JOURNAL OF APPLIED CRYSTALLOGRAPHY

ISSN 1600-5767

Received 7 July 2025
Accepted 26 November 2025

Edited by A. Barty, DESY, Hamburg, Germany

Keywords: Python packages; data reduction; data analysis; grazing-incidence X-ray diffraction GIXD; grazing-incidence wide-angle X-ray scattering GIWAXS; grazing-incidence small-angle X-ray scattering GISAXS; wide-angle X-ray scattering WAXS; small-angle X-ray scattering SAXS.

Supporting information: this article has supporting information at journals.iucr.org/j

pygid: a Python package for fast data reduction in grazing-incidence diffraction

Ainur Abukaev,[a] Constantin Völter,[a] Mikhail Romodin,[a] Sebastian Schwartzkopff,[a] Florian Bertram,[b] Oleg Konovalov,[c] Alexander Hinderhofer,[a] Dmitry Lapkin[a]* and Frank Schreiber[a]*

[a]Institut für Angewandte Physik, Universität Tübingen, Auf der Morgenstelle 10, 72076 Tübingen, Germany, [b]Deutsches Elektronen-Synchrotron DESY, Notkestraße 85, 22607 Hamburg, Germany, and [c]European Synchrotron Radiation Facility (ESRF), 71 avenue des Martyrs, 38000 Grenoble, France. *Correspondence e-mail: dmitry.lapkin@uni-tuebingen.de, frank.schreiber@uni-tuebingen.de

Advances in X-ray and neutron sources, as well as in area-detector technologies, enable the recording of several terabytes of raw two-dimensional detector data in a single experiment. While several efficient integration and conversion tools are available for data collected in transmission geometry, analogous solutions for grazing-incidence diffraction (including grazing-incidence X-ray diffraction and grazing-incidence wide-angle X-ray scattering) experiments have not yet achieved the same level of efficiency. The development of new data analysis tools, including machine-learning-based software for X-ray data, necessitates the establishment of a standardized format for the converted data. To address these challenges, we have developed a new Python library, *pygid*, which is designed to facilitate fast data processing while providing compatibility with various raw data formats, a standardized data storage format and an intuitive interface for straightforward use. *pygid* supports three types of coordinate systems and both transmission and grazing-incidence geometries. It is capable of handling large datasets, performing one-dimensional line cuts and simulating expected Bragg peak positions for given structures. The package facilitates sample and experimental metadata curation in accordance with the FAIR principles. As an integral part of the broader *mlgid* pipeline, *pygid* serves as the initial step linking raw scattering patterns with machine learning tools for data analysis. The *pygid* package is accessible at https://github.com/mlgid-project.

1. Introduction

X-ray and neutron scattering techniques are essential tools in materials science, chemistry, biophysics and condensed matter physics. Their widespread use is supported by the continuous progress in large-scale X-ray and neutron source infrastructures, which provide high-brilliance and tunable radiation for advanced structural investigations (Willmott, 2019). At the same time, the development of modern 2D detectors with continuous readout and minimal dead times down to 100 ns has significantly improved spatial and temporal resolution in scattering measurements, enabling fast data acquisition in *e.g.* time-resolved and *in situ* experiments (Bein *et al.*, 2015; Bommel *et al.*, 2014; Eres *et al.*, 2019; Ferrer *et al.*, 2013; Ju *et al.*, 2021; Kowarik *et al.*, 2006; Magnussen *et al.*, 2024; Nicklin *et al.*, 2017; Richard *et al.*, 2010; Ulbrandt *et al.*, 2020; Zhang *et al.*, 2024). As a result, the growing volume and complexity of collected data have created a need for efficient and scalable software tools capable of reliable data reduction and analysis.

Among the various X-ray scattering techniques, grazing-incidence wide- and small-angle X-ray scattering (GIWAXS/GISAXS) methods have become indispensable tools for

pygid
Preprocessing of Grazing Incidence Diffraction

studying thin films, nanostructured materials and surfaces (Banerjee *et al.*, 2021; Eisenberger & Marra, 1981; Feidenhans'l, 1989; Smilgies, 2025; Werzer *et al.*, 2024). These methods utilize an incident X-ray beam at a shallow angle, around the critical angle of total reflection, maximizing surface sensitivity (Robinson & Tweet, 1992). GIWAXS provides detailed information on crystal unit cells, atomic/molecular arrangement, degree of crystallinity and orientation of the crystallites on the substrate surface. These techniques have become widely applied in a broad range of research areas, such as organic and hybrid electronics and photovoltaics, where the precise structural characterization of thin films is crucial. Examples include metal halide perovskites (Barrit *et al.*, 2022; Mundt & Schelhas, 2020; Schlipf & Müller-Buschbaum, 2017; Steele *et al.*, 2023), organic small molecules (Diao *et al.*, 2014; Gu *et al.*, 2018; Hodas *et al.*, 2018; Lapkin *et al.*, 2025; Richter *et al.*, 2017; Arias *et al.*, 2021) and polymers (Manley *et al.*, 2017; Müller-Buschbaum, 2014; Posselt *et al.*, 2017; Yang *et al.*, 2020). Historically, the term GIXD (grazing-incidence X-ray diffraction) was more commonly used, but it is considered equivalent in this context. GISAXS is particularly suited for characterizing nano- and microscale morphology, including particle distribution, shape and surface roughness, by analyzing intensity scattered at small angles (Kaune *et al.*, 2009; Smilgies *et al.*, 2002; Smilgies, 2022; Smilgies, 2025). Meanwhile, GISANS – the neutron analog of GISAXS – offers complementary advantages for soft and organic materials due to its sensitivity to light elements like hydrogen, while isotopic substitution allows for tunable contrast variation (Dosch, 1992; Hamilton *et al.*, 1994; Jones *et al.*, 1999; Müller-Buschbaum *et al.*, 2003; Müller-Buschbaum, 2013; Steitz *et al.*, 2004). However, the complexity of the grazing-incidence geometry, including symmetry breaking and distortion of peaks, poses challenges for data reduction, intensity correction and analysis compared with transmission scattering experiments.

A fundamental step in grazing-incidence diffraction (GID) data analysis is the conversion of raw 2D detector scattering patterns into physically meaningful cylindrical coordinates (q_{xy}, q_z) – the in-plane and out-of-plane components of the scattering vector – or/and into polar coordinates ($q_{\rm abs}$, χ) – the absolute value of the scattering vector and azimuthal angle (Section 3.2 and Appendix *A*). This process should also be accompanied by various intensity corrections and masking of dead and hot pixels (Section 4). Several software tools have been developed to facilitate GID data processing, mostly written in Python and MATLAB programming languages. *GIXSGUI* is a MATLAB-based tool with script-based access and a graphical user interface (GUI) (Jiang, 2015). It provides software for 2D data visualization, reduction, line cutting and indexing of grazing-incidence X-ray scattering data, and for handling large datasets, such as those generated in *in situ* and *in operando* studies at synchrotron facilities. *INSIGHT* (*in situ* GIXS heuristic tool) is an object-oriented Python package that can work with data batches (Reus *et al.*, 2024). The main feature of *INSIGHT* is the usage of frame-to-frame corrections of experimental parameters, such as sample-to-detector distance, that can be changed due to thermal expansions during *in situ* experiments. *GIWAXS-SIIRkit* is a MATLAB-based package designed for quantitative structural characterization of thin films using GIWAXS scattering patterns (Savikhin *et al.*, 2020). One of its key features is the ability to assess scattering intensity variations by considering factors such as refractive index shift and incident beam footprint. *indexGIXS* provides a GUI for experimental data visualization, scattering pattern simulation and peak indexing (Smilgies & Li, 2021). *pyFAI* (Python fast azimuthal integration) is optimized for fast data reduction, supporting azimuthal integration and detector calibration. The package provides a pixel-splitting method for conversion and offers the fastest 1D integration time (down to 48 ms for a 4 megapixel pattern on a four-core office computer) (Ashiotis *et al.*, 2015). The present work introduces the new Python-based package *pygid*. Our approach considers both the functionality and practical experience gained from existing packages. *pygid* features increased efficiency and an extended range of intensity corrections.

As experimental techniques advance and high-throughput measurements become more common, traditional analysis approaches struggle to keep pace with the sheer amount of information collected. In this context, machine learning (ML) has received growing attention and development in the past few years. It has emerged as a powerful tool for large and complex datasets (Starostin *et al.*, 2022*a*; Starostin *et al.*, 2022*b*; Pithan *et al.*, 2023; Völter *et al.*, 2025; Ziletti *et al.*, 2018). However, to exploit the potential of this approach fully, a standardized data format, which *pygid* provides, is essential to facilitate seamless integration with analysis software.

This article details the architecture, geometry conventions and data processing workflow of *pygid* and provides usage examples for different experimental setups. The article is part of a series of papers (Starostin *et al.*, 2022*a*; Starostin *et al.*, 2022*b*; Völter *et al.*, 2025) focused on GID data acquisition and analysis, with *pygid* serving as the first component of our data processing pipeline *mlgid* that bridges raw detector output and structural characterization.

2. The *pygid* package

The package was developed for fast GID data reduction, including grazing-incidence small- and wide-angle scattering experiments using both X-rays and neutrons (GISAXS, GIWAXS, GISANS). It supports a wide range of raw scattering pattern formats, performs 1D and 2D data conversions, and saves data in a standardized format. The ability to process batches of raw data makes it suitable for integration into synchrotron and neutron beamlines for online data reduction during measurements. The simple and intuitive design, along with the examples and documentation provided, makes it user friendly.

Python was chosen as the programming language for the development of the *pygid* package due to its readability, flexibility and extensive ecosystem of libraries, which make it particularly well suited for data analysis and rapid prototyping

(Nagpal & Gabrani, 2019; Saabith *et al.*, 2019). Additionally, Python's versatility and the modular nature of the *pygid* package allow for easy integration with other libraries and software and with scientific workflows, enabling seamless interaction with existing tools and systems, including synchrotron beamlines. However, since Python is an interpreted language it lacks computational performance when processing large datasets. To compensate for that, we used the *numexpr* library, which minimizes memory access and significantly accelerates mathematical operations by utilizing optimized multi-threading (McLeod *et al.*, 2018). For data conversion, we implemented the *OpenCV-Python* package (Bradski, 2000), which efficiently handles image processing tasks, enabling fast conversion and manipulation of 2D detector data. This combination of the Python environment for flexibility and the C-based libraries *numexpr* and *OpenCV* for computational speed allowed us to create an effective and high-performance tool for working with X-ray diffraction data, providing a balance between usability and performance.

3. Data processing flow

In this section, we describe the structure of the *pygid* package and the processing pipeline for raw data within the script, including geometry representation, experimental data handling and metadata curation. The first step of the conceptual design of *pygid* involves calculating coordinate maps and intensity correction matrices based on the experimental parameters (Fig. 1). Raw data loaded from the specified path are intensity-corrected, masked and then transformed using these maps. To facilitate the handling of large datasets, batch processing can be enabled. Finally, the processed data in 32-bit floating-point format, along with sample metadata and experimental parameters, are stored for further analysis.

3.1. Experimental parameters

To store and operate the experimental parameters, *pygid* uses a class named `ExpParams`. It defines six parameters related to the detector orientation: the sample position projection onto the detector plane (`poni1`, `poni2` in metres) or the direct beam position (`centerX`, `centerY` in pixels), the sample-to-detector distance (SDD) along the normal to the detector plane before applying any rotations, and three detector rotation angles around the laboratory coordinate axes (`rot1`, `rot2`, `rot3`) with the origin at the sample positions. Additionally, it stores experimental details such as the X-ray wavelength, detector pixel size, and image transformation flags (`fliplr`, `flipud`, `transp`) for horizontal flipping, vertical flipping and transpose, respectively. All these parameters, except for the last set of keys, can be imported from a PONI file created using the *pyFAI* package or its GUI (Ashiotis *et al.*, 2015). However, manual input of these values is also supported. The `ExpParams` class can additionally handle both static and dynamic masks. Users can provide either a 2D array for the static mask (`mask`) or a file path (`mask_path`) pointing to a mask file in *NumPy* (https://numpy.org/), EDF or TIFF format. The static mask is applied uniformly to all images to exclude detector gaps, the direct beam region or the beam-stop shadow. Dynamic masks, in contrast, are generated from each raw scattering frame and are based on user-defined minimum and maximum intensity thresholds (`count_range`), effectively excluding hot and dead pixels.

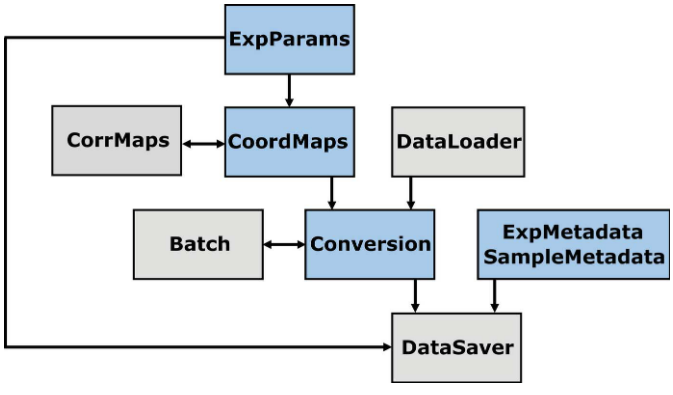

Figure 1
Conceptual architecture of *pygid*. Blue boxes correspond to the classes that the user interacts with, while gray boxes represent internal classes.

3.2. Coordinate map calculation

The functionality of the `CoordMaps` class can be described in three steps:

(i) computation of the detector pixel coordinates in Cartesian (q_1, q_2), cylindrical (q_{xy}, q_z), polar (q_{abs}, χ) or pseudo-polar (q_{abs}, $q_{abs}\chi$) systems in reciprocal space for both transmission and GID geometries;

(ii) estimation of maximum measured q values based on the detector position and size (optional);

(iii) calculation of intensity correction matrices (optional).

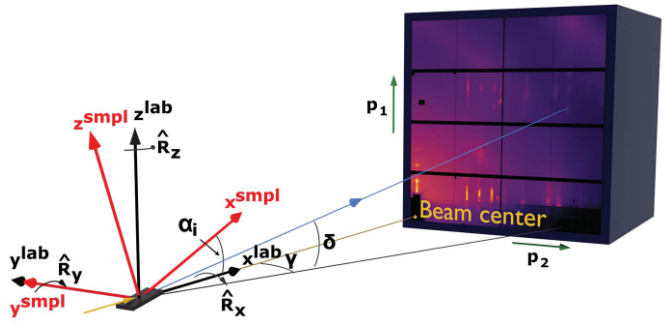

Figure 2
Scattering geometry and the three coordinate systems used in *pygid*: (p_1, p_2) – pixel positions in the detector real-space coordinate system (green); (x^{lab}, y^{lab}, z^{lab}) – laboratory reciprocal-space coordinate system used in transmission geometry (black); (x^{smpl}, y^{smpl}, z^{smpl}) – sample coordinate system in reciprocal space, rotated by the angle of incidence α_i counter-clockwise around the y_{lab} axis, for grazing-incidence geometry (red), direct beam (yellow) and scattered beam (blue) with horizontal (γ) and vertical (δ) scattering angles. The angle of incidence α_i is exaggerated for clarity in the visualization.

All calculations rely on three mutually connected orthonormal right-handed coordinate systems (Fig. 2), similar to those described by Breiby *et al.* (2008) and Smilgies & Blasini (2007). The first is the detector coordinate system (DCS) in real space (d_1, d_2, d_3), which is defined by pixel positions (p_1, p_2) in the raw pattern and the SDD. The second is the laboratory coordinate system (LCS) in reciprocal space (indicated by the superscript lab, *e.g.* q^{lab}), centered at the point where the X-ray beam intersects the sample. In this system, the direct beam propagates along the x^{lab} axis. These two frames are related through detector rotations (`rot1`, `rot2`, `rot3`) around the laboratory coordinate frame. While this description is sufficient for transmission geometry, grazing-incidence experiments require an additional sample coordinate system (SCS, denoted using the superscript smpl). This system is linked to the laboratory frame via a rotation matrix around the y^{lab} axis to the angle of incidence.

The primary function of the `CoordMaps` class is to compute pixel coordinates of the transformed image in detector space, given predefined coordinate ranges (Fig. 3). First, the given ranges in polar cylindrical (q_{xy}, q_z), (q_{abs}, χ), pseudo-polar (q_{abs}, $q_{\mathrm{abs}}\chi$) or 2D Cartesian (q_1, q_2) coordinates are transformed into Cartesian q-space coordinates, $\mathbf{q} = (q_x, q_y, q_z)$. In the case of GID geometry, the calculated $\mathbf{q}$ vector is initially defined in the sample coordinate system and is then rotated by the incidence angle to be represented in the laboratory coordinate system. The corresponding final wavevector $\mathbf{k}_{\mathrm{f}}$ is then calculated in the LCS as $\mathbf{k}_{\mathrm{f}} = \mathbf{k}_{\mathrm{i}} + \mathbf{q}$, where $\mathbf{k}_{\mathrm{i}} = (2\pi/\lambda, 0, 0)$. To transform $\mathbf{k}_{\mathrm{f}}$ into detector space, three rotation matrices, defined by the detector rotation angles (`rot1`, `rot2`, `rot3`), are applied. The resulting vector is proportional to the pixel position $\mathbf{d}^* = (d_1^*, d_2^*, d_3^*)$ in real space. In the final step, the pixel coordinates (p_1, p_2) are computed from $\mathbf{d}^*$ using experimental parameters, including the direct beam position and pixel size.

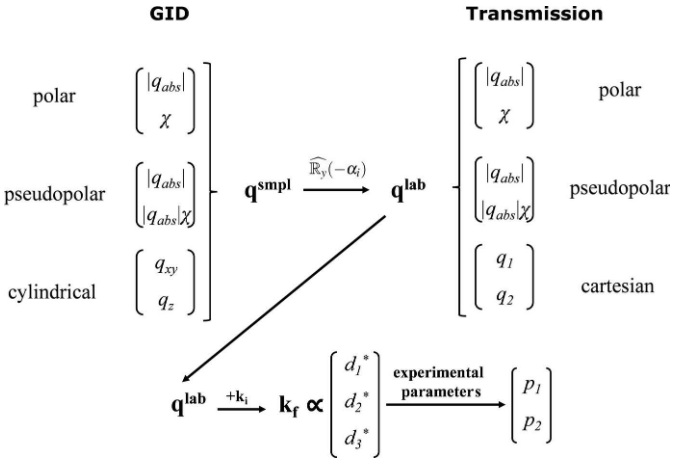

Figure 3
Schematic representation of coordinate map calculation from the given coordinate types and ranges. $\mathbf{q}^{\mathrm{smpl}}$ and $\mathbf{q}^{\mathrm{lab}}$ are scattering vectors in the sample and laboratory coordinate systems, respectively, related to each other via a rotation matrix around the y^{lab} axis. $\mathbf{k}_{\mathrm{i}}$ is the incident wavevector in the LCS, $\mathbf{k}_{\mathrm{f}}$ the scattered wavevector in the LCS, (d_1^*, d_2^*, d_3^*) the real-space pixel positions in the LCS and (p_1, p_2) the pixel positions in the converted images.

A key feature of the *pygid* package is that it reuses the calculated coordinate map (p_1, p_2), representing the converted image pixel positions, multiple times for different scattering patterns recorded under the same experimental conditions (*e.g.* fixed angle of incidence for time scans). This reusability significantly reduces the conversion time, as the coordinate map does not need to be recalculated for each individual pattern, thereby improving the overall efficiency of the data processing pipeline.

The estimation of the q range is based on the opposite conversion process from the pixel coordinates of the raw image in detector space to q values in laboratory and sample spaces for transmission and GID, respectively. Only corner pixels and edge pixels on the same horizontal and vertical lines as the direct beam pixel are processed for maximum scattering vector and q values in cylindrical (q_{xy}, q_z) and Cartesian (q_1, q_2) calculations. However, for angular range evaluation all border pixels are processed. Finally, intensity correction matrices require pixel positions in reciprocal space for each pixel of the raw scattering pattern. The implemented intensity corrections will be described further in Section 4.

3.3. Data loading

The `DataLoading` class is designed to handle raw detector image files in a variety of formats. It supports file types that can be opened by the *FabIO* library (EDF, TIFF, CBF) (Knudsen *et al.*, 2013) and the *H5py* library (Collette *et al.*, 2023) for files with more complex structure, including HDF5 and NeXus formats. The *FabIO* library provides efficient access to a wide range of 2D detector images. The *H5py* library has demonstrated superior performance in terms of data loading speed compared with other packages such as *PyTables* (Alted *et al.*, 2002), *netCDF4* (Pierce, 2025) and *h5netcdf* for HDF5 files (Table S1 in the supporting information). However, the actual loading time is highly dependent on the data storage infrastructure, the internal file structure of the files, and any external references to other libraries or resources.

Users also have the option to load the scattering patterns externally and transfer the raw data as 2D or 3D arrays into *pygid*. The `DataLoading` class operates internally and is not intended for direct user interaction. Instead, users interact with the `Conversion` class, where they specify the data file path and the location of raw data arrays (for HDF5 and Nexus files). These arrays are then processed and prepared for subsequent analysis.

3.4. Conversion

The preliminary calculated coordinate maps and loaded data are passed to the `Conversion` class, which first applies the correction matrices calculated in the `CoordMaps` class. According to the calculated coordinate map, the image can be represented in polar, pseudo-polar, cylindrical or 2D Cartesian coordinates (Table 1).

Table 1
2D conversion types and corresponding axes.

Experiment geometry	Converted image type	Function name	Resulting image name	Corresponding axis names
GID	Cylindrical	`det2q_gid()`	`img_gid_q`	`q_xy`, `q_z`
GID	Polar	`det2pol_gid()`	`img_gid_pol`	`q_gid_pol`, `ang_gid_pol`
GID	Pseudo-polar	`det2pseudopol_gid()`	`img_gid_pseudopol`	`q_gid_rad`, `q_gid_azimuth`
Transmission	Cartesian	`det2q()`	`img_q`	`q_1`, `q_2`
Transmission	Polar	`det2pol()`	`img_pol`	`q_pol`, `ang_pol`
Transmission	Pseudo-polar	`det2pseudopol()`	`img_pseudopol`	`q_rad`, `q_azimuth`

The primary remapping function utilizes geometric image transformations from the *OpenCV-Python* library [`cv2.remap()`] (Gonzalez & Woods, 2018). Since new pixel positions may be non-integer and pixel intensities need to be accurately estimated, the package employs several interpolation techniques. Five interpolation methods are implemented in the script: nearest-neighbor, bilinear, bicubic and Lanczos (Cullum & Willoughby, 2002) interpolation, and resampling based on the pixel area relation. This allows users to balance speed and quality depending on the task, whether it involves image downscaling or upscaling.

3.5. Data saving

To store individual converted images, complete datasets and even multiple datasets within a single file, we employ the widely adopted NeXus format (Klosowski *et al.*, 1997; Könnecke *et al.*, 2015), which provides a standardized framework for data exchange and archiving in neutron and X-ray experiments. File writing is implemented using the *H5py* library (Collette *et al.*, 2023). The format allows for the storage not only of converted patterns but also of experimental parameters and sample descriptions.

The data type closely related to GIWAXS/GISAXS data is the NXsas application definition, which was designed for storing small-angle scattering (SAS) data in the NeXus format. However, we have slightly modified the `data` group to store arrays of scattering data for motor or time scans, as is implemented in the NXscan definition (Fig. 4). An additional `analysis` group is used to store the results of peak detection and fitting at the next analysis step. Converted images can be stacked to the previously calculated data arrays if they have the same shape, or can be saved in a separate `NXentry` group. The naming of datasets for different types of coordinates is shown in Tables 1 and 2. Additionally, a single converted image can be visualized and saved using the *matplotlib* library, which supports both vector (PDF, SVG, EPS, PGF) and raster (PNG, JPG/JPEG, TIFF, BMP) formats (Hunter, 2007).

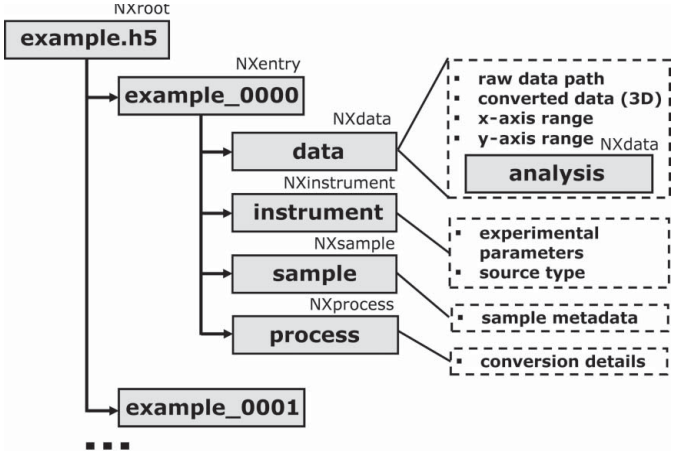

Figure 4
Overview of the structure of a saved NeXus file as a modified NXsas application definition.

The `instrument` group contains data from the `ExpParams` class, following the standard naming defined in the NXsas format. Additional information about the experiment and source can be added using the `ExpMetadata` class (Table S2). Details of the transformation, such as the date and the applied intensity corrections, are stored in the `process` group.

Finally, the `sample` group stores the sample-related metadata. We strongly recommend that the metadata include the sample name, structure, preparation description and experimental conditions via the `SampleMetadata` class (Tables S2 and S3). Sample metadata can also be imported directly from a YAML file similar to the ORSO (Open Reflectometry Standards Organization, https://www.reflectometry.org/) specification. Users may further extend the sample group with custom fields, for example chemical formula, temperature, pressure, mass *etc.*, as proposed by the DAPHNE4NFDI initiative (Barty *et al.*, 2023; Lohstroh *et al.*, 2024; Amelung *et al.*, 2025) in accordance with the FAIR Guiding Principles (Wilkinson *et al.*, 2016). Both metadata classes support formats such as strings, lists, integer or float values, and *NumPy* arrays.

3.6. Other features

3.6.1. Batch analysis

The package automatically uses the batch mode when the number of loaded patterns is larger than `batch_size` (32 is the default value). Here, scattering patterns are loaded in batches when the remapping function is called. After remapping, the loaded scattering patterns are deleted from the memory to allow the next batch to be processed and to avoid extensive memory usage. The result will not be plotted and returned; only saving in an HDF5 file is allowed.

3.6.2. Line profiles

In addition to 2D conversion functions, we have added 1D radial and azimuthal integration functions (Table 2). The corresponding functions call remapping into polar coordinates

Table 2
1D profiling and corresponding axes.

Function name	Resulting data name	Corresponding axis name	Description
`radial_profile_gid()`	`rad_cut_gid`	`q_gid_pol`	Makes polar remapping and averages in the given angular range for the GID geometry
`radial_profile()`	`rad_cut`	`q_pol`	Makes polar remapping and averages in the given angular range for the transmission geometry
`azim_profile_gid()`	`azim_cut_gid`	`ang_gid_pol`	Makes polar remapping and averages in the given radial range for the GID geometry
`azim_profile()`	`azim_cut`	`ang_pol`	Makes polar remapping and averages in the given radial range for the transmission geometry
`horiz_profile_gid()`	`horiz_cut_gid`	`q_xy`	Makes cylindrical remapping and averages in the given q_z range for the GID geometry

Table 3
Intensity corrections implemented in *pygid*.

Correction type	Correction usage key (Boolean)	Variables	Data type
Flat field	–	`flat_field`	ndarray (2D)
Dark current	–	`dark_current`	ndarray (2D)
Solid angle	`make_solid_angle_corr`	–	–
Polarization	`make_pol_corr`	`pol_type`	Float $\in (0, 1)$
Air attenuation	`make_air_attenuation_corr`	`air_attenuation_coeff`	Float > 0 (m^{-1})
Sensor attenuation	`make_sensor_attenuation_corr`	`sensor_attenuation_coeff`	Float > 0 (m^{-1})
		`sensor_thickness`	Float > 0 (m)
Sample absorption	`make_absorption_corr`	`sample_attenuation_coeff`	Float > 0 (m^{-1})
		`sample_thickness`	Float > 0 (m)
Lorentz	`make_lorentz_corr`	`powder_dim`	Integer, 2 or 3

in sample or laboratory spaces and perform averaging along the angular and radial axis. Users can adjust the resolution in both angular and radial directions, depending on their preference for accuracy or speed. For GID data, it is also possible to obtain horizontal profiles that are calculated from patterns in cylindrical coordinates by averaging in a small q_z range close to 0. For vertical profiles, which are not directly accessible due to the missing edge, we recommend calculating radial profiles with a small angular range close to the vertical axis according to the size of the missing wedge. The profiles can be plotted with adjustable color map, limits and distance between curves for multiple datasets, saved as a figure or in an HDF5 file, and returned as a *NumPy* array.

3.6.3. GIWAXS pattern simulation

We integrated the *pygidSIM* package (Romodin, 2025), which simulates expected Bragg peak positions in GIWAXS patterns based on crystallographic information (unit-cell parameters and atomic positions) provided in CIFs (Hall *et al.*, 1991). The package outputs the positions as cylindrical coordinates and intensities of Bragg reflections for various Miller indices. *pygid* enables users to overlay experimental patterns with simulated data for a set of CIFs and crystal orientations.

4. Intensity corrections

The importance and details of intensity corrections for 2D detector data in X-ray scattering experiments have been discussed in several articles (Gasser *et al.*, 2025; Jiang, 2015; Pauw *et al.*, 2017). While the positions of Bragg peaks in WAXS/GIWAXS experiments provide information about the crystal type and unit-cell parameters, their intensities make it possible to estimate the amplitude of structure factors corresponding to the arrangement of individual atoms within the unit cell. In SAXS/GISAXS experiments, the intensity of the scattering pattern contributes to the scattering-invariant parameters of phase separation efficiency and intermolecular structure dispersity. This underscores the importance of accurate corrections, even in small-angle scattering, despite the reduced impact of most corrections due to the large sample-to-detector distances. Since many of these corrections are independent of the angle of incidence, *pygid* applies them to the raw scattering pattern prior to conversion. A list of the *pygid* corrections, along with their input parameters and types, is presented in Table 3.

(i) Flat-field correction accounts for the varying sensitivity of different detector pixels. The creation of the corresponding correction matrix is not a part of the *pygid* workflow but can be performed *e.g.* using the approach described by Weng *et al.* (2023). The matrix is then loaded as a 2D array.

(ii) Dark-current correction involves subtracting the signal recorded in the absence of the X-ray beam. The correction matrix should be loaded as a 2D array.

(iii) Solid-angle correction accounts for the geometric effects arising from the proportional relationship between the signal detected by a pixel and its corresponding solid angle $\Delta\Omega$,

$$\Delta\Omega = \frac{A_{\rm px}\cos\alpha}{\mathrm{SPD}^2} \propto \cos^3\alpha = \mathrm{CorrMatrix}_{\rm SA}, \qquad (1)$$

where $A_{\rm px}$ is the pixel area, SPD is the sample-to-pixel distance, and α is the angle between the normal vector to the

Table 4
Comparison of processing time distribution (in ms) across different computational stages.

Raw scattering patterns in HDF5 format with a resolution of 2162 × 2068 pixels were used for testing.

Computational stage	Office PC	ESRF VISA cluster	DESY Maxwell cluster
Coordinate map calculation	530.88 ± 22.89	455.65 ± 58.75	271.41 ± 2.3
Correction map calculation (with MP)	607.20 ± 22.59 (332.34 ± 26.95)	531.73 ± 106.43 (87.09 ± 3.38)	406.98 ± 1.19 (79.61 ± 0.48)
q-range determination	22.63 ± 2.74	13.61 ± 3.71	10.79 ± 0.07
Total time for preliminary calculations	1160.7 ± 32.3	1001.0 ± 121.6	689.18 ± 2.56
Raw data loading	40.95 ± 5.09	30.71 ± 3.78	27.44 ± 0.32
Conversion (with MP)	14.71 ± 0.94 (15.48 ± 0.26)	3.43 ± 1.14 (2.81 ± 0.17)	3.73 ± 0.32 (2.55 ± 0.09)
Data saving	47.55 ± 9.74	49.39 ± 2.0	73.71 ± 24.73
Total time per pattern	103.21 ± 11.03	83.53 ± 4.43	104.88 ± 24.73

pixel surface **n** and the wavevector $\mathbf{k}_f$ corresponding to that pixel position.

(iv) Polarization correction accounts for the polarization of the incident X-ray beam. The scattered intensity depends on the angle between the polarization of the incident and scattered waves, following a squared cosine dependence. This can be described by the horizontal (γ) and vertical (δ) scattering angles in both in-plane and out-of-plane scattering geometries,

$$\mathrm{CorrMatrix_P} = \zeta P_\mathrm{h} + (1-\zeta)\,P_\mathrm{v}, \tag{2}$$

where the subscripts h and v refer to, respectively, horizontal and vertical components and

$$P_\mathrm{h} = 1 - \cos^2\delta \sin^2\gamma, \tag{3}$$

$$P_\mathrm{v} = \cos^2\delta. \tag{4}$$

The polarization parameter ζ is approximately equal to 1 for typical synchrotron radiation (horizontal polarization) and 0.5 for an unpolarized laboratory X-ray tube. Users are required to specify the ζ parameter using the `pol_type` key.

(v) Air attenuation correction is based on the Beer–Lambert extinction law and arises due to the varying X-ray beam paths to each pixel. The linear attenuation coefficient (μ_air) depends on the X-ray energy and air density and must be provided by the user, while the sample-to-pixel distance (SPD) is derived from the coordinate maps:

$$\mathrm{CorrMatrix_{AAtt}} = \exp(-\mu_\mathrm{air} \times \mathrm{SPD}) \propto \exp\left(\frac{-\mu_\mathrm{air}}{\cos\alpha}\right). \tag{5}$$

(vi) Sensor attenuation and sample absorption corrections are based on the X-ray beam path through the detector sensor and sample. Linear attenuation coefficients and thicknesses are required for these calculations. For a more detailed description of the correction process and its mathematical aspects, we refer the reader to Gasser *et al.* (2025).

(vii) The Lorentz correction is related to the distribution of the Bragg peaks on circles/spheres with different radii in reciprocal space (Jiang, 2015). The usage of this correction in *pygid* is limited to the most common thin-film 2D and 3D powder-like cases, which can be chosen using `powder_dim` = 2 or 3, respectively:

$$\mathrm{CorrMatrix_L}(\texttt{power\_dim} = 3) = \frac{1}{\sin^2\Theta\cos\Theta}, \tag{6}$$

$$\mathrm{CorrMatrix_L}(\texttt{power\_dim} = 2) = \frac{1}{\sin\gamma^\mathrm{smpl}}. \tag{7}$$

Here γ^smpl is the horizontal scattering angle and 2Θ is the total scattering angle in the SCS.

5. Performance

To assess the performance of the *pygid* package, we conducted benchmark tests on three computing systems with different capabilities: a typical office desktop PC (Intel i5-6500, four cores, 16 GB RAM, Windows 10), the ESRF VISA cluster (AMD CPU, 32 cores, 128 GB RAM; https://visa.readthedocs.io/en/latest/) and the DESY Maxwell cluster (AMD CPU, 48 cores, 512 GB RAM; https://docs.desy.de/maxwell/). As a test dataset, we used an HDF5 file with a single GIWAXS pattern of diindenoperylene (DIP), acquired with an EIGER2 X CdTe 4M detector (2162 × 2068 pixels) on the ID10-SURF beamline at the ESRF. The X-ray energy was E = 20 keV. To test the multiprocessing mode (MP) during conversion and coordinate map calculations, we used an HDF5 file containing 13 scattering patterns measured at different angles of incidence (ranging from 0.04° to 0.1°) using the same experimental setup. For each angle, a separate coordinate map was calculated in MP mode. Preliminary calibration was based on the lanthanum hexaboride (LaB_6) scattering pattern using the *pyFAI* GUI (Ashiotis *et al.*, 2015). Each performance test was averaged over 100 runs.

Table 4 summarizes the execution times for different stages of the *pygid* workflow. The initial stage of preliminary calculations, which includes pixel position transformation into cylindrical coordinates, generation of polarization and solid-angle correction maps, and determination of converted axis ranges, takes approximately 1161 ± 32 ms on an office PC. This step is slightly faster on the VISA and Maxwell clusters, requiring 1001 ± 122 and 689 ± 3 ms, respectively. Here, the calculated q_{xy} and q_z ranges were [0, 4.32] and [0, 4.06], respectively. The resolution was determined automatically on the basis of the pixel size. As a result, the converted image had a size of 1749 × 1861 pixels. The second stage, performed for each scattering pattern, involves raw data loading, conversion with linear pixel interpolation and saving into an HDF5 file, taking 103 ± 11 and 105 ± 25 ms on the office PC and

Maxwell cluster, respectively, while the VISA cluster reduced this processing time to 84 ± 4 ms. As the file input/output performance may vary depending on the storage system and specific configuration, the following analysis focuses on the core image conversion functionality and the coordinate map computation, which represents the most computationally demanding step.

Thus, the execution time of the coordinate map calculation scales nearly linearly with the resolution, *i.e.* the size of the converted image (Fig. S1 and Table 4). The highest calculation speed was observed on the Maxwell cluster. When multiple coordinate maps were calculated in multiprocessing mode, the processing speed increased significantly, by up to 45%, 83% and 81% for the PC and the VISA and Maxwell clusters, respectively.

The conversion time depends on both the raw scattering pattern size [Figs. 5(*a*) and 5(*b*)] and the chosen resolution [Figs. 5(*c*) and 5(*d*)]. The benefits of multiprocessing become apparent primarily on cluster systems: the conversion time was reduced by up to 34% on the VISA cluster and up to 43% on the Maxwell cluster.

As previously mentioned, the conversion speed can also be influenced by the interpolation method [Figs. 5(*e*) and 5(*f*)]. For conversion to cylindrical coordinates no significant

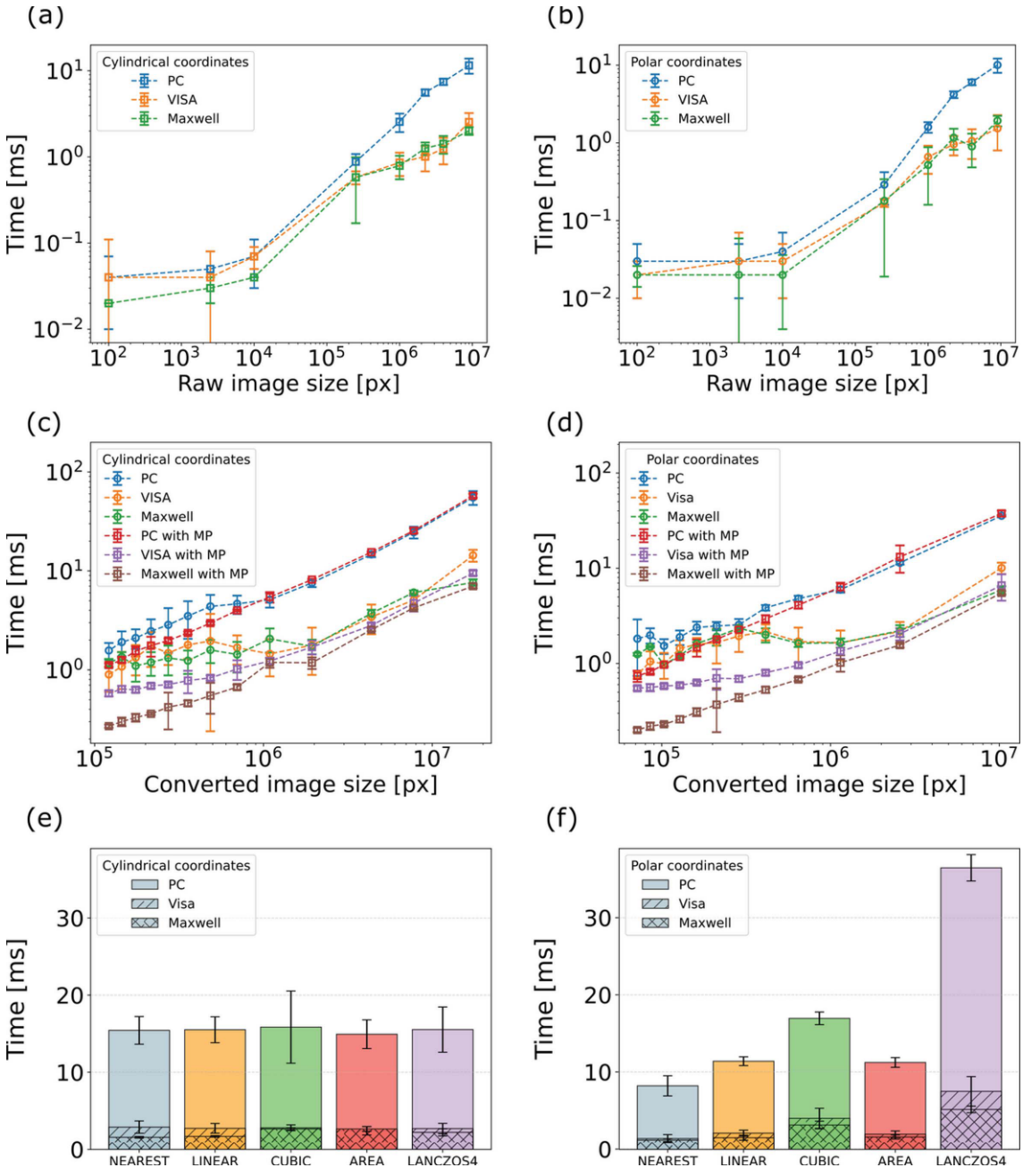

Figure 5
Performance results of the *pygid* package, showing the dependence of the conversion time on (*a*) and (*b*) raw pattern size, (*c*) and (*d*) resolution, and (*e*) and (*f*) interpolation type for (*a*), (*c*) and (*e*) cylindrical and (*b*), (*d*) and (*f*) polar coordinates.

differences were observed. On the other hand, for polar coordinates, nearest-neighbor interpolation resulted in faster processing, whereas more complex methods, such as cubic and Lanczos (Cullum & Willoughby, 2002), significantly slowed down the conversion. The differences between interpolation methods are illustrated in Fig. S2. In our previous work, we demonstrated that the interpolation can introduce artifacts in the low-q region, which consequently affect the ML-based peak detection process (Völter *et al.*, 2025). From this perspective, the pixel-splitting approach implemented in the *pyFAI* package is more suitable. However, interpolation remains crucial in the high-q region and near the missing wedge, where some bins may be empty after conversion. These differences may become more pronounced for lower-resolution images. Therefore, users are encouraged to choose the interpolation method that best suits their specific case, considering resolution, processing time and the region of interest.

6. Data reduction and simulation examples

As a demonstration of the capabilities of the *pygid* package, we present three examples of data obtained using different detectors at various X-ray sources, each with distinct raw data formats. Fig. 6 shows the raw scattering patterns alongside their representations in cylindrical and polar coordinates after transformation.

The first example features the GIWAXS pattern of a DIP thin film acquired on the ESRF ID10 beamline (X-ray beam energy E = 20 keV). The scattering data were recorded using an EIGER2 X CdTe 4M detector (2162 × 2068 pixels) and saved in the NeXus format. A Python-based script for coordinate transformation and expected peak position simulation

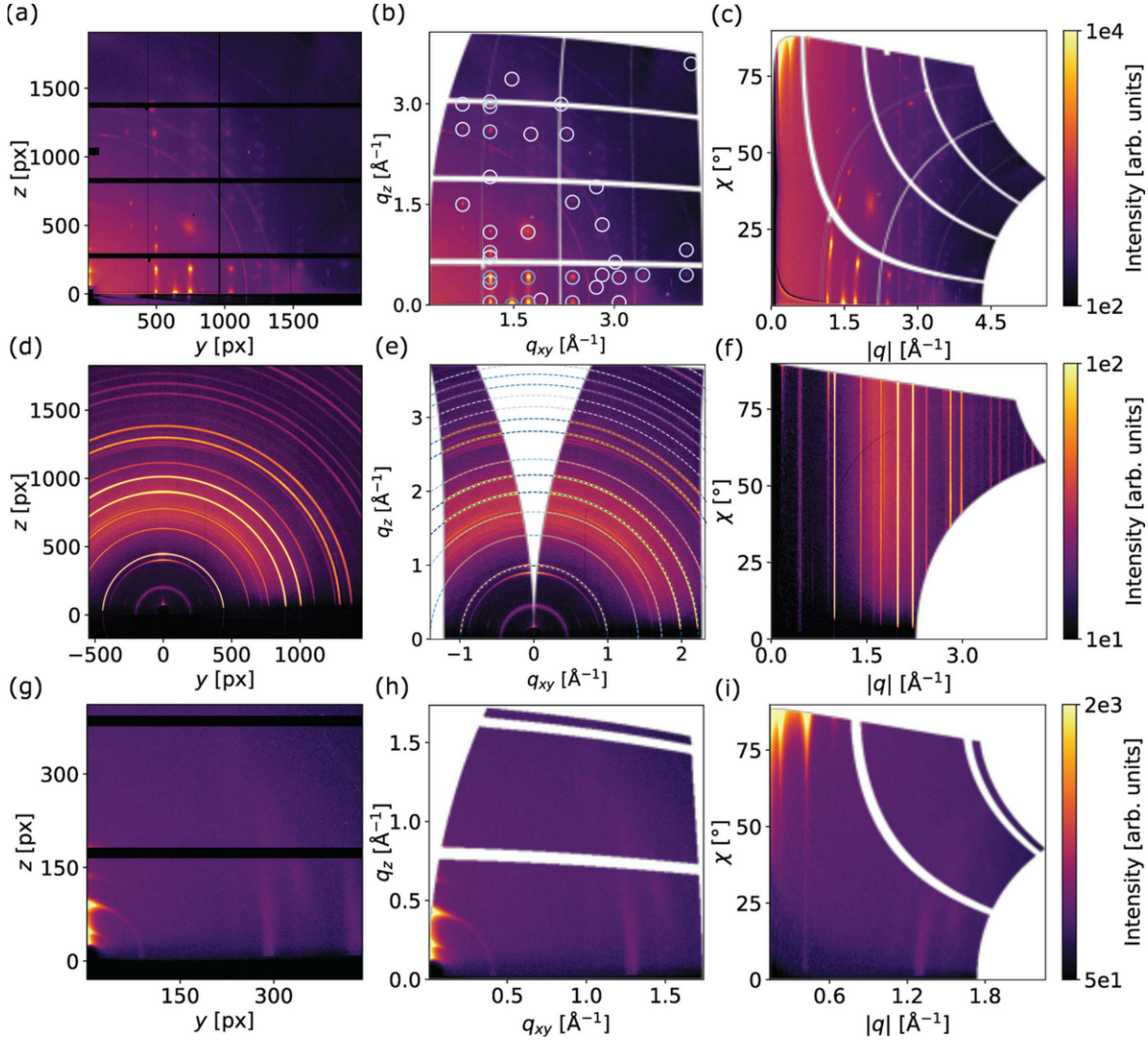

Figure 6
pygid package usage examples. (*a*)–(*c*) GIWAXS patterns of a DIP thin film measured on the ESRF ID10 beamline (EIGER2 X CdTe 4M detector). (*d*)–(*f*) GIWAXS patterns of $MAPbI_3$ measured on the DESY P08 beamline (PerkinElmer flat-panel detector). (*g*)–(*i*) GIWAXS patterns of Pb nanoparticles measured using a laboratory scattering setup (Pilatus 300k detector). For each dataset, (*a*), (*d*), (*g*) raw scattering patterns, (*b*), (*e*), (*h*) converted images in cylindrical coordinates and (*c*), (*f*), (*i*) converted images in polar coordinates. The simulation of the scattering patterns was done using the *pygidSIM* package (blue rings). The color of the simulated data is proportional to the structure factor.

Table 5
Overview of key steps in *pygid* data reduction workflow.

Code	Description
`>> import pygid`	Import of the package
`>> params = pygid.ExpParams( poni_path='LaB6.poni', mask_path='mask.npy', ai=0.075)`	Creation of the `ExpParams` class instance from the loaded PONI file and mask, and input of incident angle
`>> matrix = pygid.CoordMaps(params)`	Creation of the `CoordMaps` class instance
`>> exp_metadata = pygid.ExpMetadata( start_time="2024-03-17T14:30:00Z", source_type="synchrotron", source_name="ESRF", detector="eiger4m", instrument_name="ID10")` `>> smpl_metadata = pygid.SampleMetadata( path_to_load="DIP_metadata.yaml")`	Metadata definition; an example of the sample metadata is shown in Table S3
`>> analysis = pygid.Conversion( matrix=matrix, path="DIP.h5", dataset="/measurement/eiger4m")`	Creation of the `Conversion` class instance and a raw scattering pattern loading
`>> analysis.det2q_gid( plot_result=True, save_result=True, path_to_save="DIP_result.h5", exp_metadata=exp_metadata, smpl_metadata=smpl_metadata)`	Image conversion to GID cylindrical coordinates, plotting and saving the result with metadata
`>> analysis.make_simulation( path_to_cif="DIP_structure.cif", orientation=[0, 0, 1], min_int=1e-3, plot_result=True, return_result=True)`	Simulation based on CIF and given orientation, plotting with the experimental scattering pattern and returning Miller indices with positions

is provided in Table 5 as an example of *pygid* package usage. The resulting pattern reveals a highly oriented 2D DIP powder structure [Figs. 6(*a*) to 6(*c*)]. For simulations using *pygidSIM*, a monoclinic σ-phase oriented along the [001] direction was used (Heinrich *et al.*, 2007). Only peaks with intensities exceeding 0.1% of the maximum are shown in the figure.

Secondly, a typical GIWAXS pattern of methylammonium perovskite $MAPbI_3$ is shown (Kneschaurek *et al.*, 2023). The experiment was conducted on the PETRA III P08 beamline using a PerkinElmer flat-panel detector (2048 × 2048 pixels). The X-ray energy was $E = 18$ keV. The raw data were saved as TIFF format. The tetragonal perovskite structure appears to be randomly oriented [Figs. 6(*d*) to 6(*f*)] (Druzbicki *et al.*, 2023). The rings and arcs observed at 0.45, 0.51 and 0.65 $Å^{-1}$ are not simulated, as they originate from a Pb_3I_8 intermediate complex.

Finally, clusters of PbTe nanoplatelets were measured in GID geometry using an in-house X-ray scattering setup (Xeuss 2.0, Xenocs, X-ray beam energy $E = 8$ keV) equipped with a Pilatus 300k detector (487 × 619 pixels). The experiment produced raw data in EDF format. Bragg reflections in the horizontal and vertical directions are observed, indicating the formation of a superlattice from stacked nanoplatelets (Biesterfeld *et al.*, 2024) [Figs. 6(*g*) to 6(*i*)].

The presented examples highlight the usefulness of *pygid* in handling diverse file formats generated using different experimental setups and detectors, including a flat-panel PerkinElmer detector. The integration of the *pygidSIM* package provides a user-friendly framework for simulating the positions and intensities of Bragg reflections and diffraction rings from CIFs.

7. Summary and conclusions

The present work introduces the new *pygid* package, developed to address the increasing demand for fast and reliable data reduction of 2D scattering data. As the volume of X-ray and neutron scattering data continues to grow – particularly at high-throughput synchrotron and laboratory facilities – there is a critical need for automated and efficient processing tools. *pygid* is specifically designed for batch conversion and simplifies this process for users with varying levels of expertise. It supports raw data loading across all common formats used in both laboratory and synchrotron environments.

The package is applicable to both small- and wide-angle scattering, in grazing-incidence and transmission geometries. It enables conversion to 2D Cartesian, polar and pseudo-polar reciprocal-space coordinates and provides radial and azimuthal integration routines. The average conversion speed (14.71 ± 0.94 ms per image) is superior to that of other existing tools of 2D remapping and can be further reduced by tuning the resolution, selecting appropriate interpolation methods and using cluster computing. To ensure reliable

quantitative analysis, a wide range of intensity corrections have been implemented. In addition, the *pygidSIM* package is included to simulate scattering patterns based on crystallographic information, enabling direct comparison with experimental data. As part of our *mlgid* pipeline, the package provides an interface between raw experimental data and machine-learning-based tools for peak detection, labeling and structure determination.

Future developments will focus on GPU acceleration and enhanced multiprocessing to increase performance further. We also plan to integrate *pygid* directly into beamline workflows at synchrotron facilities, enabling *in situ* data conversion immediately after acquisition, which will help provide rapid feedback during measurements.

APPENDIX *A*

Equations used for coordinate map calculations

From cylindrical ($q_{xy}^{\rm smpl}, q_z^{\rm smpl}$) to 3D Cartesian ($q_x^{\rm smpl}, q_y^{\rm smpl}, q_z^{\rm smpl}$) coordinates in the SCS for a given angle of incidence $\alpha_{\rm i}$:

$$q_x^{\rm smpl} = \frac{-[(q_{xy}^{\rm smpl})^2 + (q_z^{\rm smpl})^2]}{2k} + q_z^{\rm smpl}\tan(\alpha_{\rm i}), \tag{8}$$

$$q_y^{\rm smpl} = -{\rm sgn}(q_{xy}^{\rm smpl})\sqrt{(q_{xy}^{\rm smpl})^2 - (q_x^{\rm smpl})^2}, \tag{9}$$

$$q_z^{\rm smpl} = q_z^{\rm smpl}, \tag{10}$$

where k is the wavevector magnitude.

From 3D Cartesian ($q_x^{\rm smpl}, q_y^{\rm smpl}, q_z^{\rm smpl}$) to cylindrical ($q_{xy}^{\rm smpl}, q_z^{\rm smpl}$) in the SCS:

$$q_{xy}^{\rm smpl} = -{\rm sgn}(q_y^{\rm smpl})\sqrt{(q_x^{\rm smpl})^2 + (q_y^{\rm smpl})^2}, \tag{11}$$

$$q_z^{\rm smpl} = q_z^{\rm smpl}. \tag{12}$$

From polar ($q_{\rm abs}^{\rm smpl}$, χ) and pseudo-polar ($q_{\rm abs}^{\rm smpl}$, $q_{\rm abs}^{\rm smpl}\chi$) to cylindrical ($q_{xy}^{\rm smpl}, q_z^{\rm smpl}$) in the SCS:

$$q_{xy}^{\rm smpl} = q_{\rm abs}^{\rm smpl}\cos\chi, \tag{13}$$

$$q_z^{\rm smpl} = q_{\rm abs}^{\rm smpl}\sin\chi. \tag{14}$$

From 2D Cartesian ($q_1^{\rm lab}, q_2^{\rm lab}$) to 3D Cartesian ($q_x^{\rm lab}, q_y^{\rm lab}, q_z^{\rm lab}$) in the LCS:

$$q_x^{\rm lab} = -\frac{(q_1^{\rm lab})^2 + (q_2^{\rm lab})^2}{2k}, \tag{15}$$

$$q_y^{\rm lab} = {\rm sgn}(q_1^{\rm lab})\frac{\sqrt{(q_1^{\rm lab})^2 + (q_2^{\rm lab})^2 - (q_x^{\rm lab})^2}}{\sqrt{1 + (q_2^{\rm lab}/q_1^{\rm lab})^2}}, \tag{16}$$

$$q_z^{\rm lab} = {\rm sgn}(q_2^{\rm lab})\frac{\sqrt{(q_1^{\rm lab})^2 + (q_2^{\rm lab})^2 - (q_x^{\rm lab})^2}}{\sqrt{1 + (q_1^{\rm lab}/q_2^{\rm lab})^2}}. \tag{17}$$

From 3D Cartesian ($q_x^{\rm lab}, q_y^{\rm lab}, q_z^{\rm lab}$) to 2D Cartesian ($q_1^{\rm lab}$, $q_2^{\rm lab}$) in the LCS:

$$q_1^{\rm lab} = q_y^{\rm lab}\sqrt{\frac{1}{1 + q_x^{\rm lab}/(2k)}}, \tag{18}$$

$$q_2^{\rm lab} = q_z^{\rm lab}\sqrt{\frac{1}{1 + q_x^{\rm lab}/(2k)}}. \tag{19}$$

From polar ($q_{\rm abs}^{\rm lab}$, χ) and pseudo-polar ($q_{\rm abs}^{\rm lab}$, $q_{\rm abs}^{\rm lab}\chi$) to 3D Cartesian ($q_x^{\rm lab}, q_y^{\rm lab}, q_z^{\rm lab}$) to 2D Cartesian ($q_1^{\rm lab}, q_2^{\rm lab}$) in the LCS:

$$q_x^{\rm lab} = -\frac{(q_{\rm abs}^{\rm lab})^2}{2k}, \tag{20}$$

$$q_y^{\rm lab} = \sqrt{(q_{\rm abs}^{\rm lab})^2 - (q_x^{\rm lab})^2}\cos\chi, \tag{21}$$

$$q_z^{\rm lab} = \sqrt{(q_{\rm abs}^{\rm lab})^2 - (q_x^{\rm lab})^2}\sin\chi. \tag{22}$$

The relations between the SCS ($\mathbf{q}^{\rm smpl}$) and LCS ($\mathbf{q}^{\rm lab}$) are

$$\mathbf{q}^{\rm lab} = \hat{R}_y(-\alpha_{\rm i})\,\mathbf{q}^{\rm smpl}, \tag{23}$$

$$\mathbf{q}^{\rm smpl} = \hat{R}_y(\alpha_{\rm i})\,\mathbf{q}^{\rm lab}. \tag{24}$$

The relation between the LCS ($\mathbf{q}^{\rm lab}$) and detector pixel positions (p_1, p_2) is

$$\mathbf{q}^{\rm lab} + \mathbf{k}_{\rm i} = \mathbf{k}_{\rm f} \propto \mathbf{d}^* = \widehat{\mathbb{R}}(-\Theta_1, \Theta_2, -\Theta_3)\,\mathbf{d}, \tag{25}$$

where $\widehat{\mathbb{R}}(-\Theta_1, \Theta_2, -\Theta_3) = \widehat{\mathbb{R}}_x(-\Theta_1)\,\widehat{\mathbb{R}}_y(\Theta_2)\,\widehat{\mathbb{R}}_z(-\Theta_3)$, $\widehat{\mathbb{R}}_i(\Theta)$ is a rotation matrix over i axes of the LCS to Θ, and $\mathbf{d}$ is a pixel position in the DCS that is related to the pixel coordinates via the pixel size $p_{\rm size}$ and the sample projection onto the detector plane (`poni1`, `poni2`):

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} p_1 p_{\rm size} - {\tt poni1} \\ -p_1 p_{\rm size} + {\tt poni2} \\ {\rm SDD} \end{bmatrix}. \tag{26}$$

Acknowledgements

We acknowledge the European Synchrotron Radiation Facility (ESRF) for the provision of synchrotron radiation facilities under proposal No. SC-5641 at the beamline ID10-SURF. We also acknowledge DESY (Hamburg, Germany), a member of the Helmholtz Association HGF, for the provision of experimental facilities. Parts of this research were carried out at PETRA III at the beamline P08. Beamtime was allocated for proposal I-20231315. Computational resources were provided in part by the Maxwell HPC Cluster (DESY) and the VISA cluster (ESRF). Open access funding enabled and organized by Projekt DEAL.

Conflict of interest

The authors have no conflicts to disclose.

Data availability

The supplementary materials contain recommendations for metadata fields and additional performance testing results. The *pygid* package is open source and freely available on the GitHub page https://github.com/mlgid-project, along with additional tools for simulation, labeling and machine-learning-based analysis of GID data. The examples presented in the article are available at https://doi.org/10.5281/zenodo.17466183. Additional data supporting the findings of this study are available from the corresponding authors upon reasonable request.

Funding information

Financial support was provided via DAPHNE4NFDI (German Research Foundation DFG project 460248799) and VIPR (German Federal Ministry for Science and Education project 05D23VT1 ErUM-Data). We thank the Cluster of Excellence – Machine Learning for Science, funded by the DFG under Germany's Excellence Strategy (EXC No. 2064/1, project No. 390727645), for support and access to computer resources. This project was also funded by the DFG project No. 557510229. We acknowledge the OSCARS project, which has received funding from the European Commission's Horizon Europe Research and Innovation programme under grant agreement No. 101129751.

References

Alted, F., Vilata, I., Prater, S., Mas, V., Hedley, T., Valentino, A., Whitaker, J., Moore, J., Scopatz, A., Bedini, A., Kooij, T. & Sancho, J. (2002). *PyTables*, https://www.pytables.org/.

Amelung, L., Barty, A., Murphy, B. M., Grunwaldt, J.-D., Hövelmann, S., Leonau, A., Paripsa, S., Schneidewind, A., Busch, S., Gutt, C., Lohstroh, W., Schreiber, F. & Unruh, T. (2025). *J. Phys. Conf. Ser.* **3010**, 012133.

Arias, J. J. R., Mota, I. C., Albuquerque, L. S., Dahmouche, K. & Marques, M. F. V. (2021). *J. Mater. Sci.* **33**, 1838–1850.

Ashiotis, G., Deschildre, A., Nawaz, Z., Wright, J. P., Karkoulis, D., Picca, F. E. & Kieffer, J. (2015). *J. Appl. Cryst.* **48**, 510–519.

Banerjee, R., Kowarik, S. & Schreiber, F. (2021). *Advanced characterization of nanostructured materials: probing the structure and dynamics with synchrotron X-rays and neutrons*, edited by S. K. Sinha, M. K. Sanyal & C.-K. Loong, pp. 49–96. Singapore: World Scientific.

Barrit, D., Tang, M.-C., Munir, R., Li, R., Zhao, K. & Smilgies, D.-M. (2022). *Appl. Mater. Interfaces* **14**, 26315–26326.

Barty, A., Gutt, C., Lohstroh, W., Murphy, B., Schneidewind, A., Grunwaldt, J.-D., Schreiber, F., Busch, S., Unruh, T., Bussmann, M., Fangohr, H., Görzig, H., Houben, A., Kluge, T., Manke, I., Lützenkirchen-Hecht, D., Schneider, T. R., Weber, F., Bruno, G., Einsle, O., Felder, C., Herzig, E. M., Konrad, U., Markötter, H., Rossnagel, K., Sheppard, T. & Turchinovich, D. (2023). *DAPHNE4NFDI – consortium proposal*, https://doi.org/10.5281/zenodo.8040606.

Bein, B., Hsing, H.-C., Callori, S. J., Sinsheimer, J., Chinta, P. V., Headrick, R. L. & Dawber, M. (2015). *Nat. Commun.* **6**, 10136.

Biesterfeld, L., Vochezer, M. T., Kögel, M., Zaluzhnyy, I. A., Rosebrock, M., Klepzig, L. F., Leis, W., Seitz, M., Meyer, J. C. & Lauth, J. (2024). *Chem. Mater.* **36**, 7197–7206.

Bommel, S., Kleppmann, N., Weber, C., Spranger, H., Schäfer, P., Novák, J., Roth, S., Schreiber, F., Klapp, S. & Kowarik, S. (2014). *Nat. Commun.* **5**, 5388.

Bradski, G. (2000). *Dr Dobb's J. Softw. Tools* **25**, 120–125.

Breiby, D. W., Bunk, O., Andreasen, J. W., Lemke, H. T. & Nielsen, M. M. (2008). *J. Appl. Cryst.* **41**, 262–271.

Collette, A., Kluyver, T., Caswell, T. A., Tocknell, J., Kieffer, J., Jelenak, A., Scopatz, A., Dale, D., Chen, Vincent, T., Einhorn, M., Payno, Garriga, J., Sciarelli, P., Valls, V., Ghosh, S., Pedersen, U. K., Kittisopikul, M., Kirkham, J., Raspaud, M., Danilevski, C., Abbasi, H., Readey, J., Mühlbauer, K., Paramonov, A., Chan, L., De Schepper, R., Solé, V. A., Jialin & Guest, D. H. (2023). *h5py/h5py: 3.8.0*, https://doi.org/10.5281/zenodo.7560547.

Cullum, J. K. & Willoughby, R. A. (2002). *Lanczos algorithms for large symmetric eigenvalue computations*, Vol. I, *Theory*. SIAM.

Diao, Y., Shaw, L., Bao, Z. & Mannsfeld, S. C. (2014). *Energy Environ. Sci.* **7**, 2145–2159.

Dosch, H. (1992). *Critical phenomena at surfaces and interfaces: evanescent X-ray and neutron scattering*. Heidelberg: Springer.

Drużbicki, K., Gila-Herranz, P., Marin-Villa, P., Gaboardi, M., Armstrong, J. & Fernandez-Alonso, F. (2024). *Cryst. Growth Des.* **24**, 391–404.

Eisenberger, P. & Marra, W. C. (1981). *Phys. Rev. Lett.* **46**, 1081–1084.

Eres, G., Rouleau, C. M., Lu, Q., Zhang, Z., Benda, E., Lee, H. N., Tischler, J. Z. & Fong, D. D. (2019). *Rev. Sci. Instrum.* **90**, 093902.

Feidenhans'l, R. (1989). *Surf. Sci. Rep.* **10**, 105–188.

Ferrer, P., Rubio-Zuazo, J., Heyman, C., Esteban-Betegón, F. & Castro, G. R. (2013). *J. Synchrotron Rad.* **20**, 474–481.

Gasser, F., Simbrunner, J., Huck, M., Moser, A., Steinrück, H.-G. & Resel, R. (2025). *J. Appl. Cryst.* **58**, 96–106.

Gonzalez, R. C. & Woods, R. E. (2018). *Digital image processing*. London: Pearson.

Gu, X., Shaw, L., Gu, K., Toney, M. F. & Bao, Z. (2018). *Nat. Commun.* **9**, 534.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *Acta Cryst.* A**47**, 655–685.

Hamilton, W., Butler, P., Baker, S., Smith, G., Hayter, J. B., Magid, L. & Pynn, R. (1994). *Phys. Rev. Lett.* **72**, 2219–2222.

Heinrich, M. A., Pflaum, J., Tripathi, A. K., Frey, W., Steigerwald, M. L. & Siegrist, T. (2007). *J. Phys. Chem. C* **111**, 18878–18881.

Hodas, M., Siffalovic, P., Nádaždy, P., Mrkyvková, N., Bodík, M., Halahovets, Y., Duva, G., Reisz, B., Konovalov, O., Ohm, W., Jergel, M., Majková, E., Gerlach, A., Hinderhofer, A. & Schreiber, F. (2018). *ACS Appl. Nano Mater.* **1**, 2819–2826.

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

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

Jones, R. L., Kumar, S. K., Ho, D. L., Briber, R. M. & Russell, T. P. (1999). *Nature* **400**, 146–149.

Ju, G., Xu, D., Thompson, C., Highland, M. J., Eastman, J. A., Walkosz, W., Zapol, P. & Stephenson, G. B. (2021). *Nat. Commun.* **12**, 1721.

Kaune, G., Ruderer, M. A., Metwalli, E., Wang, W., Couet, S., Schlage, K., Röhlsberger, R., Roth, S. V. & Müller-Buschbaum, P. (2009). *Appl. Mater. Interfaces* **1**, 353–360.

Klosowski, P., Koennecke, M., Tischler, J. Z. & Osborn, R. (1997). *J. Open Source Softw.* **241**, 151–153.

Kneschaurek, E., Hinderhofer, A., Hofferberth, B., Scheffczyk, N., Pithan, L., Zimmermann, P., Merten, L., Bertram, F. & Schreiber, F. (2023). *Rev. Sci. Instrum.* **94**, 063901.

Knudsen, E. B., Sørensen, H. O., Wright, J. P., Goret, G. & Kieffer, J. (2013). *J. Appl. Cryst.* **46**, 537–539.

Könnecke, M., Akeroyd, F. A., Bernstein, H. J., Brewster, A. S., Campbell, S. I., Clausen, B., Cottrell, S., Hoffmann, J. U., Jemian, P. R., Männicke, D., Osborn, R., Peterson, P. F., Richter, T., Suzuki, J., Watts, B., Wintersberger, E. & Wuttke, J. (2015). *J. Appl. Cryst.* **48**, 301–305.

Kowarik, S., Gerlach, A., Sellner, S., Schreiber, F., Cavalcanti, L. & Konovalov, O. (2006). *Phys. Rev. Lett.* **96**, 125504.

Lapkin, D., Nasro, R., Hagara, J., Hofferberth, B., Hinderhofer, A., Gerlach, A. & Schreiber, F. (2025). *Rev. Sci. Instrum.* **96**, 053905.
Lohstroh, W., Weber, F., Busch, S., Görzig, H., Murphy, B., Coan, P., Fahad, H., Osterhoff, M., Tymoshenko, Y., Paripsa, S., Schneidewind, A., & Herb, C (2024). *DAPHNE4NFDI – draft recommendations on metadata capture and specifications*, https://doi.org/10.5281/zenodo.12169110.
Magnussen, O. M., Drnec, J., Qiu, C., Martens, I., Huang, J. J., Chattot, R. & Singer, A. (2024). *Chem. Rev.* **124**, 629–721.
Manley, E. F., Strzalka, J., Fauvell, T. J., Jackson, N. E., Leonardi, M. J., Eastham, N. D., Marks, T. J. & Chen, L. X. (2017). *ACS Nano* **29**, 1703933.
McLeod, R., Alted, F., Valentino, A., de Menten, G., Wiebe, M., cgohlke , Bedini, A., mamrehn, techtonik, a., Erb, S., Shadchin, A., Bunin, A., Kooij, T., Pavlyk, O., Ben Jelloul, M., Garrison, L., Hurtado, D. M., Carey, C. J., Sarahan, M., Cox, D., Plesivčak, Z., Borgdorff, J., Courbet, C., Dickinson, M., Leitao, B., de Laat, A., Pitrou, A., Portnoy, A., Ortega, A. L. & Böhn, A. (2018). *pydata/numexpr: Numexpr v2.6.9*, https://doi.org/10.5281/zenodo.2483274.
Müller-Buschbaum, P. (2013). *Polym. J.* **45**, 34–42.
Müller-Buschbaum, P. (2014). *Adv. Mater.* **26**, 7692–7709.
Müller-Buschbaum, P., Cubitt, R. & Petry, W. (2003). *Langmuir* **19**, 7778–7782.
Mundt, L. E. & Schelhas, L. T. (2020). *Adv. Energy Mater.* **10**, 1903074.
Nagpal, A. & Gabrani, G. (2019). *2019 Amity international conference on artificial intelligence (AICAI)*, pp. 140–145. New York: IEEE.
Nicklin, C., Martinez-Hardigree, J., Warne, A., Green, S., Burt, M., Naylor, J., Dorman, A., Wicks, D., Din, S. & Riede, M. (2017). *Rev. Sci. Instrum.* **88**, 103901.
Pauw, B. R., Smith, A. J., Snow, T., Terrill, N. J. & Thünemann, A. F. (2017). *J. Appl. Cryst.* **50**, 1800–1811.
Pierce, D. (2025). *ncdf4: Interface to Unidata netCDF (Version 4 or earlier) format data files.* R package Version 1.24. https://CRAN.R-project.org/package=ncdf4.
Pithan, L., Starostin, V., Mareček, D., Petersdorf, L., Völter, C., Munteanu, V., Jankowski, M., Konovalov, O., Gerlach, A., Hinderhofer, A., Murphy, B., Kowarik, S. & Schreiber, F. (2023). *J. Synchrotron Rad.* **30**, 1064–1075.
Posselt, D., Zhang, J., Smilgies, D.-M., Berezkin, A. V., Potemkin, I. I. & Papadakis, C. M. (2017). *Prog. Polym. Sci.* **66**, 80–115.
Reus, M. A., Reb, L. K., Kosbahn, D. P., Roth, S. V. & Müller-Buschbaum, P. (2024). *J. Appl. Cryst.* **57**, 509–528.
Richard, M.-I., Highland, M., Fister, T., Munkholm, A., Mei, J., Streiffer, S., Thompson, C., Fuoss, P. & Stephenson, G. (2010). *Appl. Phys. Lett.* **96**, 051911.
Richter, L. J., DeLongchamp, D. M. & Amassian, A. (2017). *Chem. Rev.* **117**, 6332–6366.
Robinson, I. & Tweet, D. (1992). *Rep. Prog. Phys.* **55**, 599–651.
Romodin, M., Starostin, V., Lapkin, D., Hinderhofer, A., & Schreiber, F. (2025). *mlgid-project/pygidSIM: v0.1.1*, https://doi.org/10.5281/zenodo.17609569.
Saabith, A., Fareez, M. & Vinothraj, T. (2019). *Int. J. Adv. Eng. Res. Dev.* **6**(10), 6–12.
Savikhin, V., Steinrück, H.-G., Liang, R.-Z., Collins, B. A., Oosterhout, S. D., Beaujuge, P. M. & Toney, M. F. (2020). *J. Appl. Cryst.* **53**, 1108–1129.
Schlipf, J. & Müller-Buschbaum, P. (2017). *Adv. Energy Mater.* **7**, 1700131.
Smilgies, D.-M. (2022). *J. Polym. Sci.* **60**, 1023–1041.
Smilgies, D.-M. (2025). *Crystals* **15**, 63.
Smilgies, D.-M. & Blasini, D. R. (2007). *J. Appl. Cryst.* **40**, 716–718.
Smilgies, D.-M., Busch, P., Papadakis, C. M. & Posselt, D. (2002). *Synchrotron Radiat. News* **15**(5), 35–42.
Smilgies, D. M. & Li, R. (2021). *ChemRxiv*, https://doi.org/10.26434/chemrxiv-2021-j1bww.
Starostin, V., Munteanu, V., Greco, A., Kneschaurek, E., Pleli, A., Bertram, F., Gerlach, A., Hinderhofer, A. & Schreiber, F. (2022*a*). *npj Comput. Mater.* **8**, 101.
Starostin, V., Pithan, L., Greco, A., Munteanu, V., Gerlach, A., Hinderhofer, A. & Schreiber, F. (2022*b*). *Synchrotron Radiat. News* **35**(4), 21–27.
Steele, J. A., Solano, E., Hardy, D., Dayton, D., Ladd, D., White, K., Chen, P., Hou, J., Huang, H., Saha, R. A., Wang, L., Gao, F., Hofkens, J., Roeffaers, M. B. J., Chernyshov, D. & Toney, M. F. (2023). *Adv. Energy Mater.* **13**, 2300760.
Steitz, R., Müller-Buschbaum, P., Schemmel, S., Cubitt, R. & Findenegg, G. (2004). *Europhys. Lett.* **67**, 962–968.
Ulbrandt, J. G., Zhang, X., Headrick, R. L., Liu, R., Dawber, M. & Evans-Lutterodt, K. (2020). *Phys. Rev. B* **101**, 241406.
Völter, C., Starostin, V., Lapkin, D., Munteanu, V., Romodin, M., Hylinski, M., Gerlach, A., Hinderhofer, A. & Schreiber, F. (2025). *J. Appl. Cryst.* **58**, 513–522.
Weng, J., Xu, W., Wiaderek, K. M., Borkiewicz, O. J., Chen, J., Von Dreele, R. B., Gallington, L. C. & Ruett, U. (2023). *J. Synchrotron Rad.* **30**, 855.
Werzer, O., Kowarik, S., Gasser, F., Jiang, Z., Strzalka, J., Nicklin, C. & Resel, R. (2024). *Nat. Rev. Methods Primers* **4**, 15.
Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Santos, L. B. da S., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J. & Mons, B. (2016). *Sci. Data* **3**, 160018.
Willmott, P. (2019). *An introduction to synchrotron radiation: techniques and applications.* Chichester: John Wiley & Sons.
Yang, D., Löhrer, F. C., Körstgens, V., Schreiber, A., Cao, B., Bernstorff, S. & Müller-Buschbaum, P. (2020). *Adv. Sci.* **7**, 2001117.
Zhang, X., Ulbrandt, J. G., Myint, P., Fluerasu, A., Wiegart, L., Zhang, Y., Nelson, C., Ludwig, K. F. & Headrick, R. L. (2024). *Phys. Rev. Mater.* **8**, 033403.
Ziletti, A., Kumar, D., Scheffler, M. & Ghiringhelli, L. M. (2018). *Nat. Commun.* **9**, 2775.