**PrestoPronto**: a software package for large EXAFS data sets

Carmelo Prestipino*

University of Rennes 1, CNRS, 263 Avenue du Général Leclerc, 35042 Rennes, France. *Correspondence e-mail: carmelo.prestipino@univ-rennes1.fr

**Keywords**: Quick-EXAFS; PrestoPronto; dispersive EXAFS.

**PrestoPronto** is a free and open-source collection of programs aimed at performing X-ray absorption spectroscopy analysis of large data sets. The software is composed of three programs: (i) **PrestoPronto_GUI**, which imports spectra in various formats, pre-processes data and performs classical data analysis, (ii) **LinearCom_GUI**, which performs linear combination analysis, and (iii) **PCA_GUI**, which performs principal component analysis. All parts of the software package include practical and intuitive graphical user interfaces (GUIs) with plotting capabilities. The main objective of these programs is to quickly and interactively monitor time-resolved experiments from Quick-EXAFS (QEXAFS) and dispersive EXAFS beamlines.

1. **Introduction**

Since the early development of rapid-acquisition X-ray absorption spectroscopy, such as dispersive EXAFS (Kaminaga et al., 1981; Matsushita & Phizackerley, 1981) and Quick-EXAFS (QEXAFS; Frahm, 1988; Stötzel et al., 2010), time-resolved experiments have become more and more widespread. Modern EXAFS beamlines now implement acquisitions methods with a time resolution spanning from minutes to seconds or even lower (a few milliseconds) on beamlines equipped with dedicated optics (Müller et al., 2016).

With the availability of such facilities and the high photon flux delivered by third-generation synchrotron sources, the acquisition of hundreds or thousands of spectra in each experiment is usual, indicating the utility of a tool to efficiently follow the modification of the spectra in real time, without which the analysis can become tedious and time-consuming.

These reasons motivated the development of **PrestoPronto** (from the Italian for ‘soon ready’), interactive software with a graphical user interface (GUI) for the analysis of large data sets (hundreds or thousands of spectra), as described in the present contribution. Several codes for analysis have already been developed with the same objective (San-Miguel, 1995; Ressler, 1998; Sanchez del Rio & Dejus, 2004; Stötzel et al., 2012), and the use of macro languages such as **IFEFFIT** or **LARCH** (Newville, 2001b, 2012; Newville & Ravel, 2020) can help in the analysis of large data sets. However, no open-source code in a high-level programming language is available, while the learning curve for analysis packages based on macro languages is rather steep and relatively time-consuming for users without coding experience.

The main goal of the **PrestoPronto** set of programs is to cover the requirement to screen and analyze data in real time, dealing with numerous sets of spectra in a quick, intuitive and easy way, using the algorithms most commonly used by the XAFS community, while maintaining the possibility of
customizing the code for specific requirements. The interfaces of the codes have been designed *ad hoc* to deal with large data sets, allowing the user to rapidly evaluate the parameters related to the implemented algorithms and to efficiently plot data-analysis results.

The *PrestoPronto* package has already been used in several studies (Bahout *et al.*, 2012; Achilli *et al.*, 2014; Moog *et al.*, 2014; Zuvich *et al.*, 2014; Nilsson *et al.*, 2015). It is free and open source and can be adapted to specific requirements such as different data formats or experimental setups or different types of analysis.

2. Structure of the code

The code is written in Python 2.7 using several common libraries: the themed Tk/tkinter graphical toolkit for user interfaces, Matplotlib (Hunter, 2007) for plotting and NumPy (Walt *et al.*, 2011) for the computation of matrices and arrays. For more XAFS-specific algorithms, the software is mainly built on the methods and functions coded in *LARCH*, the successor to the *IFEFFIT* macro language (Newville, 2001b, 2012; Newville & Ravel, 2020). The sequence of spectra is stored in a modified Python list, in which each element is a class containing a spectrum with associated methods and properties. The codes are not fully parallelized: the sequential fit approach (see below) implies that parallelization must be implemented at the array-computation level, with a consequent complexity of optimization. At the moment, only the few parts of the codes that use NumPy operations based on Basic Linear Algebra Subroutines (BLAS) can run in parallel; further parallelization is dependent on future development of NumPy.

In order to improve the user-interaction and data-processing experiences, the codes upload the entire sequence of spectra into random access memory (RAM) and create new arrays at each analysis step. This approach implies that the code is limited to handling data sets smaller than the available memory. Although in conventional QEXAFS, with a time resolution of seconds or minutes, this is not a practical limitation (the program currently works with a $1300 \times 1050$ data matrix), it is worth noting that for a time resolution lower than 100 ms, as attainable using the most time-resolved optimized beamlines (Müller *et al.*, 2016; Pascarelli *et al.*, 2016), this limitation could be reached in experiments of longer than an hour. The software package is composed of three programs: *PrestoPronto_GUI*, *Linearcomb_GUI* and *PCA_GUI*.

3. *PrestoPronto_GUI*

This module is the core of the package. *PrestoPronto_GUI* uploads a sequence of spectra into memory, performs the calibration, averages and resamples the data and finally performs an XAS analysis.

At the start of the program, a GUI appears as a single notebook window (Fig. 1) composed of six panes corresponding to the main steps of data analysis (Data input, Averages, XANES, EXAFS-FT, Fit and Attributes) and a terminal window used to communicate error detection. Each pane allows a set of parameters for the corresponding analysis step to be defined, the data to be processed and the results to be saved as image or ASCII files. In order to avoid multiple data processing, in each pane the parameters can be optimized in dedicated interfaces, in which the parameter values can be selected directly on the plot and applied interactively to the

![Figure 1](image.png)

*Figure 1* Appearance of *PrestoPronto_GUI*. (a) Data input tab. (b) Plot windows with shift slider.
different spectra just by moving a slider, as shown in Fig. 2 for EXAFS extraction.

In the ‘Data input’ tab, QEXAFS or dispersive EXAFS data are imported as a sequence of multicolumn ASCII files, each containing the energy and detector readings for one spectrum. Possible energy shifts can be corrected if a reference spectrum has been collected simultaneously. In the interface, it is possible to associate each spectrum with several supplementary attributes, for example temperature, valve status or current intensity, if the corresponding numerical values are present in the original data files. Alternatively, spectra can also be read from a multicolumn ASCII file containing a common energy column and a series of spectra.

The second tab, ‘Averages’, is dedicated to the binning and averaging of the spectra. This operation represents one of the most important steps in QEXAFS data analysis. Effectively, the energy sampling is typically defined before the experiment by the optimal energy resolution in the XANES region, i.e. below the core-hole width, resulting in an oversampling of the EXAFS region (Bunker, 2010). Moreover, if the evolution kinetics of the studied process is unknown a priori, the acquisition time adopted is generally also shorter than the optimal time. As a consequence, the data resulting from QEXAFS experiments are very often oversampled. The role of the algorithms implemented in this tab is to reduce the size of the data matrix without a significant loss of information, decreasing computation times and improving the quality of the results. This task is performed by the interpolation of each spectrum in photoelectron wavevector units and by reducing the number of spectra, averaging the data along the time coordinate. During this step, supplementary attributes are also averaged in order to maintain the same time resolution of the data set. The attribute arrays averaged consistently as for the data are available in the ‘Attributes’ tab, where they can be visualized or saved as ASCII files.

As previously mentioned, one of the main goals of PrestoPronto_GUI is to rapidly monitor the evolution of spectra during time-resolved XAFS measurements. This task is achieved by using the ‘XANES’ and ‘EXAFS-FT’ tabs to perform a qualitative data analysis. In the ‘XANES’ tab, the spectra are background-subtracted and normalized. It is possible to plot and save the evolution of the edge jump, the position of the first-derivative maximum and the normalized XANES integral in a defined energy range along the set. These features are quite effective during experiments, showing the investigated process kinetics or allowing systematic errors such as sample movements to be recognized. The ‘EXAFS-FT’ tab is devoted to EXAFS signal extraction and calculation of the Fourier transform using the XAFS-specialized functions AUTOBK (Newville et al., 1993) and XFFT imported from LARCH (Newville, 2012). The choice of function parameters is performed graphically and interactively, as shown in Fig. 2 for EXAFS extraction; the interface maintains the same names for parameters as the widely known ATHENA software (Ravel & Newville, 2005, 2020).

Finally, the ‘Fit’ tab is a GUI for the FEFFIT function (Newville, 2001a) of the LARCH engine that models the experimental \( k^n \chi(k) \) signal as a sum of paths. The phase and amplitude for each path can be calculated by means of an internal interface to the FEFF6 code (Rehr et al., 1992) for the first coordination shell in the more common geometries, or can be imported by reading an feffn.dat file calculated by more recent versions of the FEFF code (Rehr & Albers, 2000; Kas et al., 2020). For each path, it is possible to fit the coordination number \((n)\), path length \((r)\), mean-square displacement \((ss)\) and energy shift \((e_0)\). In order to improve the rate of convergence, each spectrum is fitted sequentially, i.e. the refined values of a spectrum are used as starting-parameter values for the fit of the next spectrum in the series. At the end of the sequential fit, the evolution of the agreement factors, the refined parameter values and their evaluated errors can be plotted and saved.

4. PCA_GUI

This module is devoted to principal component analysis (PCA) and accepts as input a multicolumn ASCII file such as those saved by PrestoPronto_GUI. PCA is a robust linear
algebra method that is able to determine the number of independent components in a series of spectra, assuming that each data point can be expressed as a linear sum of product functions (Gemperline, 2006), as described in equation (1), in which $A$ is an $n \times m$ data matrix ($n$ rows of spectra recorded at $m$ energy points), $T_k$ is the $n \times k$ matrix of principal component concentration and $V_k$ is the $m \times k$ matrix of principal components:

$$A = T_k V_k^T.$$  (1)

The method is based on the diagonalization of the variance–covariance matrix $[Z_{(m \times m)} = A^T A]$ of the data set, as shown in equation (2). If the data set and the calculation are free of errors, then the number of principal components $k$, i.e. the primary rank of the matrix, is equal to the number of eigenvalues that differ from zero in the $D$ matrix (Harman, 1976), and the corresponding eigenvectors are contained in the $V$ matrix. The $T_k$ matrix is calculated in accordance with equation (3).

$$V^T Z V = D,$$  (2)

$$T_k = AV_k.$$  (3)

However, experimental data are affected by statistical and systematic errors and the calculations are subjected to a numerical roundoff; therefore, diagonalization of the $Z_{(m \times m)}$ matrix always produces $m$ nonzero eigenvalues. Consequently, determining the number of principal components, i.e. discriminating between significant and negligible eigenvalues, is no longer a trivial task, but is the first and most important step in PCA and derived methods.

Numerous methods devoted to this task have been developed, as described in Malinowski (2002); however, it is worth noting that the use of different methods provides different results for the same data matrix and it is not apparent which one is the best (Gemperline, 2006). The $PCA\_GUI$ code has implemented the IND function (Malinowski, 1977), the significance level $F$-test (%SL) of the reduced eigenvalue function (REV; Malinowski, 1989), the residual standard deviation (RSD) and the method of determination of rank by median absolute deviation (DRMAD; Malinowski, 2009), as visible in the interface table in Fig. 3(a).

However, in the opinion of the author, determination of the rank of the data set should not be limited to only numerical analysis. All a priori knowledge of the studied system and a careful visual inspection of the data should be taken into account together with numerical analysis. The plotting capabilities of the program allow three rapid tests to be performed to help in evaluation of the primary rank.

(i) The evolution of the concentration of principal components (the $T_k$ columns) should be in reasonable agreement with the expected kinetics of the investigated process. Spurious components tend to vary very rapidly.

(ii) Only improvement of the description of a meaningful spectroscopic feature along the data set can justify an increase in the rank. Components related only by small intensity changes of previously described features can be caused by imperfect normalization or background subtraction.

(iii) The residual between the experimental and reconstructed data should be constant or vary only as a function of the parameter investigated during the experiment. In the case shown in Fig. 3, it is clearly visible that the residual increases between the 60th and 90th spectra, and consequently an increase in the rank should be considered.

Generally, the components found by PCA are not physically meaningful spectra, but a linear combination thereof. To
obtain the real spectra, it is necessary to perform a rotation in the vector space defined by the eigenvectors. To perform this task, PCA_GUI implements iterative transformation factor analysis (ITFA), a technique of self-modelling curve resolution (Fernandez-Garcia et al., 1995; Rossberg et al., 2003).

The first step of the method is to define an approximate concentration profile matrix $T_k$. A widespread approach, it makes use of the vectors obtained by a varimax rotation of the concentration matrix (Vandeginste et al., 1985). Such a rotation maximizes the sum of the squared factor loadings, i.e. it confines each component to only a few ranges at high concentration, reducing their concentration to zero or low values in other ranges. However, as suggested by Fernandez-Garcia et al. (1995), in order to reduce the weight of the arbitrary choice of the type of starting rotation, PCA_GUI uses as an approximate concentration profile a zero matrix in which the elements corresponding to the maximum concentration of the profile matrix obtained by varimax rotation have been replaced with 1. Such a matrix is generally called a ‘needle matrix’.

The second step of the method consists of performing an iterative projection of the approximate concentration profiles in the space of the concentrations in accordance with

$$T_{k+1} = T_k T_k^T T_k. \quad (4)$$

From a purely mathematical point of view, the projected target always gives an acceptable description of the data set; however, it still might not be physically acceptable. For this reason, appropriate constraints should be applied to the concentration profiles during the iteration. In PCA_GUI, by default the profile concentration must be positive and the sum of the concentrations for each spectrum should be 1. Other implemented constraints are optional; for example, the presence of only one maximum or concentration limits in a range of the data set.

When the iterative projection reaches convergence, the obtained concentration profiles satisfy equation (1) and the constraints and the corresponding spectra can be calculated by least squares from $A$ and $T$ (Gemperline, 2006). Nevertheless, it is important to underline that the results obtained from ITFA are not unique, but rather a specimen in the restricted space of rotation defined by the applied constrains.

5. LinearCom_GUI

The last program in the package is devoted to fitting a sequence of spectra as a linear combination of standards. Such an analysis can be applied to the normalized XANES, the derivative of the normalized XANES or the $k^\alpha \chi(k)$ spectra. In analogy with PrestoPronto_GUI, the fit is sequential, i.e. the starting values for the fitting parameter are set to be equal to the refined values obtained from the previous spectrum in the set. The code is built around the LMFIT Python library developed by Newville et al. (2014) and uses the Levenberg–Marquardt algorithm with parameters and standard errors calculated at a 1σ interval of confidence. The sequence of spectra and reference-compound spectra are imported as a multicolumn ASCII file. If necessary, the code automatically performs a cubic spline interpolation to define a common abscissa axis for experimental and standard spectra. Users can choose to analyse only a subset of the sequence and can graphically define the range of the fit.

6. Future developments

Presently in PrestoPronto_GUI the EXAFS is fitted with a basic sequential algorithm, in which all parameters are refined independently. Creating a new module with a dedicated interface will allow constraints between different path contributions and also between different spectra in the sequence to be introduced. Another possible improvement would be a better management of spectra import and the data matrix in order to avoid the RAM memory-size limit issue related to very large data matrices.

7. Resources

A project page for PrestoPronto is accessible at http://soonready.github.io/PrestoPronto/. A self-installer and the source code are available. The screen shots in this article were made on a Windows 7 computer. The programs may differ in appearance on other systems.

Acknowledgements

The author thanks Santiago Figueroa, Sakura Pascarelli and Mark A. Newton for help in developing the programs and Matt Newville and Marcos Fernández-García for kindly opening their codes.

References


