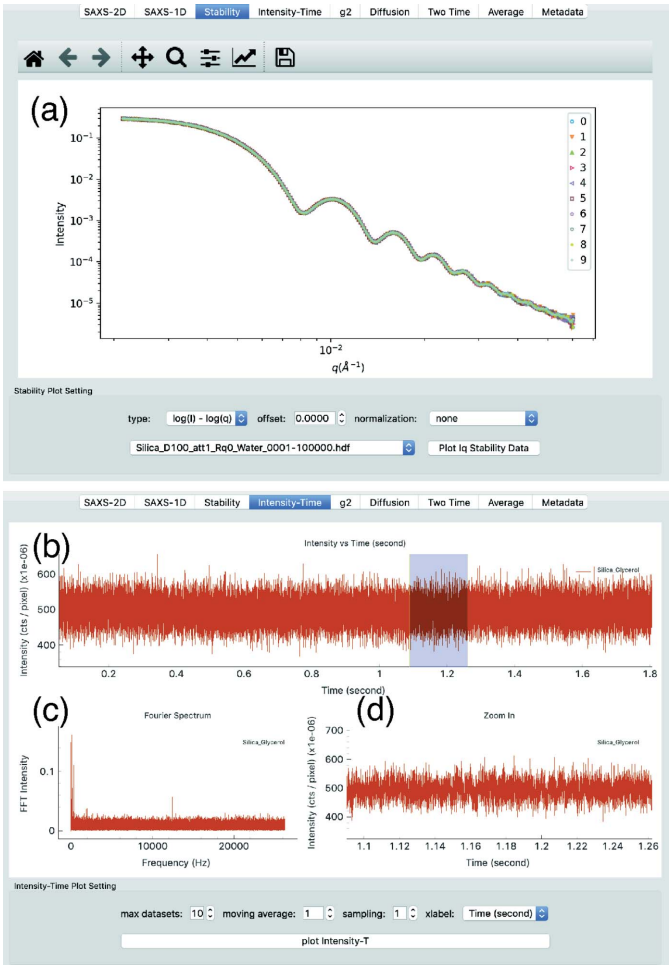


Figure 3
(*a*) Stability plot showing the SAXS 1-D $I(Q)$ curves generated from segments of the time series. In this example, the ten curves match indicating that the sample is stable during the measurement. (*b*) The intensity versus time during the XPCS measurement. (*c*) Fourier component analysis of (*b*). (*d*) Zoom-in of the region highlighted by the light blue rectangle in (*b*).

Components of different length scales exhibit different fluctuations. While each pixel on the area detector contains information about a certain length scale defined by the scattering geometry, a pixel usually does not have enough photon statistics to compute a high-fidelity correlation function. In practice, a Q-map is usually applied that groups the equivalent Q-regions to improve statistics while maintaining reciprocal space resolution.

In *pyXPCSviewer*'s g_2 tab that is shown in Fig. 4, users can set a Q range to exclude the g_2 curves with large error bars. To better present the data, *pyXPCSviewer* provides three g_2 display modes: (i) multiple, (ii) single and (iii) combined. In multiple mode, each Q value has subplots and g_2 curves from different datasets are plotted in the same subplot. This mode is useful for comparing different datasets that are often the same material under different conditions. In single mode, the g_2 curves of one dataset are plotted together in one image that makes it possible to compare g_2 from different Q values. In combined mode, there is only one figure and it shows all g_2 curves from all datasets.

pyXPCSviewer provides two widely used functions for parameterizing the decay of correlations. The single exponential decay model captures the de-correlation of simple diffusion systems represented by

$$g_2 = a \exp\left[-2\left(\frac{x}{b}\right)^c\right] + d, \tag{1}$$

in which a, b, c and d correspond to the contrast, the characteristic decorrelation time, the stretch (compression) ratio and the baseline, respectively. For material systems that possess two-stage diffusion phenomena, the double exponential decay model is represented by

$$g_2 = a\left\{f \exp\left[-2\left(\frac{x}{b}\right)^c\right] + (1-f)\exp\left[-2\left(\frac{x}{b_2}\right)^{c_2}\right]\right\}^2 + d. \tag{2}$$

This model has a second exponential term with its decay time (b_2) and stretch (compression) ratio (c_2). Parameter f describes the fractional ratios of the two terms.

Users can define the bounds for each fitting variable. In addition, some variables can be fixed to values pre-determined at the beamline. For example, the contrast value can be calibrated with a standard static sample and thus be fixed. *pyXPCSviewer* will use the fixed values and compute the fitting uncertainty accordingly. This reduces the degrees of freedom and makes the fitting converge faster. This tab also supports the use of additional models from the user community. If the single exponential model is selected to fit the data, it is also possible to perform power-law fitting in the diffusion tab.

2.5. Two-time visualization

When a sample is not in a stationary state, the dynamics will not be time-invariant and so the decorrelation time-scale is a function of time. In this case, the multi-tau correlation, which assumes time-invariance, does not represent the dynamical behavior and a two-time correlation is required (Sutton *et al.*, 2003). As the name suggests, the two-time correlation $C2(t_1, t_2, Q)$ computes the correlation at time t_1 and t_2 for equivalent Q.

To visualize the result of a two-time correlation, *pyXPCSviewer* provides two panels that are shown in Fig. 5. The upper panel plots the scattering pattern and the Q-map applied when computing the two-time correlation. The two plots are linked so they can be zoomed in and out simultaneously. Users can interactively select regions on either the scattering or Q-map plot. The two-time correlation for the last two selected points is then plotted in the bottom panel. Users can compare two correlation figures side by side. From the two-time correlation map, the g_2 curves can be computed. Full g_2 is obtained by averaging the whole C_2 map while partial g_2 curves are computed from the sub-regions. This feature can help users determine whether the sample dynamics are stationary or not.

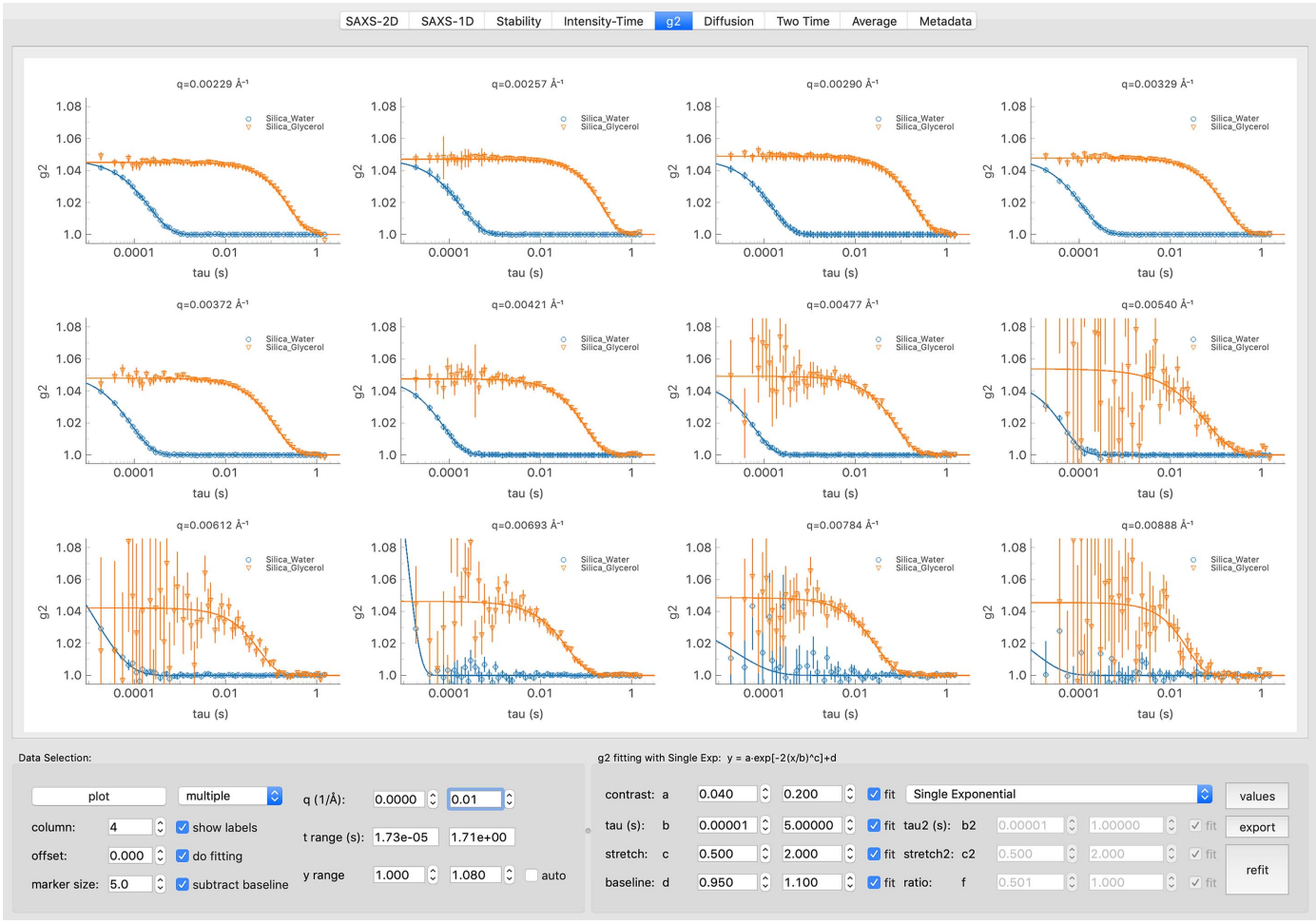


Figure 4
Multi-tau visualization and modeling. Users can customize the plot ranges, offset and the layout. Two widely used fitting models are included. Additional details about this module can be found in the text.

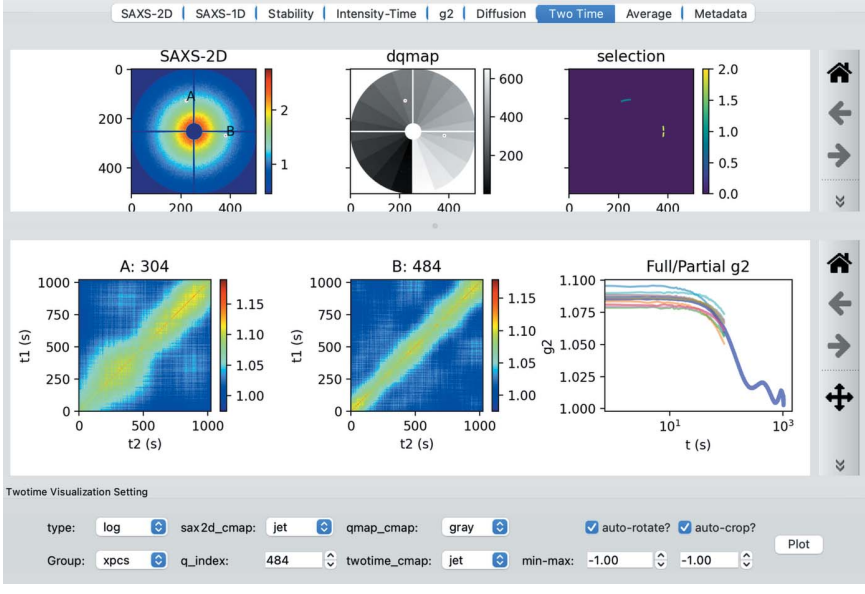


Figure 5
Two-time correlation visualization. The upper panels shows the scattering pattern as well as the dynamical Q-map that allows users to select regions (A and B) of interest to explore. The lower panels show the two-time maps for the selected points. The full (blue) and partial (red) g_2 curves for selection B are also displayed.

2.6. Averaging of XPCS results

XPCS can provide meaningful information from samples with extremely low scattering intensities (Sheyfer *et al.*, 2022; Lurio *et al.*, 2021; Partain *et al.*, 2021), but doing so requires averaging of XPCS results from hundreds or sometimes even thousands of repeating measurements to achieve the signal-to-noise ratio (SNR) of g_2 sufficient for quantitative analysis (Zhang *et al.*, 2021). To avoid prolonged X-ray exposure and mitigate the beam damage, such repeated measurements are usually performed on different locations of the sample, which makes the averaging of the results susceptible to spatial heterogeneity of the sample. These 'outlier' measurements can be difficult to identify manually especially when there are thousands of repeating

measurements from a single sample condition. The outliers usually have g_2 with a numerical value much higher than meaningful measurements due to near-zero or abnormal scattering intensity distributions and will significantly skew the statistics if not removed from the averaging.

pyXPCSviewer implements an averaging toolbox in the Average tab for these purposes as shown in Fig. 6. It identifies outliers based on the g_2 baseline, which is very sensitive to abnomalities in the intensity distribution within the region of interest where g_2 is calculated (Sheyfer *et al.*, 2020). As shown in Fig. 6, multiple outliers can be removed by adjusting the upper and lower thresholds (blue dots that fall beyond the double red lines), and the averaging yields g_2 with improved statistics as shown in the inset. Averaging also drastically reduces the data size that users need to interact with accelerating scientific interpretation of XPCS results both during and after the beam time. Future development will enhance the outlier identification with established machine learning algorithms (Pedregosa *et al.*, 2011) to pick up minute sample heterogeneity from 2-D SAXS patterns or slight radiation damage trends from stability plots that will not only lead to artifact-free XPCS results but that can also be combined with real-time XPCS analysis to enable autonomously guided XPCS measurements.

2.7. Metadata search

The Metadata tab gives users instant access to the metadata inside *pyXPCSviewer* without reading the file using separate software. The metadata for an XPCS dataset captures all the information about the measurement, such as the X-ray energy, frame rates and exposure time, temperature, and the synchrotron ring current. In addition, parameters passed to the correlation algorithms, such as the Q-map and the starting and ending frames, are also recorded in the metadata. Using the Data Exchange HDF format, the metadata is stored separately and independently from the XPCS analysis results. Using the metadata one can reproduce the conditions of the measurement and re-run the correlation algorithms. Due to the hierarchical structure of HDF files, the metadata is easily organized and stored with the data type, and units if applicable. *pyXPCSviewer* implements a simple metadata viewer function that performs a breadth-first search (BFS) to retrieve the metadata fields and shows the results in a tree structure. Users can use a string filter to search and locate any metadata fields easily.

2.8. Script mode

One of the essential features in *XPCSGUI* and *GIXSGUI* (Jiang, 2015) is script mode, which allows GUI functionalities to be called via a command line interface (CLI) in a programming environment (*e.g.* a Matlab terminal). This feature has been impactful for the user community over the past decade as it allows much of the XPCS data reduction process to be automated. *pyXPCSviewer* also includes a script mode, where scripts created by users can import *pyXPCSviewer*'s underlying modules and use the internal attributes and methods (*e.g.* loading HDF files, averaging of g_2, g_2 fitting) defined in *pyXPCSviewer*. The script mode is more efficient than the GUI when processing large batches of data. All script-mode functions, including the syntax and the mathematical formulas, are documented on the GitHub Wiki page of *pyXPCSviewer*.

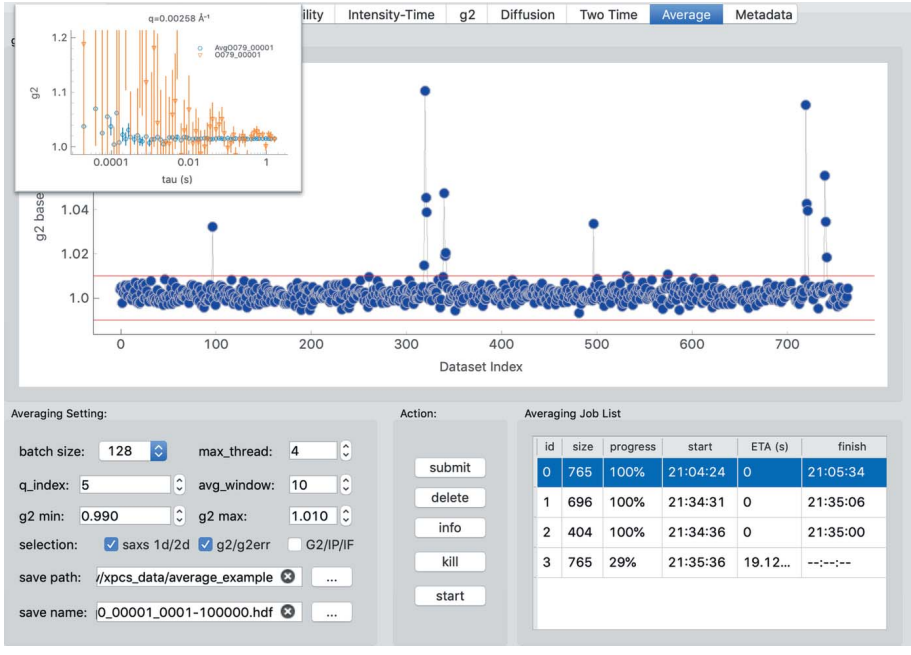


Figure 6
The averaging toolbox. Users set up a cutoff to remove outliers. The averaging jobs can be managed in the Action panel. The plot inset (top left) shows the g_2 curves for a single dataset (orange) and the averaged result of 765 datasets (blue), showing a significant improvement in signal-to-noise ratio obtained by suitable averaging.

3. Discussion and outlook

Built upon the design of *XPCSGUI*, which has been deployed and tested by users at 8-ID-I for more than a decade, *pyXPCSviewer* provides both the interactive features that allow for rapid result visualization and decision making during the beam time and script mode that streamlines data reading, fitting and plotting after beam time. Compared with the Matlab-based predecessor, *pyXPCSviewer* is open source and built with established packages and design patterns for easy maintenance and customization from the user community, a timely and essential feature given the emerging new XPCS techniques enabled by the commissioning of diffraction-limited storage rings around the world. With established Python platforms for beamline control

(Bluesky) and automated data transfer (DM Workflow), *pyXPCSviewer* fits naturally into the ongoing development of a beamline and analysis scientific Python ecosystem that is being leveraged by synchrotron facilities world-wide for enhanced information exchange and more efficient use of computing resources. In addition, by enabling feedback loops between XPCS result generation and beamline control using the Python ecosystem, AI-driven autonomous scientific studies are possible by using the abundant tools for AI/ML in Python. These developments will greatly accelerate scientific discoveries in XPCS that require large-scale combinatorial studies where the data processing bandwidth of humans is a key limiting factor.

Funding information

This research used resources of the Advanced Photon Source, a US Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory, and is based on research supported by the US DOE Office of Science-Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.

References

Arkilic, A., Allan, D. B., Caswell, T. A., Li, L., Lauer, K. & Abeykoon, S. (2017). *Synchrotron Radiat. News*, **30**(2), 44–45.

Begam, N., Ragulskaya, A., Girelli, A., Rahmann, H., Chandran, S., Westermeier, F., Reiser, M., Sprung, M., Zhang, F., Gutt, C. & Schreiber, F. (2021). *Phys. Rev. Lett.* **126**, 098001.

Campagnola, L. (2022). *PyQtGraph – Scientific Graphics and GUI Library for Python*, https://www.pyqtgraph.org.

Chenevier, D. & Joly, A. (2018). *Synchrotron Radiat. News*, **31**(1), 32–35.

Chu, M. (2021). *pyXpcsViewer*, https://github.com/AdvancedPhotonSource/pyXpcsViewer.

Dallari, F., Martinelli, A., Caporaletti, F., Sprung, M., Grübel, G. & Monaco, G. (2020). *Sci. Adv.* **6**, eaaz2982.

De Carlo, F., Gürsoy, D., Marone, F., Rivers, M., Parkinson, D. Y., Khan, F., Schwarz, N., Vine, D. J., Vogt, S., Gleber, S.-C., Narayanan, S., Newville, M., Lanzirotti, T., Sun, Y., Hong, Y. P. & Jacobsen, C. (2014). *J. Synchrotron Rad.* **21**, 1224–1230.

Dierker, S. B., Pindak, R., Fleming, R. M., Robinson, I. K. & Berman, L. (1995). *Phys. Rev. Lett.* **75**, 449–452.

Evenson, Z., Ruta, B., Hechler, S., Stolpe, M., Pineda, E., Gallino, I. & Busch, R. (2015). *Phys. Rev. Lett.* **115**, 175701.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. & Patterns, D. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Vol. 99 of *Addison-Wesley Professional Computing Series*. Reading, MA: Addison-Wesley.

Giordano, V. M. & Ruta, B. (2016). *Nat. Commun.* **7**, 10344.

Grübel, G., Madsen, A. & Robert, A. (2008). *Soft Matter Characterization*, pp. 953–995. Dordrecht: Springer.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020). *Nature*, **585**, 357–362.

Hunter, J. D. (2007). *Comput. Sci. Eng.* **9**, 90–95.

Ilavsky, J. & Jemian, P. R. (2009). *J. Appl. Cryst.* **42**, 347–353.

Jiang, Z. (2015). *J. Appl. Cryst.* **48**, 917–926.

Khan, F., Narayanan, S., Sersted, R., Schwarz, N. & Sandy, A. (2018). *J. Synchrotron Rad.* **25**, 1135–1143.

Lin, C., Dyro, K., Chen, O., Yen, D., Zheng, B., Arango, M. T., Bhatia, S., Sun, K., Meng, Q., Wiegart, L. & Chen-Wiegart, Y. K. (2021). *Appl. Mater. Today*, **24**, 101075.

Lou, W., Mayes, C., Cai, Y. & White, G. (2021). *Proceedings of the 12th International Particle Accelerator Conference (IPAC2021)*, 24–28 May 2021, Campinas, Brazil, pp. 3177–3180. WEPAB234.

Lurio, L. B., Thurston, G. M., Zhang, Q., Narayanan, S. & Dufresne, E. M. (2021). *J. Synchrotron Rad.* **28**, 490–498.

Martensson, N. & Eriksson, M. (2018). *Nucl. Instrum. Methods Phys. Res. A*, **907**, 97–104.

Myint, P., Ludwig, K. F., Wiegart, L., Zhang, Y., Fluerasu, A., Zhang, X. & Headrick, R. L. (2021). *Phys. Rev. Lett.* **126**, 016101.

Nakaye, Y., Sakumura, T., Sakuma, Y., Mikusu, S., Dawiec, A., Orsini, F., Grybos, P., Szczygiel, R., Maj, P., Ferrara, J. D. & Taguchi, T. (2021). *J. Synchrotron Rad.* **28**, 439–447.

Partain, B. D., Zhang, Q., Unni, M., Aldrich, J., Rinaldi-Ramos, C. M., Narayanan, S. & Allen, K. D. (2021). *Osteoarthritis Cartilage*, **29**, 1351–1361.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). *J. Mach. Learn. Res.* **12**, 2825–2830.

Perakis, F., Amann-Winkel, K., Lehmkühler, F., Sprung, M., Mariedahl, D., Sellberg, J. A., Pathak, H., *sp*äh, A., Cavalca, F., Schlesinger, D., Ricci, A., Jain, A., Massani, B., Aubree, F., Benmore, C. J., Loerting, T., Grübel, G., Pettersson, L. G. M. & Nilsson, A. (2017). *Proc. Natl Acad. Sci. USA*, **114**, 8193–8198.

Petoukhov, M. V., Franke, D., Shkumatov, A. V., Tria, G., Kikhney, A. G., Gajda, M., Gorba, C., Mertens, H. D. T., Konarev, P. V. & Svergun, D. I. (2012). *J. Appl. Cryst.* **45**, 342–350.

Rio, M. S. & Rebuffi, L. (2019). *AIP Conf. Proc.* **2054**, 060081.

Riverbank Computing (2022). *PyQt*, https://riverbankcomputing.com/software/pyqt/.

Rogers, W., Nicholls, T. J. R. & Wilson, A. A. (2019). *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, 5–11 October 2019, New York, NY, USA, pp. 232–235. MOPHA017.

Ruta, B., Hechler, S., Neuber, N., Orsi, D., Cristofolini, L., Gross, O., Bochtler, B., Frey, M., Kuball, A., Riegler, S. S., Stolpe, M., Evenson, Z., Gutt, C., Westermeier, F., Busch, R. & Gallino, I. (2020). *Phys. Rev. Lett.* **125**, 055701.

Schirmer, D. & Althaus, A. (2020). *Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS2019)*, 5–11 October 2019, New York, NY, USA, pp. 1421–1425. WEPHA137

Schroer, C. G., Agapov, I., Brefeld, W., Brinkmann, R., Chae, Y.-C., Chao, H.-C., Eriksson, M., Keil, J., Nuel Gavaldà, X., Röhlsberger, R., Seeck, O. H., Sprung, M., Tischer, M., Wanzenberg, R. & Weckert, E. (2018). *J. Synchrotron Rad.* **25**, 1277–1290.

Senses, E., Ansar, S. M., Kitchens, C. L., Mao, Y., Narayanan, S., Natarajan, B. & Faraone, A. (2017). *Phys. Rev. Lett.* **118**, 147801.

Sheyfer, D., Servis, M. J., Zhang, Q., Lal, J., Loeffler, T., Dufresne, E. M., Sandy, A. R., Narayanan, S., Sankaranarayanan, S. K. R. S., Szczygiel, R., Maj, P., Soderholm, L., Antonio, M. R. & Stephenson, G. B. (2022). *J. Phys. Chem. B*, **126**, 2420–2429.

Sheyfer, D., Zhang, Q., Lal, J., Loeffler, T., Dufresne, E. M., Sandy, A. R., Narayanan, S., Sankaranarayanan, S. K. R. S., Szczygiel, R., Maj, P., Soderholm, L., Antonio, M. R. & Stephenson, G. B. (2020). *Phys. Rev. Lett.* **125**, 125504.

Shi, X. B., Reininger, R., Harder, R. & Haeffner, D. (2017). *Proc. SPIE*, **10388**, 103880C.

Shpyrko, O. G. (2014). *J. Synchrotron Rad.* **21**, 1057–1064.

Sikorski, M., Jiang, Z., Sprung, M., Narayanan, S., Sandy, A. R. & Tieman, B. (2011). *Nucl. Instrum. Methods Phys. Res. A*, **649**, 234–236.

Sutton, M., Laaziri, K., Livet, F. & Bley, F. (2003). *Opt. Express*, **11**, 2268.

Sutton, M., Lhermitte, J. R. M., Ehrburger-Dolle, F. & Livet, F. (2021). *Phys. Rev. Res.* **3**, 013119.

Veseli, S., Schwarz, N. & Schmitz, C. (2018). *J. Synchrotron Rad.* **25**, 1574–1580.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Bardelli, A. P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C. N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D. A., Hagen, D. R., Pasechnik, D. V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G. A., Ingold, G., Allen, G. E., Lee, G. R., Audren, H., Probst, I., Dietrich, J. P., Silterra, J., Webber, J. T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J. L., de Miranda Cardoso, J. V., Reimer, J., Harrington, J., Rodríguez, J. L. C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N. J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P. A., Lee, P., McGibbon, R. T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T. J., Robitaille, T. P., Spura, T., Jones, T. R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y. O. & Vázquez-Baeza, Y. (2020). *Nat. Methods*, **17**, 261–272.

Yavitt, B. M., Salatto, D., Zhou, Y., Huang, Z., Endoh, M., Wiegart, L., Bocharova, V., Ribbe, A. E., Sokolov, A. P., Schweizer, K. S. & Koga, T. (2021). *ACS Nano*, **15**, 11501–11513.

Zhang, L., Barrett, R., Cloetens, P., Detlefs, C. & Sanchez del Rio, M. (2014). *J. Synchrotron Rad.* **21**, 507–517.

Zhang, Q., Dufresne, E. M., Nakaye, Y., Jemian, P. R., Sakumura, T., Sakuma, Y., Ferrara, J. D., Maj, P., Hassan, A., Bahadur, D., Ramakrishnan, S., Khan, F., Veseli, S., Sandy, A. R., Schwarz, N. & Narayanan, S. (2021). *J. Synchrotron Rad.* **28**, 259–265.