Documentation: SAEDProcessor Class & ePDF Workflow

ePDF Suite

Overview

The SAEDProcessor class and ePDF Workflow provide a complete pipeline for extracting Pair Distribution Functions (PDF) from Selected Area Electron Diffraction (SAED) data. The workflow combines geometric calibration, image recalibration, azimuthal integration, and PDF computation with interactive parameter optimization.

Key Components:

- `SAEDProcessor`: Main data processing orchestrator
- `PDFInteractive`: Interactive widget-based parameter tuning
- `ePDF_Workflow.ipynb`: Complete end-to-end usage example

Part I
Module: SAEDProcessor.py

1 SAEDProcessor Class

1.1 Overview

`SAEDProcessor` is the primary interface for SAED data processing. It handles:

1. Loading diffraction images and metadata
2. Integrating 2D diffraction data to 1D scattering profiles
3. Computing electron PDFs with optional background subtraction
4. Interactive or batch PDF parameter optimization

1.2 __init__(dm4_file, poni_file, beamstop=False)

Purpose: Initialize processor with diffraction image and calibration parameters.

Parameters:

- `dm4_file`: Path to SAED image (DM4, DM3, TIFF format)
- `poni_file`: Path to geometric calibration file (PONI format)
- `beamstop`: Boolean indicating beamstop presence (default: False)

Internal State:

- `self.metadata`: Dictionary containing wavelength, camera title, etc.
- `self.img`: 2D diffraction image array
- `self.ai`: pyFAI AzimuthalIntegrator object for integration

Example:

```
from SAEDProcessor import SAEDProcessor

processor = SAEDProcessor(
    dm4_file='sample.dm4',
    poni_file='calibration.poni',
    beamstop=True
)
```

1.3 integrate(npt=2500, beamstop=False)

Purpose: Perform azimuthal integration with optional recalibration.

Algorithm:

Step 1: Beam Center Recalibration

If `beamstop=True`, automatically refine beam center position:

$$\text{recalibrate\_with\_beamstop}(dm4\_file, poni\_file, \ldots)$$

Otherwise, use maximum intensity method:

$$\text{recalibrate\_no\_beamstop}(dm4\_file, poni\_file)$$

Step 2: Azimuthal Integration
Average diffraction intensity over azimuthal angle:

$$I(q) = \frac{1}{2\pi} \int_0^{2\pi} I(r, \theta)\, r\, dr\, d\theta$$

where:

- $q = \frac{4\pi \sin\theta}{\lambda}$ is scattering vector
- λ is electron wavelength (from PONI file)
- `npt`: number of radial bins (default: 2500)
- Polarization factor: 0.99 (electron scattering)

Parameters:

- `npt`: Number of integration points (default: 2500)
- `beamstop`: Whether to recalibrate before integration

Returns:

- `q`: Scattering vector array in $^\circ$ Å^{-1}
- `i`: Integrated intensity array

Example:

```
q, intensity = processor.integrate(npt=2500, beamstop=True)
```

1.4 extract_epdf()

```
extract_epdf(ref_diffraction_image=None,
             composition='Au',
             rmin=0.1, rmax=50.0, rstep=0.01,
             outputfile=None,
             interactive=True,
             plot=False)
```

Purpose: Complete PDF extraction workflow with optional interactive optimization.
Workflow Steps:
Step 1: Load Data
Load sample and reference (background) diffraction images:

- Extract diffraction data and metadata
- Store for later access in PDFInteractive

Step 2: Recalibrate Beam Center
Refine geometric calibration based on diffraction rings:

$$\text{ai} = \begin{cases} \text{recalibrate\_no\_beamstop}(...) & \text{if } \neg\text{beamstop} \\ \text{recalibrate\_with\_beamstop}(...) & \text{if beamstop} \end{cases}$$

Step 3: Azimuthal Integration
Convert 2D diffraction to 1D profile:

$$I(q) = \text{ai.integrate1d}(\text{image}, \text{npt} = 2500, \text{unit} = \text{“q\_A}^{-1}\text{”})$$

Separately integrate sample and reference if provided.
Step 4: PDF Computation
Compute pair distribution function:

$$G(r) = \frac{4\pi r \rho_0}{\text{Lorch}(q_{\max})} \int_0^{q_{\max}} q[I(q) - f^2(q)] \sin(qr)\, dq$$

where:

- ρ_0 is average atomic density
- $f(q)$ is electron scattering factor
- `Lorch`: damping function to eliminate termination ripples

Step 5: Output Generation
Write results to .gr file (PDFGETX3 compatible format) with metadata header.
Parameters:

- `ref_diffraction_image`: Background image path (optional)
- `composition`: Chemical formula for scattering factors (default: 'Au')
- `rmin`, `rmax`, `rstep`: Real-space range (default: 0.1-50 °Å, step 0.01)
- `outputfile`: Output filename (auto-generated if None)
- `interactive`: Enable interactive tuning (default: True)
- `plot`: Display result plots (default: False)

Output File Format (.gr):
PDFGETX3-compatible format with metadata header:

```
[DEFAULT]

version = ePDFsuite 1.0

# input and output specifications
 dataformat = q_A
 inputfile = sample.dm4
 backgroundfile = background.dm4
 outputtype = gr

 #PDF calculation setup
 mode = electrons
 wavelength = 0.0251
 composition = Au
```

```
 bgscale = 1.00
 qmin = 1.50
 qmax = 24.00
 rmin = 0.00
 rmax = 50.00
 rstep = 0.01

#### start data

#S 1
#L r(  )  G(  ^-2)
 0.010000   1.234567
 0.020000   2.345678
 ...
```

Non-Interactive Mode:
Compute PDF with fixed parameters:

```
processor.extract_epdf(
    ref_diffraction_image='background.dm4',
    composition='Au',
    interactive=False,
    plot=True,
    outputfile='result.gr'
)
```

Interactive Mode:
Launch PDFInteractive GUI for real-time parameter tuning (see below).

2 PDFInteractive Class

2.1 Overview

`PDFInteractive` provides an ipywidgets-based interface for interactive PDF parameter optimization with real-time feedback.

2.2 __init__(q, Iexp, composition, Iref=None, rmin=0, rmax=50, rstep=0.01, xray=False, outputfile='./pdf_results.gr', SAEDProcessor=None)

Purpose: Initialize interactive interface with data and parameters.

Metadata Extraction:
If SAEDProcessor provided, retrieve:

- Wavelength from metadata
- Camera description
- File paths for output header

Widget Creation:
Create adjustable sliders for PDF parameters:

Parameter	Default	Min	Max
`bgscale`	1.0	0.0	2.0
`qmin`	1.5	$q_{\min}$	24.0
`qmax`	24.0	$q_{\min}$	$q_{\max}$
`qmaxinst`	24.0	$q_{\min}$	$q_{\max}$
`rpoly`	1.4	0.1	2.5

Parameters:

- `q`: Scattering vector from integrate1d
- `Iexp`: Experimental intensity
- `composition`: Chemical formula
- `Iref`: Reference/background intensity (optional)
- `xray`: Use X-ray scattering factors if True (default: False for electrons)
- `SAEDProcessor`: Parent processor instance for metadata

2.3 update_plot(bgscale, qmin, qmax, qmaxinst, rpoly, lorch)

Purpose: Recompute PDF and update visualization on slider change.

Algorithm:

1. Clear previous plot
2. Call `compute_ePDF` with current slider values
3. Store results (`r`, `G`) for saving
4. Display updated plot

Parameters:

- `bgscale`: Background scaling factor
- `qmin`, `qmax`: Q-range selection
- `qmaxinst`: Instrumental qmax cutoff
- `rpoly`: Polynomial background order
- `lorch`: Enable Lorch damping (checkbox)

2.3.1 Physical Meaning of Parameters

bgscale: Scales background intensity before subtraction:

$$I_{\text{corrected}} = I_{\text{sample}} - \text{bgscale} \times I_{\text{background}}$$

qmin, qmax: Limits integration range to minimize noise and artifacts:

$$G(r) = \frac{4\pi r \rho_0}{\text{Lorch}(q_{\max})} \int_{q_{\min}}^{q_{\max}} q[I(q) - f^2(q)] \sin(qr)\, dq$$

qmaxinst: Experimental resolution cutoff (prevents noise amplification).
rpoly: Polynomial order for background fitting (1.4 typical for electron PDF).
Lorch: Applies Lorch damping function to eliminate Fourier termination ripples:

$$\text{Lorch}(q) = \frac{\sin(\pi q/q_{\max})}{\pi q/q_{\max}}$$

2.4 save_results(b, outputfile='./pdf_results.gr')

Purpose: Export last computed PDF with metadata header.

Algorithm:

1. Check if results exist (`last_r`, `last_G`)
2. Build metadata header from slider values and SAEDProcessor
3. Write column-stacked (r, G) data with `np.savetxt`
4. No # prefix on header lines (comments='')

Output Header:

```
[DEFAULT]

version = ePDFsuite 1.0

# input and output specifications
 camera = Gatan K2 IS
 inputfile = sample.dm4
 backgroundfile = background.dm4
 outputtype = gr

 #PDF calculation setup
 mode = electrons
 wavelength = 0.0251
 composition = Au
```

```
bgscale = 1.00
qmin = 1.50
qmax = 24.00
rpoly = 1.40
```

Parameters:

- `b`: Button widget (callback parameter, unused)
- `outputfile`: Output .gr filename

2.5 show()

Purpose: Display interactive interface with horizontal layout.

Layout:

- `Left`: Parameter sliders + Save button
- `Right`: Live plot area

Initialization:
Automatically generates initial plot with default parameters.

Example:

```
pdf_interactive.show()  # Display in Jupyter notebook
```

Part II
Module: ePDF_Workflow.ipynb

3 Jupyter Notebook Workflow

3.1 Overview

The `ePDF_Workflow.ipynb` notebook demonstrates complete end-to-end ePDF extraction pipeline with practical examples.

3.2 Cell Structure

3.2.1 Cell 1: Matplotlib Setup

```
%matplotlib widget
```

Enable interactive matplotlib in Jupyter.

3.2.2 Cell 2: Calibration Documentation

Markdown cell describing pyFAI geometric calibration:

- Uses pyFAI library for detector geometry
- Generates .poni file with calibration parameters
- Beam center automatically recalibrated before integration
- Distance and rotation angles assumed constant

3.2.3 Cell 3: Calibration Execution

```
from calibration import perform_geometric_calibration

datapath = '/path/to/data/'
sample_diff_calib = datapath + "calibrant.dm4"
cif_file = "/path/to/Au.cif"

perform_geometric_calibration(
    diffraction_image=sample_diff_calib,
    cif_file=cif_file
)
```

Workflow:

1. Load Au reference structure (CIF)
2. Generate expected diffraction peaks
3. Load calibration diffraction image
4. Convert DM4 → EDF
5. Print experimental settings
6. Launch pyFAI-calib2 GUI

7. User manually fits calibration
8. Output: `calibration.poni`

Output: PONI file with parameters:

distance, poni1, poni2, rot1, rot2, rot3, wavelength, pixel_size

3.2.4 Cell 4: PDF Extraction Documentation

Markdown describing SAEDProcessor usage.

3.2.5 Cell 5: PDF Extraction Execution

```
from SAEDProcessor import SAEDProcessor

datapath = '/path/to/data/'
sample_diff = datapath + "sample.dm4"
ref_diff = datapath + "background.dm4"
poni_file = './calibration.poni'

proc = SAEDProcessor(sample_diff, poni_file)
proc.extract_epdf(
    ref_diffraction_image=ref_diff,
    composition='Au',
    rmin=0.1, rmax=50.0, rstep=0.01,
    interactive=False,
    plot=True,
    outputfile='Au_pdf.gr'
)
```

Workflow:

1. Initialize processor with sample and calibration
2. Specify background image (optional)
3. Set real-space parameters (rmin-rmax)
4. Run non-interactive mode with plotting
5. Output: `Au_pdf.gr`

4 Complete Pipeline

4.1 Data Flow

1. **Raw Data**: SAED images (DM4/DM3/TIFF)
2. ↓
3. **Calibration**: Reference crystal + pyFAI-calib2 → PONI file
4. ↓
5. **Loading**: Read with `filereader.load_data`

6. ↓
7. **Recalibration**: Refine beam center `recalibrate_*`
8. ↓
9. **Integration**: Azimuthal integration $q(r, \theta) \rightarrow I(q)$
10. ↓
11. **PDF Computation**: `compute_ePDF`
12. ↓
13. **Interactive Tuning**: `PDFInteractive` (optional)
14. ↓
15. **Output**: PDF file (.gr format)

4.2 Key Parameters Summary

Stage	Parameter	Typical Value	Unit
Calibration	wavelength	0.0251	° Å
	distance	0.20-0.30	m
Integration	npt	2500	points
	polarization_factor	0.99	–
PDF	rmin	0.1	° Å
	rmax	50.0	° Å
	rstep	0.01	° Å
	qmin	1.5	° $Å^{-1}$
	qmax	20-24	° $Å^{-1}$
	bgscale	0.8-1.2	–
	rpoly	1.4	–

5 Usage Examples

5.1 Batch Processing

```
import glob

sample_files = glob.glob('/data/*.dm4')
for sample_file in sample_files:
    proc = SAEDProcessor(sample_file, 'calib.poni')
    basename = sample_file.split('/')[-1].split('.')[0]

    proc.extract_epdf(
        composition='Au',
        outputfile=f'{basename}_pdf.gr',
        interactive=False
    )
    print(f'Processed {basename}')
```

5.2 Interactive Optimization

```
# Launch interactive GUI for parameter tuning
proc.extract_epdf(
    ref_diffraction_image='background.dm4',
    composition='Au',
    interactive=True  # Opens PDFInteractive
)
# User adjusts sliders and clicks Save
```

5.3 Multi-Sample Comparison

```
from SAEDProcessor import SAEDProcessor
import matplotlib.pyplot as plt

samples = ['sample_1.dm4', 'sample_2.dm4']
colors = ['blue', 'red']

fig, ax = plt.subplots()
for sample, color in zip(samples, colors):
    proc = SAEDProcessor(sample, 'calib.poni')
    q, i = proc.integrate(npt=2500)
    ax.loglog(q, i, label=sample, color=color)

ax.set_xlabel('q ( $ ^{-1}$)')
ax.set_ylabel('Intensity (arb. units)')
ax.legend()
plt.show()
```

6 Dependencies

- **PyFAI**: Azimuthal integration, PONI file handling
- **HyperSpy**: DM4 file I/O, metadata extraction
- **NumPy**: Array operations
- **Matplotlib**: Visualization (interactive backend)
- **ipywidgets**: Interactive sliders and buttons
- **scikit-image**: Image processing (calibration module)
- **PyMatGen**: Crystallographic calculations (calibration module)

7 Troubleshooting

7.1 Issue: Beam center misalignment

Solution: Set `beamstop=True` in `SAEDProcessor` or `integrate()` to enable automatic recalibration.

7.2 Issue: High-frequency noise in PDF

Solution:

- Increase `bgscale` to improve background subtraction
- Reduce `qmax` to exclude high-q noise
- Enable Lorch damping (`lorch=True`)

7.3 Issue: Termination ripples in PDF

Solution: Enable Lorch windowing in `PDFInteractive` checkbox. This applies the Lorch damping function to the $I(q)$ data before Fourier transform.