

EMG Explorer Instruction

Anaïs Monteils

June 24, 2024

Abstract

EMG Explorer is a modular interface for visualization and pre-processing for multi-channel signals. This report gives an overview of EMG Explorer's functionalities

1 Introduction

Filtering and having an overview of data are essential steps before feature extraction or in any analytical process. This interface can provide a robust foundation to support this processing stage. The key word: modularity.

Interface's functionalities

This interface was designed for High-Density Electromyography processing. HDEMG involves using a grid of electrodes to record muscle activity. Typically, a large number of electrodes, 32, 64 or more, are used. It can be common for some of them to become faulty during trials, making it useful to have a tool to easily visualize the recorded data for instance. Thus, this interface allows users to open multi-channel files and offers both single channel plots and multiple channels plots in the temporal and frequency domains. Additionally, users can create customized processing pipelines, tailor these pipelines to the variables in their files and generate reports on their datasets with specific metrics and displays.

Interface's development

The interface was developed to be modular. Each component is designed to be as independent as possible from the other, allowing an integration of new elements that follow a standardized structure. The implementation relies on the PyQt and PyQtGraph libraries, which provide respectively the framework and interactive plotting functions. Jinja2 and DataTree/Xarray are respectively used to support the report templating and the internal data structure.

Interface's future

The future development of this interface will be driven by user needs and creativity. On a small scale, users can add processing function files that will dynamically be incorporated to the interface. Additional features, such as new tabs or entire windows, can also be incorporated according to users requirements.

An overview of EMG Explorer's functionalities. Each section will be illustrated by the application of the described concepts on the example file "simulated_signals.nc".

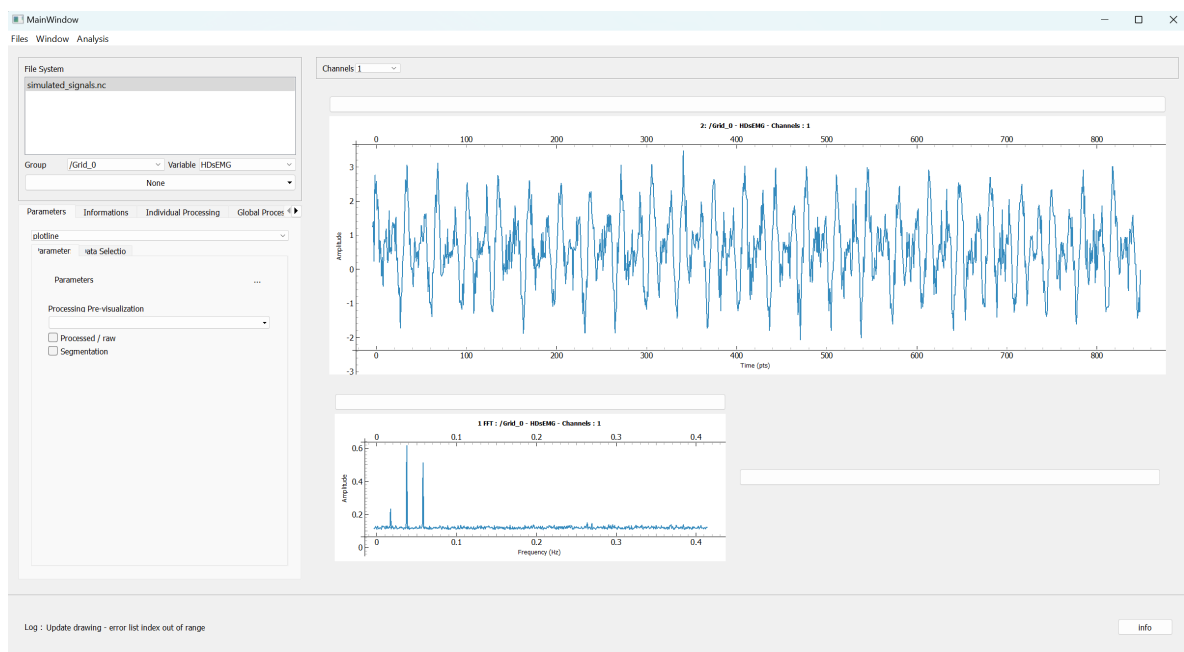


Figure 1: Principal Interface

2 Principal Interface Sections



Figure 2: Principal Interface Sections

The main components of the interface are the followings:

- File Systems & navigation : “File” in the menu bar enable the user to load or remove files. A File can be selected for visualization in the list. The comboBox Group, Variable, Channel enables the users to navigate the data
- Global Processing: Selects the global processing that will be applied to the files
- Log : Display the last line of the Log. The “Info” button displays a larger description of the log with its past outputs.
- Graphic Layout : Display the current graphics.
- Tabs
 - Informations: Displays the eventual meta-data of the file
 - Individual and Global Processing: Enables the user to create, save, open, modify processing pipelines
 - Parameters: Displays the parameters of the currently selected graphic

2.1 Load Data

Originally, the interface can only load .nc files.

The files are loaded according to hierarchical structure: Thus, each subsets of the dataset belongs to a group, have one or many variables that comprise one or multiple channels. The dimension is the name of the group of channels.

Example

Simulated_signals has three groups: Grid_0 , Grid_1 , Grid_2

Each groups has 2 variables: HDsEMG , Electrode_maps

HDsEMG has 64 “Channels” that name according to their number, from 1 to 64.

2.2 Visualization

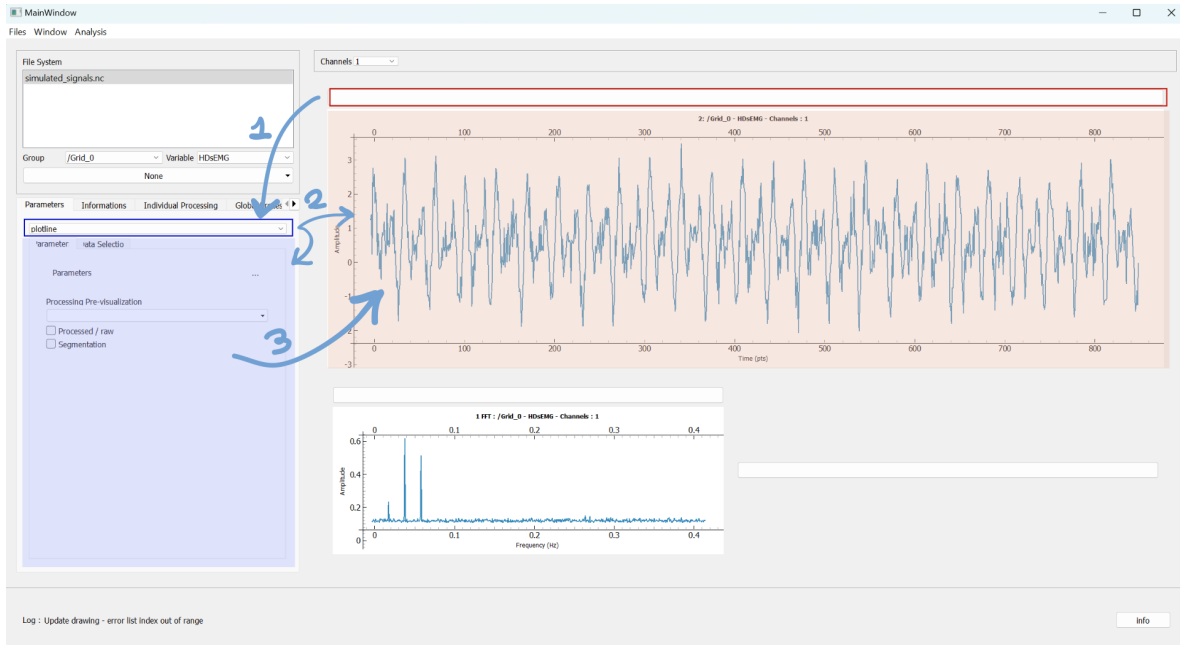


Figure 3: Visualization

Pushing the horizontal button of each graphic units updates the Parameters Tab by displaying the type of the currently selected plot and its parameters. Selecting a type of plots updates the Parameters Tab and the Plot of the Graphic Unit by displaying the parameters specific to the plot's type and the corresponding graphic.

Changing parameters in the Parameters Tab updates the corresponding Plot.

There are originally three types of Plot:

1. Graphic of a single channel with respect to time/points
2. Graphic of the FFT of a single channel with respect to frequency
3. Graphic of multiple channels with respect to time.points

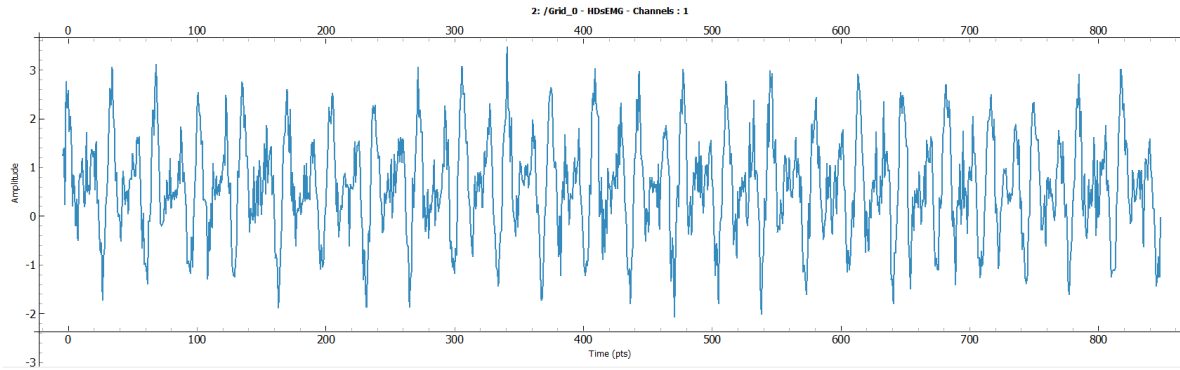


Figure 4: 1. PlotLine

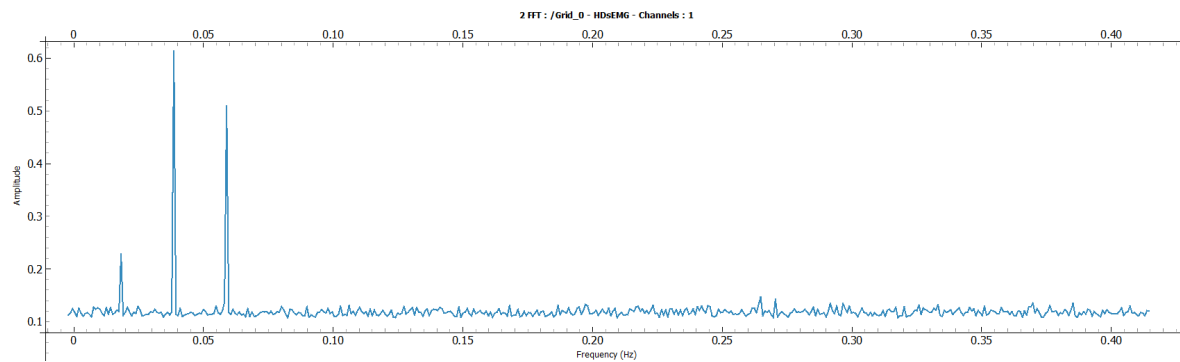


Figure 5: 2. FFT

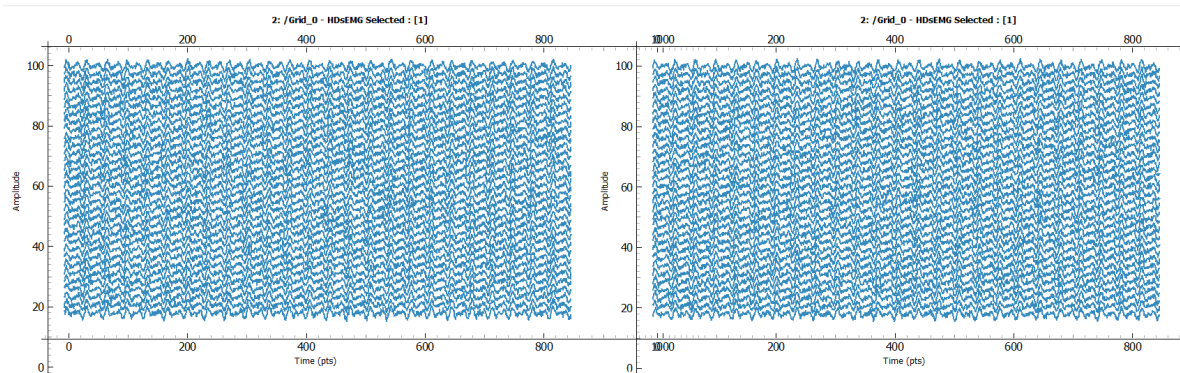


Figure 6: 3. Multiplot

2.3 Processing

Processing pipelines are a series of filter functions that can be applied to an array of values. These processing pipelines can be saved and edited as JSON files.

There are two way to visualize/create filtering pipelines: Pipelines that will process one channel only (Single / Individual Processing) Pipelines that will process multiple channels of multiple variables (Global Processing)

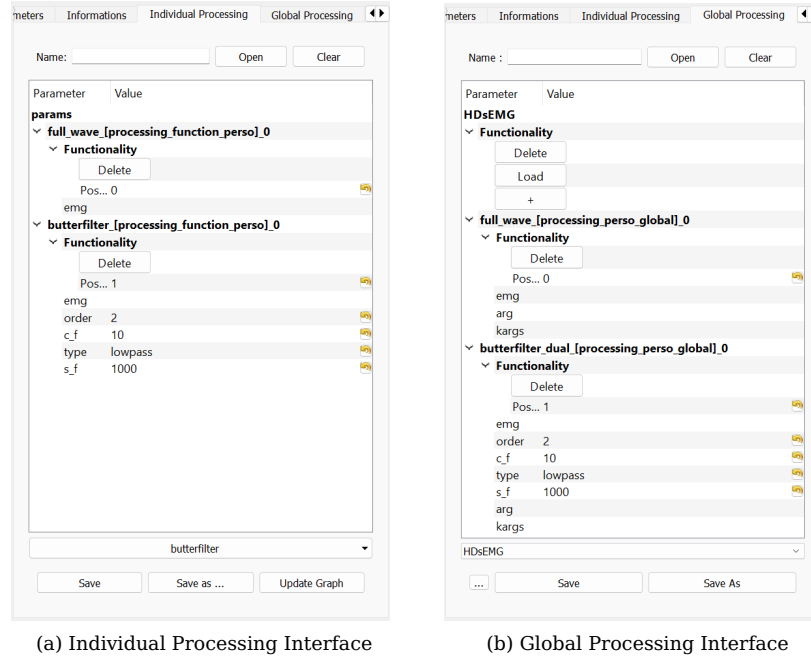


Figure 7: Processing Pipelines

A key difference between Global Processing and Single Processing is that the former is dependent on the file structure as it stores the names of the variables on which the filters are applied whereas a Single Processing is independent from any variables or channels and can be applied to any array of data.

As illustrated, a Single Processing is thus a list of filters and Global Processing is a list of filters attributed to specific variables.

A second key difference is that Single Processings are only used for pre-visualization and are selected at a graph level, in the tab parameters of a PlotLine for instance: different Single Processing can be selected on different PlotLine Plots. On the other hand, a Global Processing is selected at the file level, they will automatically filter the data and affect any visualization.

A Single Processing comes on top of a Global Processing.

A Single Processing is not taken into account while saving signals or generating summary whereas a Global Processing is taken into account.

Example of JSON

Single Processing JSON

```

1 {
2   "0": {"name": "full_wave", "process name": "full_wave_[processing_function_perso]_0", "path": ["processing_function_perso", "full_wave"], "arguments": {"emg": null}},
3   "1": {"name": "butterfilter", "process name": "butterfilter_[processing_function_perso]_0", "path": ["processing_function_perso", "butterfilter"], "arguments": {"emg": null, "order": 2, "c_f": 2, "type": "lowpass", "s_f": 1000}}
4 }
```

Global Processing JSON

```

1 {"HDsEMG":
2   {"0": {"name": "full_wave", "process name": "full_wave_[
      processing_perso_global]_0 ", "path": ["processing_perso_global", "
      full_wave"], "arguments": {"emg": null, "arg": null, "kargs": null}}, "1":
      {"name": "butterfilter_dual", "process name": "butterfilter_dual_[
      processing_perso_global]_0 ", "path": ["processing_perso_global", "
      butterfilter_dual"], "arguments": {"emg": null, "order": 2, "c_f": 10, "
      type": "lowpass", "s_f": 1000, "arg": null, "kargs": null}}},
3 "Accelerations":
4   {"0": {"name": "full_wave", "process name": "full_wave_[
      processing_perso_global]_0 ", "path": ["processing_perso_global", "
      full_wave"], "arguments": {"emg": null, "arg": null, "kargs": null}}}
5 }

```

2.4 Saving of Processing Signals and Summary Generation

Summary Window as 2 parts for:

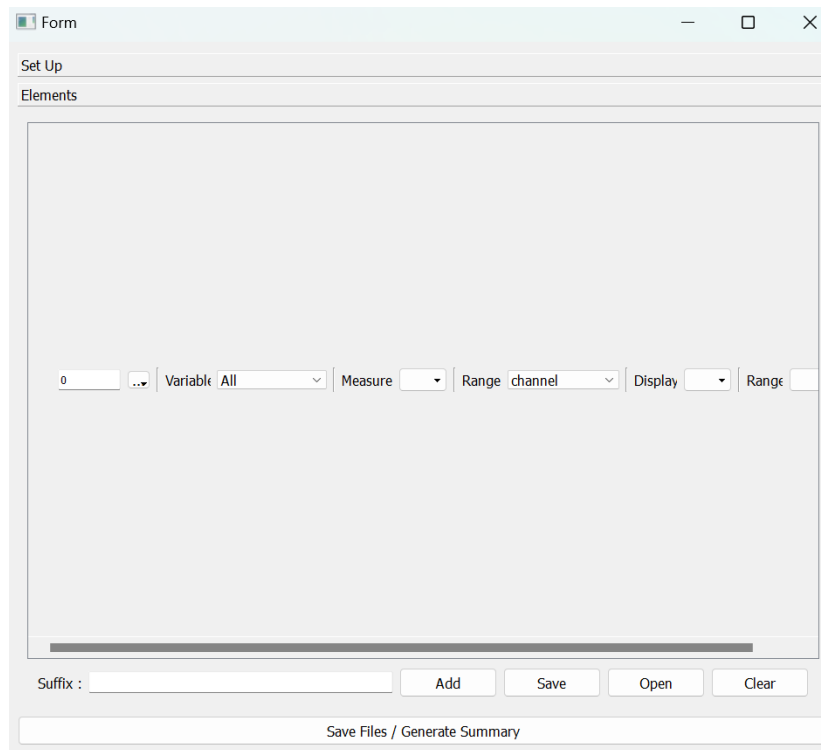
1. Set up
2. Creation of the summary (if any)

One can save the filtered signals without creating a summary or create summaries without saving the filtered signals. The processed signals can be saved in copies of the original files where a new variable is added to store the processed signals. Summaries can also be generated, displaying various metrics. One summary is created for each file.

Figure 8: Summary Set up

2.4.1 Set Up

1. **Processing** : Selection of the Global Processing that will be applied to the file. By default the current Global Processing is selected.
2. **Batch** : Selection of the files to filter and/or analyze. By default the currently loaded files are used.
3. **Save in** : Selection of the suffix of the saved file with the filtered signals. The name of the saved files are their original name, plus the suffix, plus a number starting at 0 to ensure that no files are overwritten during saving. The selected folder is also the one where the summaries will be saved
4. **Save the processed signals** : If checked, the processed signals will be saved in new variables in copies of the original files.



The screenshot shows a software window titled 'Form' with a 'Set Up' tab selected. Below the tab is an 'Elements' section which is currently empty. Underneath this, there is a row of controls: a text box containing '0', a dropdown menu, a label 'Variable:' followed by a dropdown menu showing 'All', a label 'Measure:' followed by a dropdown menu, a label 'Range:' followed by a dropdown menu showing 'channel', a label 'Display:' followed by a dropdown menu, and a label 'Range:' followed by a text box. At the bottom of the window, there is a 'Suffix:' label followed by a text box, and four buttons: 'Add', 'Save', 'Open', and 'Clear'. A wide button labeled 'Save Files / Generate Summary' spans the bottom of the window.

Figure 9: Summary Elements

2.4.2 Summary

The summary is composed of

1. a header that summarizes the instructions given to generate the summary
2. the chosen metrics
3. the metadata of the file are shown.

Summary structure can be saved as a JSON file.

Choice of metrics

0 ... Variable HDsEMG Measure mean Range channel Display heatmap8 Range forAllGroup

Figure 10: Summary Elements Details

An element of the summary is parametrized by

1. **Position** : Its position in the summary
2. **Variables** : The variable(s) across which the metric is computed
3. **Measure** : The metrics, eg: mean, rms ...
4. **Range** : metrics are computed for each channel of the variable, it is then possible to compute an average of these metric
 - channel: no change
 - variable: mean of all channels for each groups
 - variable-general: mean of all channels across all groups
 - groups: means of all channels across all variables and all groups. It is equivalent to 'variable-general' when only one variable is chosen
5. **Display** : the way the metrics are displayed, eg: heatmap
6. **Range** : indicated to the interface what data should be passed to the display functions, eg: each channels individually or a all channels of a groups. For instance: to display a heatmap, all the channel of the group should be passed as argument of the function heatmap hence range = 'forAllGroup'