# Fitspy: A Python package for spectral decomposition

Patrick Quémeré ® [1]

1 Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France

Summary

`Fitspy` is a dedicated tool to **fit sp**ectra in **py**thon.

Spectrum decomposition (also known as fitting) is the choice of a linear combination of peak models to best represent an experimental spectrum. `Fitspy` currently has implementations for Gaussian, Lorentzian, related asymmetric functions and the Pseudovoigt model. User-defined models can also be added. For the fitting, the model parameters can be subject to bounds and constraints. From a practical point of view, the `Fitspy` GUI (see Figure 1) is designed to be as simple and intuitive to use as possible.
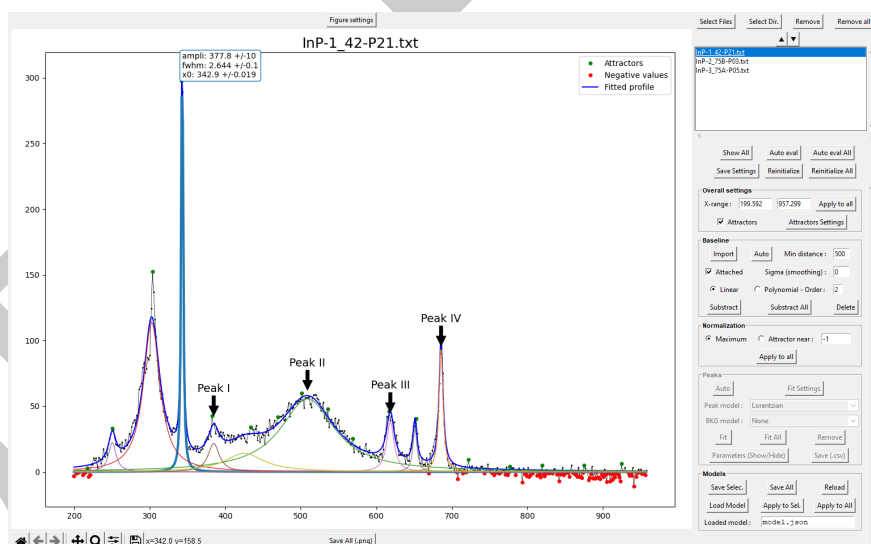
Figure 1: The `Fitpsy` GUI.

Statement of need

The analysis of spectra in many areas of physics, from materials characterisation to astrophysics, often requires their decomposition into more or less complex models to estimate the chemical composition of the subject being analysed. To carry out these decompositions, research communities can rely on spectral decomposition tools. While many open-source tools for spectral fitting exist, most of them have, however, been designed for specific application domains, offering a broad range of services beyond mere spectral fitting. Consequently, these tools can prove challenging to use, especially for less experienced individuals.

In the vein of generic tools like `Fityk` (Wojdyr, 2010) or PRISMA (Flores et al., 2022), `Fitspy` is a dedicated tool for spectral fitting — and only spectral fitting — with the following characteristics or functionalities:

Quémeré. (2024). Fitspy: A Python package for spectral decomposition. *Journal of Open Source Software*, *0*(0), 5868. https://doi.org/10.xxxxxx/ draft.

- **Agnostic Nature**: `Fitspy` is not tied to any specific physical quantity or database. It is designed to process spectra regardless of their x-support and y-intensity without any prior knowledge.

- **Python Implementation**: `Fitspy` is coded in Python. As a result, for individuals with basic knowledge of the language, spectra can be easily processed by scripts.

- **2D Maps**: `Fitspy` has been designed to handle spectra derived from 2D acquisitions. Note that "2D" can encompass time or any other dimension. When dealing with 2D data, an interactive map in the `Fitspy` GUI allows users to locate and select spectra of interest easily.

- **Multiprocessing Capabilities**: `Fitspy` enables spectral fit processing on multiple processors, enhancing efficiency.

- **Constrained Parameters**: Leveraging the `lmfit` library (Newville et al., 2014), `Fitspy` empowers users to impose constraints on parameter ranges or establish constraints between parameters using literal expressions.

- **Simple GUI**: `Fitspy` has been designed to be as intuitive and simple to use as possible (subjective criterion).

To the author's knowledge, although many open-source softwares are much more advanced in certain aspects mentioned, none of them seems to encompass all the functionalities described above. Therefore, the features of `Fitspy` make it an ideal tool for quickly fitting a few spectra through its GUI or for fitting several thousand of spectra (or more) by python batches, as can occur in the context of large-scale parametric studies (Meyer et al., 2023).

`Fitspy` workflow short description

`Fitspy` accepts at this time two types of input data file formats:

- a first format where each spectrum is stored individually in a `.txt` file. The input data file consists of a 2 columns base format where the first column is associated to the physical support of the spectrum (typically wavelengths) and the second to the corresponding spectrum intensity.

- the second format is associated to 2D-maps of spectra acquisitions. The related spectra are stored in a single file in which each spectrum is identified thanks to its grid coordinates (X,Y).

Once loaded in the GUI via a files selector widget, the spectra can be processed one by one or in groups (depending on spectra selected with the cursor in the file selector widget), or as a whole dataset.

Firstly, the users can choose to reduce the physical support (wavelength axis) to a range of interest for subsequent processing. They can also activate the `Attractors` capability, which are the local maxima of spectra obtained with the *signal.find_peaks()* function of the `scipy` library (Gommers et al., 2023). These attractors can be used subsequently to normalise spectra or to select points associated with the baseline and peaks to be modelled (see below).

Once the range of interest and the spectra are (optionally) normalised, the users can define a **baseline** in order to make the spectra share a common, flat zero-intensity level. This baseline can be defined in an automatic way or by the users, who positions characteristic points on the figure with the mouse. These points can be left at their initial positions or attached to the spectrum profile (`Attached` mode). In the latter case, the points can be attached either to the raw spectrum or to an averaged spectrum if the users wish to attenuate the effects of noise. The baseline is then approximated over the entire spectrum range using either piecewise linear interpolation or an *n*-order polynomial approximation.

Once the baseline has been subtracted, the next step consists of defining the **peaks** of interest for the decomposition. This can be done either directly in the figure by clicking, or by an `Auto` mode which consists of an iterative process to automatically determine the main peak locations. Thanks to the functionalities offered by `lmfit`, each peak can be associated to a predefined model (a Gaussian model, a Lorentzian model, their asymmetric variants, or a Pseudovoigt model) or a user-defined model. These models rely essentially on 3 main parameters: a position, an amplitude and a width (full-width half-maximum, FWHM). This width can be differentiated *left and right* in the frame for an asymmetric model.

For the fit, a additional background model can be added. **Bounds** can be imposed for each of the parameters as well as associated **constraints**. If required, these constraints can be used to link model parameters together, for instance in the case where a physical ratio between the amplitudes of 2 peaks is expected.

Finally, the parameters can be free to evolve during the fit processing or maintained fixed at their initial values.

Several fit options are also proposed, such as:

- taking into account only the positive values of the spectra.
- a (limited) choice of minimisation algorithms
- the maximum number of fit iterations
- the number of CPUs to be used during the fit processing.

At the end of the fit, the fit parameters and the statistics returned by `lmfit` are displayed in related widgets and can be exported in .csv and .txt files respectively.

All the processing steps previously described constitute a model in the sense of the `Fitspy` application that can be saved (in a .json file) and easily replayed as-is or applied to other spectra datasets.

In terms of visualization, the GUI allows the users to display all of the spectra simultaneously (`Show All` mode) or, when a spectrum is selected, to display the corresponding fit decomposition and its residual. Figure titles and axis names can be customized and peak models labeled.

All the actions described above can be performed through python scripts without using the GUI. In practice, when performing repeated analyses, for users having some Python skill, it may be useful to use `Fitspy` as follows:

1. create a `Fitspy` model with the GUI
2. save the model
3. apply the model to several sets of spectra using a python script

This is this approach that has been recently used by teams at CEA to process tens of thousands of PhotoLuminescence and Raman spectra acquired on wafers, taking advantage of the parallelism capabilities offered by `Fitpsy` (see Figure 2).
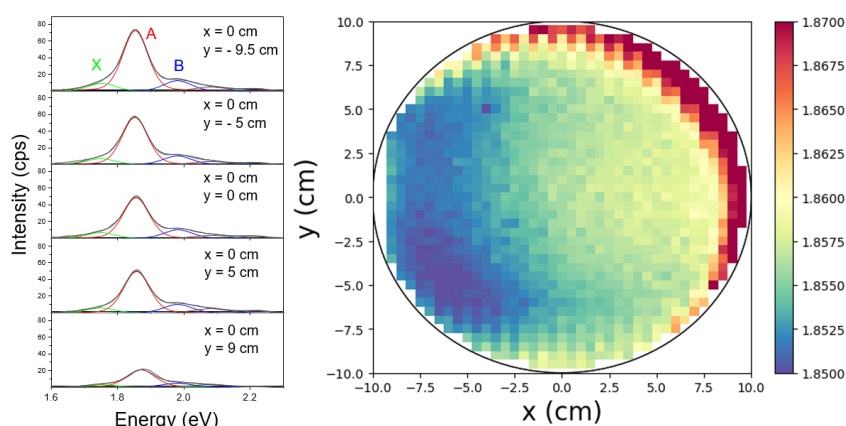
Figure 2: Example of a `Fitpsy` application used in photoluminescence to characterize exciton intensities on a wafer (Meyer et al., 2023).

Acknowledgements

References

Flores, E., Mozhzhukhina, N., Li, X., Norby, P., Matic, A., & Vegge, T. (2022). PRISMA: A robust and intuitive tool for high-throughput processing of chemical spectra. In *Chemistry–Methods* (No. 10; Vol. 2). https://doi.org/10.1002/cmtd.202100094

Gommers, R., Virtanen, P., Burovski, E., Haberland, M., Weckesser, W., Oliphant, T. E., Reddy, T., Cournapeau, D., alexbrc, Nelson, A., Peterson, P., Roy, P., Wilson, J., Polat, I., endolith, Mayorov, N., Walt, S. van der, Brett, M., Laxalde, D., … Kai. (2023). *Scipy/scipy: SciPy 1.11.1* (Version v1.11.1). Zenodo. https://doi.org/10.5281/zenodo.8092679

Meyer, T., Quéméré, P., Brunet, P., Rolland, E., Cadot, S., & Le Van-Jodin, L. (2023). *Full wafer-scale characterization method for 2D materials*. *10th International Conference on Advanced Applied Raman Spectroscopy*. https://www.ramanfestconf.com/2023/Abstracts/2023_Meyer_Thibaut_52.pdf

Newville, M., Stensitzki, T., Allen, D. B., & Ingargiola, A. (2014). *LMFIT: Non-linear least-square minimization and curve-fitting for python* (Version 0.8.0). Zenodo. https://doi.org/10.5281/zenodo.11813

Wojdyr, M. (2010). *Fityk*: a general-purpose peak fitting program. In *Journal of Applied Crystallography* (5 Part 1; Vol. 43, pp. 1126–1128). https://doi.org/10.1107/S0021889810030499