1. **Aerosol Mode Fitting Algorithm: Test Framework**

The test framework for validating an aerosol mode fitting algorithm that decomposes particle size distributions into log-normal modes. We have evaluated some synthetic test case data cooked up from individual Gaussian functions. Then added some noise to them and retest and then utilised real-world data to check. The testing framework is prepared using pytest, with checks for accuracy, robustness, and performance.

The test case data is 9 set of predefined multi-modal distributions covering: Single-mode (Cases 1-3), Bi-modal (Cases 4-6), Tri-modal (Cases 7-9). Again theses 9 cases are contaminated by noise injection. A Gaussian noise (2% of max concentration) is added to test case data to (basically deviate from very simple case) simulate real-world variability. Next real-world data loaded from hyy\_dmps\_data\_2023\_may.csv (DMPS instrument data) is used.

**The Aerosol fitting function is failing in some cases either due to kneedle not found or due to LSQ failure. I have seen that in most of the failure cases if I retry the function then it worked and fitted. So, I have created a Retry logic named safe\_fit\_row() which if you find useful then after a bit modification can be incorporated to the fitting module with necessary changes (basically the safe\_fit\_df function).**

The safe\_fit\_row() function is a fitting wrapper designed to enhance reliability in multimodal aerosol distribution fitting. It incorporates retry logic to handle the failures of knee-point detection issues (e.g., "kneedle was none") and least-squares (LSQ) fitting errors. When LSQ fails, it adaptively doubles the sample size to improve convergence. If all retry attempts are exhausted without success, the function gracefully falls back to a 2-mode fitting strategy to ensure output continuity. For consistent and reproducible results across runs, a fixed random seed (np.random.seed (42)) is also used.

**Test Strategy: Validation Metrics**

In the test strategy I have tried to check basic scientific accuracy (mode counts, total no. concentration, minimum mode wise concentration) and fitting time for computational efficiency of the multimodal fitting process.

| **Metric** | **Tolerance** | **Purpose** |
| --- | --- | --- |
| **Mode Count** | ± *mod\_relax* (default: 3 modes) | Checks if the number of fitted modes aligns reasonably with expected counts. |
| **Total Concentration** | ± 25% of observed (*obs\_to\_fit\_conc\_ratio*) | Validates total number concentration by comparing fitted and actual concentrations. |
| **Mode Concentration** | > *conc\_min\_* (default: 10 cm⁻³) | Check out low-concentration modes that are likely artifacts or noise. |
| **Runtime** | ≤ *t\_fit\_max* (default: 15 sec) | Ensures that fitting remains computationally practical for large-scale analyses. |

Other files are to validating the processing pipeline, checking real-world data compatibility, testing error handling and edge cases, Verifying performance characteristics.

There were two issues in fitting code, that I faced

1. Different no. of modes on trying many times the fitting code.
2. At times, failing of the fitting code.

**TLDR**: This issue was largely resolved by reattempt the fitting code and incorporating (random seed) and fallback logic and if kneedle failed and doubling the sample drawn in the safe\_fit\_row function.

1. **Different no. of modes on trying many times the fitting code:**

By reattempting the fitting code multiple times, I noticed that the number and properties of the detected modes varied between runs. Suspecting that this variability was due to randomness in sample selection, I introduced np.random.seed(42) to ensure consistent sampling. This significantly reduced the inconsistency. However, a contradiction remained: if the fitting failure is due to a least-squares (LSQ) optimization error, then simply reattempting with the same (seeded) samples won't change the outcome, since the LSQ failure will likely repeat. In such cases, increasing the sample size becomes necessary to avoid repeated failure which I did in the safe\_fit\_row function by doubling the number of samples in reattempts.

I tried fitting multiple times the same data frame and it gives different result each time.

* Captured total concentration very well.
* But variation in number of modes is considerable at times.

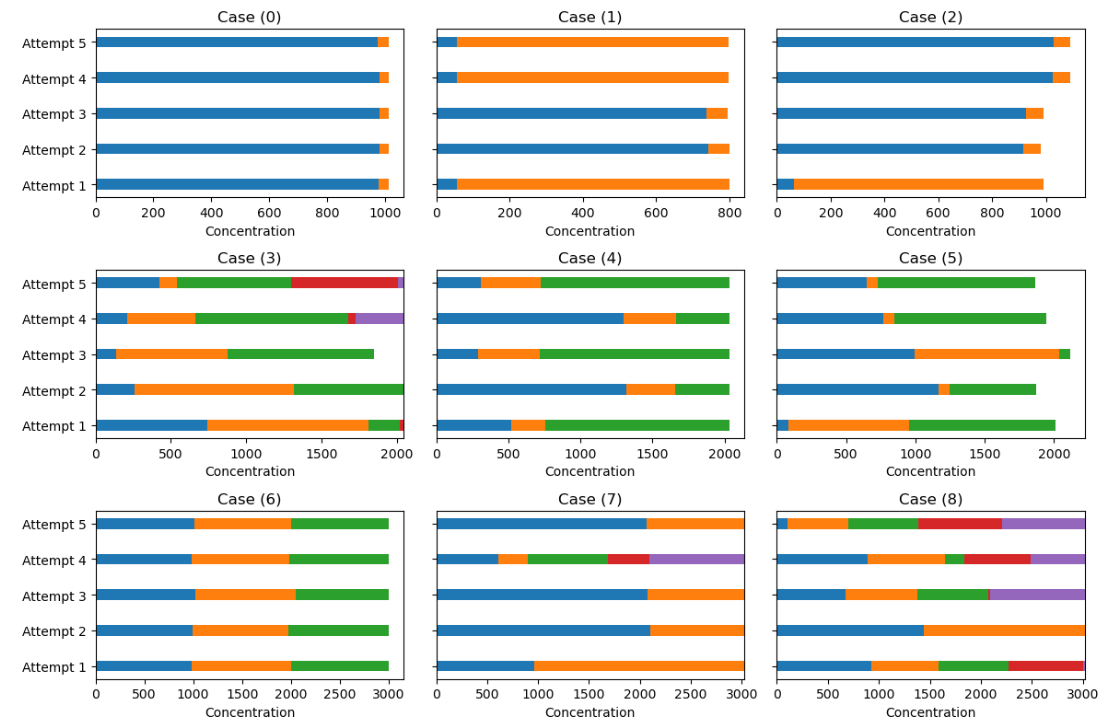
(Since fall back mode is set to two in my safe\_fit\_row function so no. of modes = 2 is understandable but for other times when fitting succeeded there is variation in no. of modes but total concentration it captures quite well.)

A graph of different types of cases

AI-generated content may be incorrect.

If we see how the distribution among the modes changes on each reattempting then we see that

* For the case where only two modes are, it just shifted back and forth
* The variation across attempts illustrates a common challenge in GMM fitting - the algorithm can converge to different local optima depending on random initialization, leading to different numbers of detected modes and varying component proportions even when analyzing the same underlying data.



Checking How the mode distribution changes in each attempt for test case data.

Same plots for test cases without any noise introduced:

A group of colorful bars

AI-generated content may be incorrect.

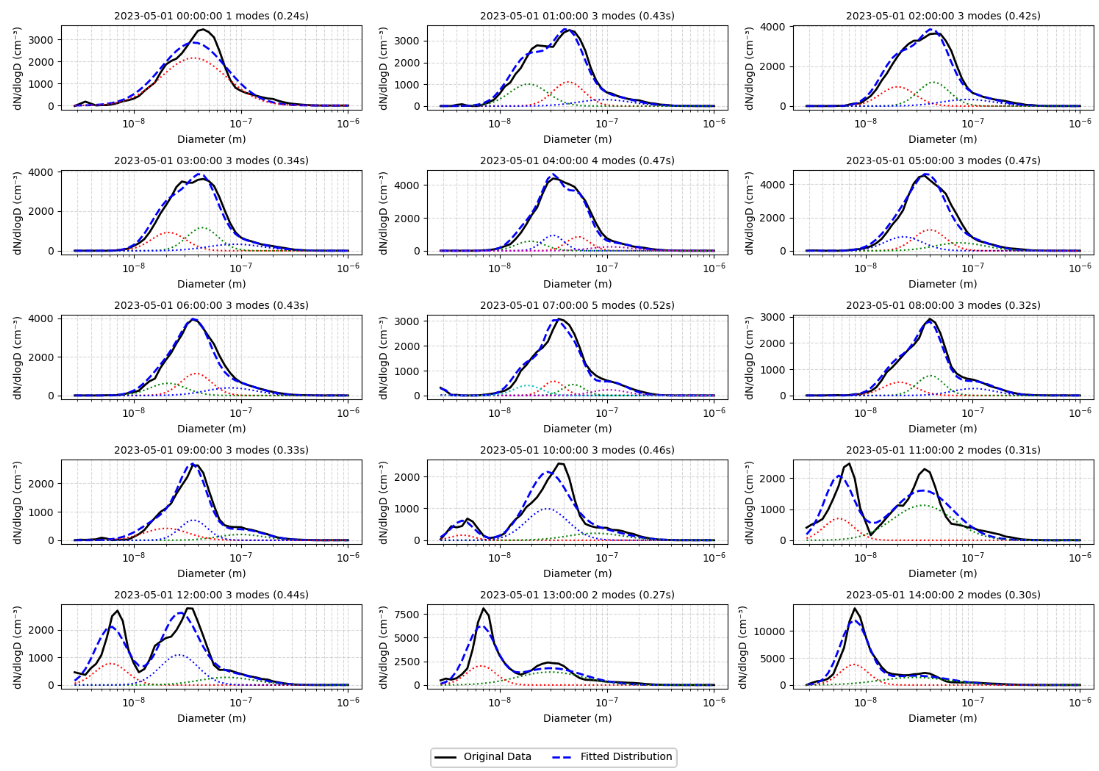
A graph of a number of cases

AI-generated content may be incorrect.

**For real data:**

A group of graphs showing different sizes of data

AI-generated content may be incorrect.

****

**A graph with many colored dots

AI-generated content may be incorrect.**A graph showing the amount of time

AI-generated content may be incorrect.

1. **Failing of the fitting code:**

The fit\_multimode function is failing in two key places:

* KneeLocator (kneedle) fails to determine the optimal number of modes.
* Least-squares (LSQ) fitting fails to converge to a valid solution.

Generally below cases can be problematic

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Kneedle Fails? | LSQ Fails? | Likely Reason |
| Single broad mode | Yes | No | No clear AIC "elbow" |
| Noisy, flat distribution | Yes | Yes | No discernible peaks |
| Too few data points (<5) | Skipped | Skipped | Insufficient data for fitting |
| Overlapping modes | No | Yes | LSQ cannot separate degenerate Gaussians |
| GMM gives bad initial guess | No | Yes | LSQ diverges due to unrealistic parameters |

1. if Kneedle Fails, can we force n modes to some default value like 2.

**B. Key Causes of LSQ Failure**

1. **Poor initial guesses from GMM**: Means (g["mean"]) or sigmas (g["sigma"]) may lie outside valid ranges.
2. **Ill-conditioned bounds:** -np.inf/np.inf bounds can lead to numerical instability.
3. **Overlapping/degenerate modes:** Two Gaussians with near-identical parameters confuse the solver.
4. **Noisy/flat data:** Lack of clear peaks makes convergence difficult.

To rescue the least-squares (LSQ) fitting failure, we can address as:

1. Constrain Bounds Physically: Replace `-np.inf`/`np.inf` with realistic bounds based on the data: Amplitude ≥ 0, Mean ≥ smallest diameter, Sigma ≥ 10% of bin width, Amplitude ≤ 2× max observed, Mean ≤ largest diameter, Sigma ≤ data range

2. Add Parameter Validation: Check GMM outputs before passing to LSQ like max of diameter <= mean of gaussian <= max of diameter, if it is not then can use median, similarly sigma <= 0.

3. Fallback to Simpler Fits: If LSQ fails, reduce n\_modes and retry.

4. Use Robust Optimization: Switch to other methods like scipy.optimize.least\_squares for better stability.

Let me know if you'd like further refinements!

Assertion with the values

Mode contribution check:

If a 0.5 < mode/total mode < 0.95 then only keep it

.