SAXS Polydispersity Simulation & Analysis App: System Descriptor

Last Updated: Wednesday, February 11, 2026 **Version:** 2.1 (Bug Fix - Function Call) **Author:** Gemini (Simulating for User)

1. Overview

This application is a scientific tool designed to simulate Small Angle X-ray Scattering (SAXS) profiles for polydisperse systems and evaluate analytical methods for recovering size distribution parameters. It specifically compares numerical reconstruction methods (NNLS) against integral-parameter methods derived by Tomchuk et al.

The app operates in two modes:

1. **Single Mode:** Interactive, real-time simulation and analysis of a single dataset.
2. **Batch Mode:** Automated parameter sweeping to generate statistical error maps for the analysis methods.

2. Physics & Theoretical Background

2.1. The Forward Problem (Simulation)

The core simulation generates a 1D scattering intensity profile $I(q)$ based on the integral:

$$I(q) = \text{scale} \cdot \int_0^\infty N(R) \cdot V^2(R) \cdot P(q, R) \cdot dR + \text{background}$$

Where:

- q: Scattering vector magnitude (nm^{-1}).
- R: Characteristic dimension (Radius for spheres, Radius of Gyration for polymers).
- $N(R)$: Number density distribution function.
- $V(R)$: Volume of the particle.
- $P(q, R)$: Normalized form factor ($P(0, R) = 1$).

Form Factors ($P(q, R)$)

The app supports two particle topologies:

1. **Hard Spheres:**

$$P(q, R) = \left(\frac{3[\sin(qR) - qR\cos(qR)]}{(qR)^3} \right)^2$$

2. **Gaussian Chains (Debye/IDP):**

$$P(q, R_g) = \frac{2}{(qR_g)^4} \left[\exp(-(qR_g)^2) + (qR_g)^2 - 1 \right]$$

Size Distributions ($N(R)$)

Polydispersity is defined by the relative standard deviation, p:

$$p = \frac{\sigma}{\mu}$$

where μ is the mean radius and σ is the standard deviation.

2.2. The Inverse Problem (Analysis)

The app attempts to recover the input parameters (μ_{rec}, p_{rec}) from the simulated $I(q)$ using two distinct approaches.

Method A: The Tomchuk Approach (Integral Invariants)

Based on *Tomchuk et al., J. Appl. Cryst. (2023)*, this method relies on calculating the **integral invariants** of the scattering curve to determine the polydispersity indices (PDI and PDI2) defined in Equations 4 and 21 of the paper.

Step 1: Extraction of Experimental Invariants The code first extracts the four fundamental invariants from the dataset $I_{meas}(q)$:

1. **Guinier Parameters ($G, R_{g,app}$):** Derived from the low-q region ($qR_g < 1.3$) where $I(q) \approx G \exp(-q^2 R_g^2/3)$.
   - G: Zero-angle intensity ($I(0)$).
   - $R_{g,app}$: Apparent Radius of Gyration.
2. **Porod Constant (K_p):** Derived from the high-q tail where the intensity decays as a power law: $I(q) \sim K_p q^{-4}$.
3. **Porod Invariant (Q^*):** The total scattering power, defined as $Q^* = \int_0^\infty q^2 I(q) dq$.
   - **Tail Correction:** Since experimental data is limited to a finite q_{max}, the code applies a correction for the unmeasured tail ($q > q_{max}$) assuming Porod behavior:

$$Q^* = \int_{q_{min}}^{q_{max}} q^2 I(q) dq + \frac{K_p}{q_{max}}$$

Step 2: Calculation of PDI and PDI2 (Eq. 4 & 21) Using the invariants extracted above, the code calculates the Polydispersity Indices directly:

- **PDI (Equation 4):** This index relates the volume-average moments. It is calculated as a function of G, Q^*, and R_g.

$$PDI_{exp} = f(G, Q^*, R_g) \quad (\text{Equation 4 from paper})$$

- **PDI2 (Equation 21):** This index relates the surface-average moments. It is calculated as a function of Q^*, K_p, and R_g.

$$PDI2_{exp} = f(Q^*, K_p, R_g) \quad (\text{Equation 21 from paper})$$

Step 3: Recovery of Distribution Parameters (p, μ) The code recovers the distribution width (p) and mean (μ) by matching the experimental PDI values to their theoretical counterparts.

1. **Solving for p:** The code assumes a distribution type (e.g., Lognormal) and uses a root-finding algorithm to find the p value where the **Theoretical PDI** (calculated from the distribution's moments) equals the **Experimental PDI** (from Eq. 4 or 21).
2. **Recovering μ (Mean Radius):** Once p is known, the code calculates the conversion factor to get the mean radius (μ) from the measured R_g.

Method B: NNLS (Non-Negative Least Squares)

A model-independent approach that discretizes the integral equation:

$$I_{meas} \approx \mathbf{A} \cdot \mathbf{x}$$

- $\mathbf{A}$: Matrix where $A_{ij} = P(q_i, R_j)$.
- $\mathbf{x}$: The weight fraction of particles at size R_j.
- **Algorithm:** Solves $\min ||\mathbf{Ax} - \mathbf{I}||^2$ subject to $\mathbf{x} \geq 0$.

3. Code Architecture

3.1. File Structure

- `streamlit_app.py` : The entry point. Handles session state initialization and routing.
- `sim_utils.py` : The "Physics Engine".
  - `sphere_form_factor` , `debye_form_factor` .
  - `get_distribution` : Generates PDFs.
  - `run_simulation_core` : Integration and noise application.
  - `calc_theoretical_guiner_rg` : Helper to verify R_g vs Radius.
- `analysis_utils.py` : The "Solver Brain" (Unmodified from Original).
  - `analyze_tomchuk` : Extracts Invariants (G, R_g, K_p, Q^*). Calculates PDI (Eq 4) and PDI2 (Eq 21). Solves for p.
  - `analyze_nnls` : NNLS solver.
- `single_mode.py` : UI logic for the interactive dashboard.
  - **Update:** Now displays "Theoretical Rg (z-avg)" to clarify weighting differences.
- `batch_mode.py` : Logic for parameter sweeps and error mapping.

3.2. Key Data Structures

- Analysis Result (`analysis_res` dictionary):
  - `Rg_scat` , `I0_scat` : Guinier parameters.
  - `Porod_K` , `Invariant_Q` : Porod parameters.
  - `pdi_val` : The calculated PDI from Eq 4.
  - `pdi2_val` : The calculated PDI2 from Eq 21.
  - `p_rec` : Recovered p using PDI (Eq 4).
  - `mean_rg_rec` : Recovered mean size.