Supporting Information

PySpecTrace: Python-based Graphical User Interface for Real-time Spectroscopy Analysis

Sigit Khoirul Anam[1,2], Suwardi[1], Ferry Anggoro Ardy Nugroho[3], Iwan Darmadi[1,*]

[1]Research Center of Photonics, National Research and Innovation Agency, B.J. Habibie Science and Technology Park, Serpong, Indonesia

[2]Department of Chemistry Education, Faculty of Mathematics and Natural Sciences, Universitas Negeri Yogyakarta, Jl. Colombo No.1, Yogyakarta, Indonesia

[3]Department of Physics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia, Depok, Indonesia

*corresponding author: iwan.darmadi@brin.go.id

Table of Contents

I. ARCHITECTURE 1

A. CLASS DIAGRAM 1

1. Model 1

2. View 2

3. Presenter 3

B. DATA FLOW DIAGRAM 4

II. SPECTRUM OUTPUT MODE AND BACKGROUND-BASELINE CORRECTION 4

III. SPECTRUM FITTING AND TRACING 5

A. PEAK POSITION 5

1. Gaussian 5

2. Lorentzian 5

3. Polynomial 6

B. CENTROID 6

C. PEAK WIDTH 6

1. Gaussian 6

2. Lorentzian 7

3. Polynomial 7

D. NON-PEAK TRACING 7

I. Architecture

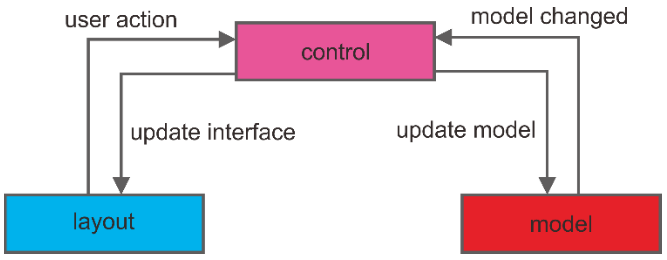

Figure SI 1 Global architecture of Model-View-Presenter. The Layout represents the View, while the Controller acts as the Presenter.

The PySpecTrace employs the Model-View-Presenter (MVP) architecture (Figure SI 1). This framework systematically arranges the application's structure to enhance clarity and comprehensibility. The Model is responsible for data management, the View for rendering data and capturing user inputs, and the Presenter functions as an intermediary: it acquires data from the Model, processes it, and subsequently forwards it to the View for presentation. The MVP architecture guarantees that each component maintains a distinct and well-defined purpose.

A. Class diagram

1. Model

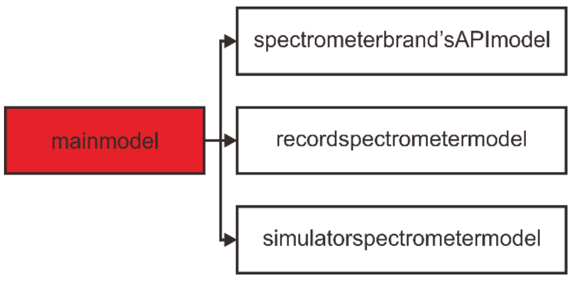

Figure SI 2 Flow diagram of the Model class in the Model-View-Presenter architecture, showing the relationships between modules.

MainModel serves as the main module, which then calls the underlying modules such as RecordSpectrometerModel, and SimulatorSpectrometerModel and Model from API some spectrometer manufacturer (Figure SI 2).

2. View

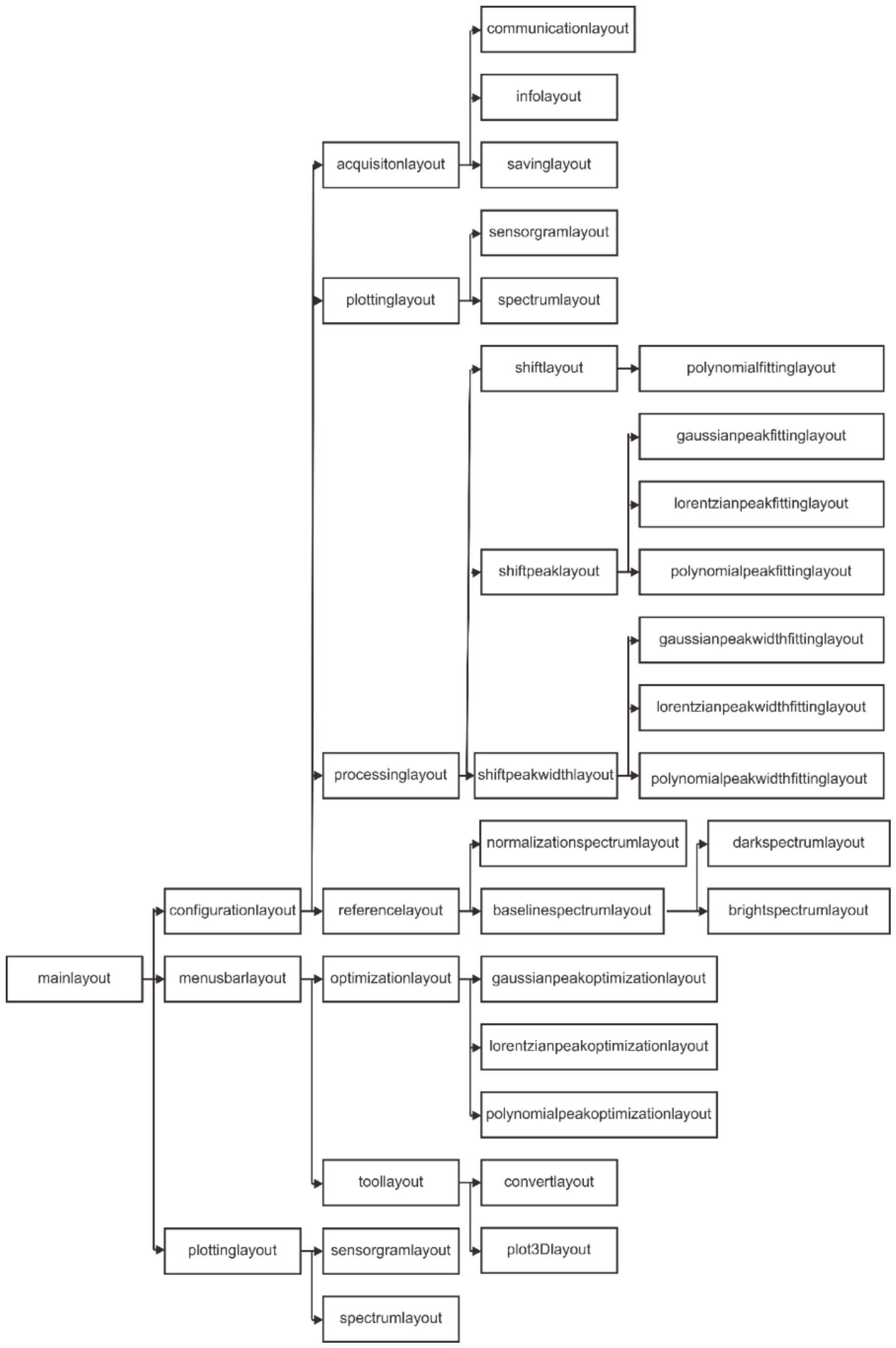

Figure SI 3 Hierarchy of the View Class.

The View class comprises several modules, with MainLayout serving as the primary module. This main module calls the underlying modules to enable their integration in presenting the interface to the user.

3. Presenter

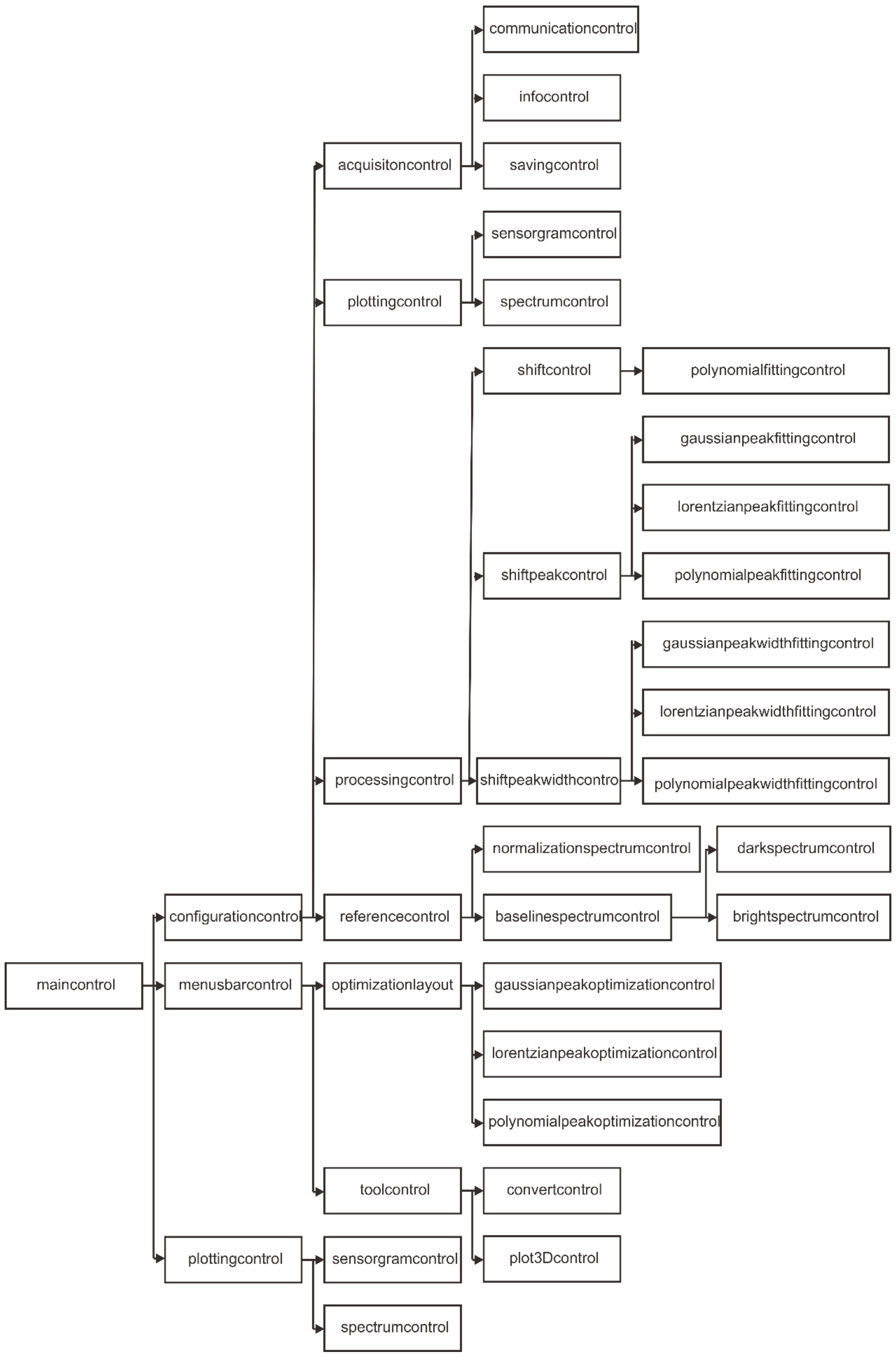

Figure SI 4 Hierarchy of the Presenter Class.

The Presenter is responsible for processing data from the Model and updating the View (Figure SI 4). It contains a main module, *MainControl*, followed by the underlying modules.

B. Data Flow Diagram

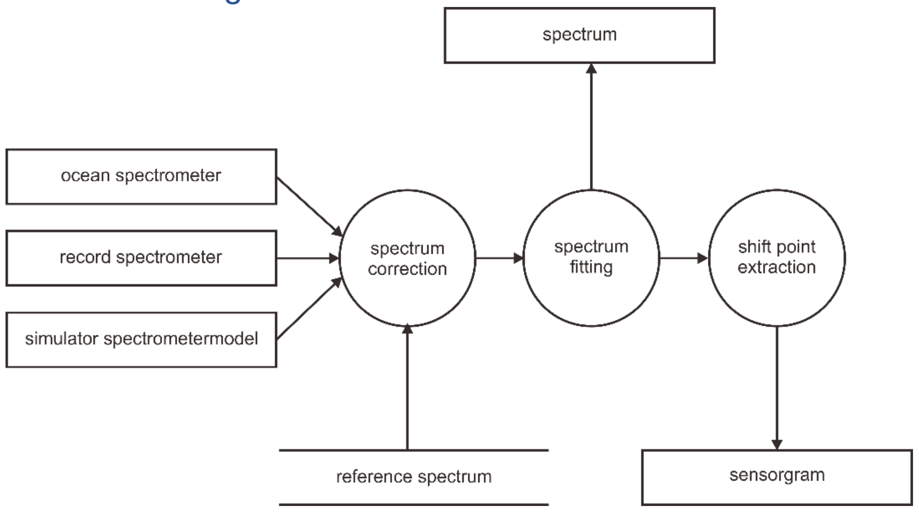

Figure SI 5 A data flow diagram illustrating the route of input data until it becomes output data for the user (spectrum and sensorgram).

The data flow diagram shows how the data source (input) is directed to the user (output) (Figure SI 5). The data source is the spectrum chosen by the user. This data is then normalized (corrected) using a reference spectrum selected by the user, involving the bright spectrum and the dark spectrum. After this step, options such as transmission or absorbance can be identified.

The next step is curve fitting, with various fitting function options available. The results of the curve fitting are then shown in the spectrum display. In the following process, each curve fitting method has its own descriptor extraction technique. This process generates descriptor values, which are then shown in the sensorgram display within the time domain.

II. Spectrum Output Mode and Background-Baseline Correction

The transmission and absorbance equations used to process the raw spectrum are explained below.

For transmission mode:

$$Transmission = \frac{I_{raw} - I_{dark}}{I_{bright} - I_{dark}} - I_{base}$$

where I_{bright} is the bright background spectrum, I_{dark} is the dark background spectrum and I_{base} is mean of baseline.

For absorbance mode:

$$\begin{aligned} Absorbance &= -\log(Transmission) \\ &= \log\left(\frac{1}{Transmission}\right) \\ &= \log\left(\frac{100\%}{Transmission}\right) \\ &= \log(100) - \log Transmission \\ &= 2 - \log\left(\frac{I_{raw} - I_{dark}}{I_{bright} - I_{dark}} - I_{base}\right) \end{aligned}$$

III. Spectrum Fitting and Tracing

All the fitting and tracing functions of PySpecTrace are described below:

A. Peak position

1. Gaussian

$$f_{fit}(\lambda) = A \cdot exp\left(-0.5 \cdot \left(\frac{\lambda - \mu}{\sigma}\right)^2\right) + c$$

where λ is wavelength, A is the peak amplitude, μ is the peak center, σ is deviation standard and c is the peak baseline. In the GUI, the fitting output "**x-axis**" is given by μ while the fitting ou μ, while the fitting output "**y-axis**" is given by A. The inflection point was calculated by setting the second-order derivative to zero as in:

$$\frac{d^2 f_{fit}(\lambda_{inflection})}{d\lambda^2} = 0$$

2. Lorentzian

$$f_{fit}(\lambda) = \frac{A}{1 + \left(\frac{\lambda - \mu}{\sigma}\right)^2} + c$$

where λ is wavelength, A is the peak amplitude, μ is the peak center, σ is deviation standard and c is the peak baseline. In the GUI, the fitting output "**x-axis**" is given by μ while the fitting ou μ, while the fitting output "**y-axis**" is given by A. The inflection point was calculated by setting the second-order derivative to zero as in:

$$\frac{d^2 f_{fit}(\lambda_{inflection})}{d\lambda^2} = 0$$

3. Polynomial

$$f_{fit}(\lambda) = \alpha_n \lambda^n + \alpha_{n-1}\lambda^{n-1} + \ldots + \alpha_1 \lambda + \alpha_0$$

where λ is wavelength with its respective fitting coefficient. In the GUI, the peak position is given by locating the maximum value of $f_{fit}(\lambda)$. The inflection point was calculated by setting the second-order derivative to zero as in:

$$\frac{d^2 f_{fit}(\lambda_{inflection})}{d\lambda^2} = 0$$

B. Centroid

$$\lambda_{centroid}(t) = \frac{\int_{\lambda_s}^{\lambda_s+S} \lambda\left(f_{fit}(\lambda) - f_{base}\right)d\lambda}{\int_{\lambda_s}^{\lambda_s+S}\left(f_{fit}(\lambda) - f_{base}\right)d\lambda}$$

$$= \frac{\sum_{i=0}^{n}\left[\frac{P_{i+1}}{i+2}\left((\lambda_s + S)^{i+2} - \lambda_s^{i+2}\right)\right] - \frac{f_{base} \times S}{2}(2\lambda_s + S)}{\sum_{i=0}^{n}\left[\frac{P_{i+1}}{i+1}\left((\lambda_s + S)^{i+1} - \lambda_s^{i+1}\right)\right] - f_{base} \times S}$$

where S, n, and f_{base} represent the span, order of the polynomial, and the baseline. Currently, PySpecTrace only provides the x-coordinate of the centroid descriptor.

C. Peak Width

1. Gaussian

$$f_{fit}(\lambda) = A \cdot exp\left(-0.5 \cdot \left(\frac{\lambda - \mu}{\sigma}\right)^2\right) + c$$

where λ is wavelength, A is the peak amplitude, μ is the peak center, σ is deviation standard and c is the peak baseline. The FWHM is described as follows.

$$\Delta x = \sigma\sqrt{-2ln(50\%)}$$

$$x_1 = \mu - \Delta x \qquad x_2 = \mu + \Delta x$$

$$FWHM = |x_2 - x_1|$$

2. Lorentzian

$$f_{fit}(\lambda) = \frac{A}{1+\left(\frac{\lambda-\mu}{\sigma}\right)^2} + c$$

where λ is wavelength, A is the peak amplitude, μ is the peak center, σ is deviation standard and c is the peak baseline. The FWHM is described as follows.

$$\Delta x = \sigma \times \sqrt{\frac{A}{50\% \times A} - 1}$$

$$\Delta x = \sigma \times \sqrt{2-1}$$

$$\Delta x = \sigma$$

$$x_1 = \mu - \Delta x \qquad x_2 = \mu + \Delta x$$

$$FWHM = |x_2 - x_1|$$

3. Polynomial

$$f_{fit}(\lambda) = \alpha_n\lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1\lambda + \alpha_0$$

The FWHM is calculated by, first, determining the half-maximum level:

$$y_{mid} = \frac{min(f_{fit}(\lambda)) + max(f_{fit}(\lambda))}{2}$$

Next, the program will find the intersection between $y_1 = y_{mid}$ and $y_2 = f_{fit}(\lambda)$ which may result in multiple intersecting λ_{cross} :

$$\lambda_{cross} \approx \lambda_k + \frac{y_{mid} - y_k}{y_{k+1} - y_k} \times (\lambda_{k+1} - \lambda_k)$$

From all the intersection points, two closest points to the peak position λ_{peak} are selected:

$$\lambda_{start}, \lambda_{end}$$

the FWHM is calculated as:

$$FWHM = |\lambda_{end} - \lambda_{start}|$$

D. Non-Peak Tracing

$$f_{fit}(\lambda) = \alpha_n\lambda^n + \alpha_{n-1}\lambda^{n-1} + \cdots + \alpha_1\lambda + \alpha_0$$

The non-peak tracing employs polynomial fitting to the data, where the order n can be adjusted accordingly. Note that when the order n is set to 0, the descriptor output will be the actual intensity at the given λ point without prior fitting.