Documentation: PDF Extraction Module

Generated Documentation

February 4, 2026

Contents

1 Overview

`pdfextraction.py` is a comprehensive module for computing Pair Distribution Functions (PDFs) from electron diffraction data using PDF extraction algorithms similar to PDFgetX3. It provides:

- Chemical composition parsing and scattering factor computation
- Electron and X-ray scattering factor calculations (Lobato parametrization)
- Polynomial background subtraction
- PDF extraction with Fourier transform methods
- Interactive GUI for parameter optimization using Jupyter widgets

This module is designed for electron diffraction analysis but also supports X-ray scattering through parametrizable options.

2 Chemistry and Scattering Utilities

2.1 parse_formula()

```
parse_formula(formula)
```

Purpose: Parse chemical formula string and extract element symbols and stoichiometric ratios.

Algorithm:

1. Use regex to extract element tokens: pattern matches capital letter + optional lowercase + optional number
2. Parse element symbols (e.g., `Au`, `Si`)
3. Extract stoichiometric counts (default 1 if omitted)
4. Normalize counts to atomic fractions

Examples:

- `'Au'` → (`['Au']`, [1.0])
- `'SiO2'` → (`['Si', 'O']`, [0.333, 0.667])
- `'Al2O3'` → (`['Al', 'O']`, [0.4, 0.6])

Parameters:

- `formula`: Chemical formula string (e.g., `'Au'`, `'SiO2'`)

Returns:

- `elements`: List of element symbols
- `ratios`: List of atomic fractions (normalized to sum = 1.0)

2.2 compute_avg_scattering_factor()

```
compute_avg_scattering_factor(formula, x_max, x_step,
                              qvalues=True, xray=False)
```

Purpose: Compute composition-weighted average scattering factor $\langle f \rangle$ using Lobato parametrization.

Physical Background:

Scattering factor $f(s)$ describes the amplitude of scattering from an atom. For a composite material:

$$\langle f(s) \rangle = \sum_i w_i f_i(s)$$

where w_i are atomic fractions and s is the scattering parameter (related to q by $s = q/(2\pi)$).

Algorithm:

1. Parse chemical formula via `parse_formula()`
2. Convert between q and s spaces if needed:
   - If `qvalues=True`: $s_{\max} = q_{\max}/(2\pi)$, $s_{\text{step}} = q_{\text{step}}/(2\pi)$
   - If `qvalues=False`: use s directly
3. Load Lobato parametrization (ABTEM library)
4. Compute line profiles for each element up to cutoff frequency
5. Compute q and s arrays
6. Weight by atomic fractions and sum

Parameters:

- `formula`: Chemical formula (e.g., `'Au'`, `'SiO2'`)
- `x_max`: Maximum scattering vector or frequency
- `x_step`: Sampling step in scattering vector or frequency
- `qvalues`: If `True`, use q space (Å^{-1}); if `False`, use s space
- `xray`: If `True`, use X-ray scattering; if `False` (default), use electron scattering

Returns:

- `q`: Scattering vector array (Å^{-1})
- `favg`: Average scattering factor array

2.3 compute_f2avg()

```
compute_f2avg(formula, x_max, x_step, qvalues=True, xray=False)
```

Purpose: Compute composition-weighted average squared scattering factor $\langle f^2 \rangle$.

Algorithm: Identical to `compute_avg_scattering_factor()`, except square each element's scattering factor before weighting:

$$\langle f^2(s) \rangle = \sum_i w_i f_i^2(s)$$

This is used for intensity normalization in PDF extraction.

Parameters:

- `formula`: Chemical formula
- `x_max`, `x_step`: Scattering range parameters
- `qvalues`, `xray`: Same as `compute_avg_scattering_factor()`

Returns:

- `q`: Scattering vector array
- `f2avg`: Average squared scattering factor array

3 PDF Extraction: Core Algorithm

3.1 fit_polynomial_background()

The purpose of this function is to refine the baseline of the F(q) function with a polynom, thus mimicking the approach implemented in pdfgetX3.

```
fit_polynomial_background(q, Fm, rpoly=0.9, qmin=0.3, qmax=None)
```

Purpose: Fit polynomial background to modified intensity curve (PDFgetX3 style).

Physical Motivation:

At high q, scattering approaches the incoherent limit where intensity plateaus. A polynomial model captures this background, allowing subtraction before Fourier transform.

Algorithm:

1. Select q range for fitting: $q_{\min}$ to $q_{\max}$
2. Compute polynomial degree: $\deg = \lfloor r_{\text{poly}} \cdot q_{\max}/\pi \rfloor$
3. Clamp degree to valid range: $1 \leq \deg < N_{\text{mask}}$
4. Normalize intensity: $y = F_m(q)/q$
5. Fit polynomial of degree `deg` to normalized data
6. Evaluate polynomial across full q range
7. Return background curve

The parameter r_{poly} controls polynomial complexity: higher values allow more oscillation.

Parameters:

- `q`: Scattering vector array
- `Fm`: Modified structure factor array
- `rpoly`: Polynomial complexity factor (default: 0.9)
- `qmin`, `qmax`: Fitting range (Ångström^{-1})

Returns:

- Array: Polynomial background curve same shape as input

INSERT FIGURES!!

3.2 compute_PDF_electrons()

```
compute_PDF_electrons(q, Iexp, composition, Iref=None, bgscale
   =1.0,
                      qmin=0.3, qmax=None, qmaxinst=None, rmin
                         =0.0,
                      rmax=50.0, rstep=0.01, rpoly=1.4, Lorch=
                         True,
                      plot=False)
```

Purpose: Complete PDF extraction pipeline for electron diffraction data.
This is the central function implementing the PDFgetX3-style algorithm for electrons.

3.2.1 Step 1: Background Subtraction

If reference background I_{ref} is provided:

$$I_{\text{corr}} = I_{\text{exp}} - \text{bgscale} \cdot I_{\text{ref}}$$

where `bgscale` is a scaling factor for reference intensity.

3.2.2 Step 2: Electron Scattering Normalization

Compute composition-weighted squared scattering factor $\langle f^2 \rangle$:

$$I_{\text{norm}} = \frac{I_{\text{corr}}}{\langle f^2(q) \rangle}$$

Estimate asymptotic intensity from high-q region ($0.9 \cdot q_{\max}$):

$$I_\infty = \langle I_{\text{norm}}(q > 0.9 q_{\max}) \rangle$$

3.2.3 Step 3: Modified Structure Factor

Compute modified structure factor:

$$F(q) = q \left(\frac{I_{\text{norm}}}{I_\infty} - 1 \right)$$

This removes the tail-off of the intensity curve.

3.2.4 Step 4: Polynomial Background Removal

Fit polynomial background via `fit_polynomial_background()`:

$$F_c(q) = F(q) - B(q)$$

Corrected structure factor without q-damping.

3.2.5 Step 5: Windowing (Optional Lorch Correction)

Apply Lorch window to eliminate termination ripples in PDF:

$$F_w(q) = F_c(q) \cdot \text{sinc}\left(\frac{q}{q_{\max}} \right)$$

where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$.
Lorch window suppresses high-q oscillations that cause artifacts at small r.

3.2.6 Step 6: Fourier Transform

Compute PDF via sine Fourier transform:

$$G(r) = \frac{2}{\pi} \int_0^{q_{\max}} F_w(q) \sin(qr)\, dq$$

Evaluate on regular grid: $r \in [r_{\min}, r_{\max}]$ with step r_{step}.
Numerical integration via trapezoidal rule on windowed q range.
Algorithm Summary:

1. Background correction if reference provided
2. Compute $\langle f^2 \rangle$ for composition normalization
3. Normalize intensity by scattering factors
4. Compute modified structure factor $F(q)$
5. Fit and subtract polynomial background
6. Apply Lorch window (optional)
7. Fourier transform to real space

Parameters:

- `q`: Scattering vector (Å^{-1})
- `Iexp`: Experimental intensity
- `composition`: Chemical formula string
- `Iref`: Background reference intensity (optional)
- `bgscale`: Background scaling factor (default: 1.0)
- `qmin`, `qmax`: Processing q range (Å^{-1})
- `qmaxinst`: Background fitting upper limit (default: `qmax`)
- `rmin`, `rmax`: PDF evaluation range (Ångströms)
- `rstep`: PDF sampling step (Ångströms)
- `rpoly`: Polynomial complexity factor
- `Lorch`: Apply Lorch window (default: `True`)
- `plot`: Display diagnostic plots (default: `False`)

Returns:

- `r`: Real-space distance array (Ångströms)
- `G`: Pair distribution function array

Diagnostic Plots (if `plot=True`):

- **Panel 1**: Raw and background-subtracted intensities
- **Panel 2**: Corrected structure factor $F_c(q)$ with polynomial fit
- **Panel 3**: Final PDF $G(r)$

4 Interactive GUI: PDFInteractive Class

4.1 Class Overview

`PDFInteractive` provides a Jupyter widget-based interface for real-time parameter optimization during PDF extraction.

Features:

- Slider controls for all key PDF parameters
- Live plot updates as parameters change
- Save results with metadata to `.gr` files
- Composition-weighted scattering factor computation

4.2 __init__()

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

Purpose: Initialize the interactive interface with data and controls.

Algorithm:

1. Store PDF configuration parameters and data
2. Create slider widgets for parameter adjustment
3. Create checkbox for Lorch window control
4. Create save button with callback
5. Organize widgets in vertical layout
6. Initialize storage for last computed results

Slider Configuration:

Parameter	Min	Default	Max	Step
`bgscale`	0.0	1.0	2.0	0.01
`qmin`	$q_{\min}$	1.5	24.0	0.01
`qmax`	$q_{\min}$	$\min(24, q_{\max})$	$q_{\max}$	0.01
`qmaxinst`	$q_{\min}$	$\min(24, q_{\max})$	$q_{\max}$	0.01
`rpoly`	0.1	1.4	2.5	0.01

Table 1: Widget slider ranges and defaults

Parameters:

- `q`: Scattering vector array (Å^{-1})
- `Iexp`: Experimental intensity
- `composition`: Chemical formula

- `Iref`: Background reference (optional)
- `rmin`, `rmax`, `rstep`: Real-space grid parameters
- `xray`: Use X-ray scattering factors (default: `False`)
- `outputfile`: Default save filename

4.3 update_plot()

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

Purpose: Recalculate PDF and update plots when sliders change (callback function).
Algorithm:

1. Clear previous plot output
2. Call `compute_PDF_electrons()` with new parameters
3. Store results in `self.last_r`, `self.last_G`
4. Display diagnostic plots

This function is automatically triggered by widget value changes via `interactive_output`.
Parameters:

- All parameters mirror `compute_PDF_electrons()` sliders

4.4 save_results()

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

Purpose: Save computed PDF data and metadata to `.gr` file.
File Format:
Text format with metadata header and data columns:

```
#Composition.............................Au
#bgscale.................................1.0000
#qmin (       ).........................1.5000
...metadata lines...

#DATA: r (  ) vs G(r)

0.010000       -0.12345678
0.020000       -0.11234567
...data lines...
```

Metadata Captured:

- Composition
- All PDF parameters: `bgscale`, `qmin`, `qmax`, `qmaxinst`, `rpoly`
- Real-space grid: `rmin`, `rmax`, `rstep`

- Lorch correction status

Parameters:

- `b`: Button widget (unused, required by callback protocol)
- `outputfile`: Output filename (default: `./pdf_results.gr`)

Error Handling:

- Returns with warning if no results have been computed yet
- Automatically converts `.csv` extension to `.gr`

4.5 show()

```
show()
```

Purpose: Display the interactive interface in Jupyter notebook.
Layout:

- Left panel: Parameter sliders and save button
- Right panel: Output area for plots

Algorithm:

1. Create horizontal layout with sliders and plot output
2. Display using IPython `display()` function
3. Generate initial plot with default parameters

5 PDF Processing Workflow

5.1 Typical Usage Example

```
from pdfextraction import compute_PDF_electrons, PDFInteractive

# Load experimental data (q in         , I in counts)
q_data = ...   # Shape (N,)
I_exp = ...    # Shape (N,)
I_bg = ...     # Background reference (optional)

# Compute PDF with fixed parameters
r, G = compute_PDF_electrons(
    q=q_data,
    Iexp=I_exp,
    composition='Au',
    Iref=I_bg,
    bgscale=1.2,
    qmin=1.5,
    qmax=24.0,
    rpoly=1.4,
    Lorch=True,
    plot=True
)

# Interactive optimization in Jupyter
pdf_gui = PDFInteractive(
    q=q_data, Iexp=I_exp, composition='Au',
    Iref=I_bg, outputfile='./sample_pdf.gr'
)
pdf_gui.show()
```

5.2 Parameter Selection Guide

5.2.1 Background Scaling (bgscale)

Controls background subtraction strength:

- bgscale = 1.0: Use reference intensity as-is
- bgscale ¿ 1.0: Strengthen background removal (reduce high-q tail)
- bgscale ¡ 1.0: Weaken background removal

Optimization: Adjust to minimize artificial oscillations in PDF at large r.

5.2.2 Q-Range (qmin, qmax)

- qmin: Lower limit cuts low-q noise; typical range: 0.5–2.0 Å^{-1}
- qmax: Upper limit where data becomes noise-dominated; typical: 20–30 Å^{-1}

Real-space resolution: $\Delta r \approx \pi/q_{\max}$

5.2.3 Polynomial Order (`rpoly`)

Controls background subtraction complexity:

- Low `rpoly` (0.5–1.0): Smooth background (risk of over-subtraction)
- Mid `rpoly` (1.0–1.5): Balanced fit
- High `rpoly` (1.5–2.5): Complex oscillations (risk of under-subtraction)

5.2.4 Lorch Window

Applies sine cardinal damping to eliminate Fourier termination ripples:

- `Lorch=True`: Recommended for most applications
- `Lorch=False`: Preserve maximum high-r resolution (inspect for ripples)

6 Physical Constants and Conventions

6.1 Scattering Vector

$$q = \frac{4\pi}{\lambda} \sin\theta$$

where λ is wavelength and 2θ is scattering angle.
Equivalently: $q = 2\pi s$ where $s = 1/d$ is spatial frequency.

6.2 Structure Factor and PDF

Total structure function:

$$F(q) = |S(q) - 1| \cdot q$$

where $S(q)$ is the static structure factor.
PDF relates to pair correlation function $g(r)$:

$$G(r) = 4\pi\rho_0 r[g(r) - 1]$$

where ρ_0 is atomic number density.

6.3 Fourier Transform Relationship

$$G(r) = \frac{2}{\pi} \int_0^\infty qF(q) \sin(qr)\, dq$$

Finite upper limit $q_{\max}$ produces termination ripples unless windowed.

7 Computational Considerations

7.1 Performance

- Scattering factor computation: $O(N \times M)$ where N = elements, $M = q$ points
- Fourier transform: $O(M^2)$ for naive integration; $O(M \log M)$ for FFT (not used here)
- Interactive updates: Real-time with typical sizes ($M \sim 1000$)

7.2 Numerical Stability

- Polynomial fitting: Uses orthogonal Chebyshev basis internally (NumPy)
- Trapezoid integration: Adequate for smooth integrands
- Lorch window: Prevents numerical overflow at small r

7.3 Memory Usage

For typical data ($N_q = 1000$, $N_r = 5000$):

- Single PDF computation: ~ 50 MB
- Interactive GUI with caching: ~ 100 MB

8 Dependencies

- `matplotlib`: Plotting and visualization
- `abtem`: Lobato scattering parametrization
- `numpy`: Numerical operations and array manipulation
- `numpy.polynomial`: Polynomial fitting
- `scipy`: (Optional, not directly used but recommended)
- `ipywidgets`: Interactive Jupyter widgets
- `IPython`: Jupyter display functions