

GAS SENSOR DATA ANALYSIS SYSTEM

DOCUMENTATION

Version 0.9 (beta test)

TABLE OF CONTENTS

Constructor	1
startMenu	2
startMainLayout	3
openFileDialog.....	4
openVisualizationDialog	5
openNormalizationDialog	6
openResponseDialog.....	7
openExportDataDialog.....	8
openSettingsDialog	9
openFileRoutine	10
previewData	10
setVisualizationDF	10
setNormalizationDF.....	11
calcResponse	11
appendResponseToDF.....	12
clearResponseDF	12
powerLawFunc	12
fitRespData	12
getExportFileDirectory	13
setSettings	13
plotDataFrame	13
plotPreviewDF	13
plotVisualizationData.....	13
plotNormalizationData	13
plotResponseData	13
plotFittedData	14
exportData.....	14
showAboutDialog.....	14
invalidParametersWarning	14
closeEvent.....	14

Constructor

The constructor creates the main variables for the analysis process and settings. It also calls the two functions that will build the menu and the main layout. Below there is a list and a small explanation of each variable.

matplotlib.style.use('seaborn') > set this style as soon as the software is running

matplotlib.use('Qt5Agg') > This is the backend for rendering matplotlib figures in Qt5

Now the system creates the six main data frames that will be filled according to the analysis process:

DATA FRAMES:

rawDF > first DF that will hold data no matter what. It is used to verify if the data can be converted into a table with at least two columns (time and sample signal).

previewDF > After filling the rawDF, the preview DF will get its value and do the conversions according to the user input on the openInputDialog function

visualizationDF > this is the DF that will show the time interval of interested

normalizationDF > This DF is composed of the normalized data

propertiesDF > this will hold the RESPONSE, RESPONSE TIME, and RECOVERY TIME for each channel

fitDF > this holds the fitted DATA

propertiesDF.index.name = 'Concentration' > This sets the index column name to concentration of the properties DF

VARIABLES:

fileName > variable that will hold the file path + name that the user chose to open

concentrationValues > this is a list that will hold the concentration values according to the user's input

separatorList = ['\t', ',', ' ', ';'] > This list holds the possible characters for column separator

separator = "" > this holds the character choosen. It is empty at the start

responseLabel = u'\u0394R/RO (%)' > this holds the response string that will be used in graphs and data tables. It initially writes $\Delta R/RO$ (%)

for visualization:

startVisualizationTime > variable that will set the initial time for the visualization DF

endVisualizationTime > variable that will set the initial time for the visualization DF

startVisualizationTime from the index column (time data) of the **visualizationDF**

for properties:

startExposureTime > variable that holds the start of exposure time

endExposureTime > variable that holds the end of exposure time

endRecoveryTime > variable that holds the end of recovery time

for fitting:

x_fit_values = [] > Holds a list of the concentration values calculated after fit

coef1_list = [] > Holds the list of coefficient “a” in the power law for each channel

coef2_list = [] > Holds the list of coefficient “b” in the power law for each channel

fitListLabel = [] > Holds the list of Labels that will contain the information about each coefficient

numberOfFitPoints = 100 > this is the number of fit points. Default setting is 100

Color settings are based on the lists below

colorsList1 = ['black', 'firebrick', 'orange', 'olive', 'yellowgreen', 'seagreen', 'skyblue', 'violet']

colorsList2 = ['k', 'b', 'g', 'r', 'c', 'm', 'y', 'cyan']

colorsList3 = ['black', 'lightcoral', 'chocolate', 'gold', 'limegreen', 'royalblue', 'indigo', 'crimson']

DICTIONARIES

responseType = {'dR/R0': True, 'Rgas/Rair': False} > this Dictionary holds the control for possible response types

showingChannelsControl = {'ch1': False, 'ch2': False, 'ch3': False, 'ch4': False, 'ch5': False, 'ch6': False, 'ch7': False, 'ch8': False} > this dictionary is used to check what channels the user wants to analyse

plottingControl = {'previewDF': False, 'visualizationDF': False, 'normalizationDF': False, 'propertiesDF': False, 'fittingDF': False} > this dictionary is used to control what is the data being plotted. It is used in the settings dialog box to change styles and color and plot the same data as before.

colorDic = {'Palette1': self.colorsList1, 'Palette2': self.colorsList2, 'Palette3': self.colorsList3} >

This dictionary holds the color options

palette = colorDic['Palette1'] > This variable sets the first colors list as the default color pallet

Call functions:

startMenu() > This function builds the menu

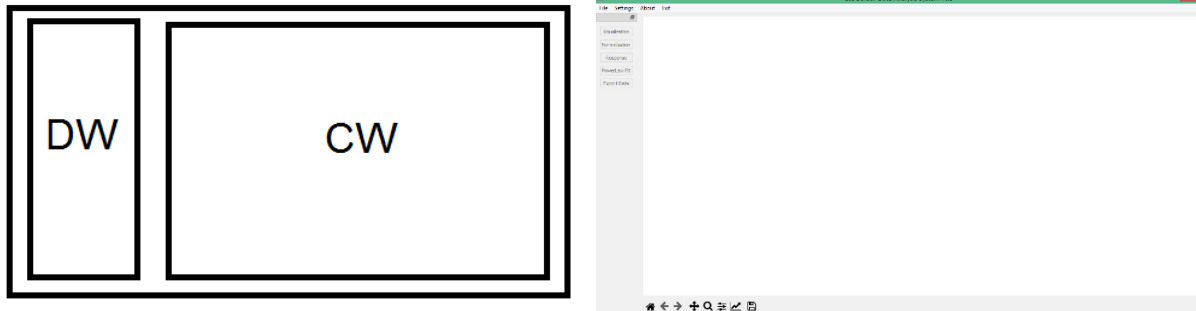
startMainLayout() > This function builds the layout

startMenu

This function creates the main menu. The menu has File, Settings, About, and Exit options.

startMainLayout

This function builds the main layout of the software. The main layout is composed of a centralWidget and a dockWidget. The main layout is a HBox, the scheme is shown below:



cw > is the central widget

mainFigure > is the object inserted in the CW. This object is a matplotlib Figure. The commands after it are set the menu bar and backend canvas for pyqt

dw > is the dock widget

the DW has the following buttons:

visualizationBtn > To access the Visualization Dialog Box

normalizationBtn > To access the Normalization Dialog Box

responseBtn > To access the response Dialog Box

fitBtn > To fit the response data

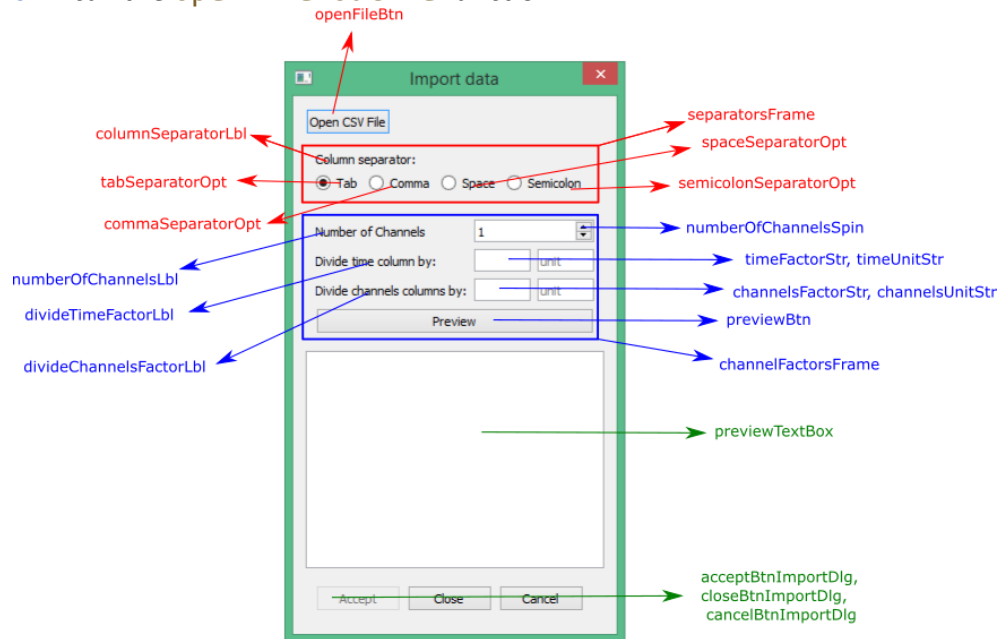
exportBtn > To access the export data Dialog Box

openFileDialog

This function will construct the dialog to open/import data. Here, the user has to browse for the file to open, define what is the separator used in the data table, define the number of channels, being 8 the maximum value. Also, there is the possibility of dividing the time or the channel columns by a factor to convert the time/resistance to the desired unit.

importDlg > Dialog

openFileBtn > call the **openFileRoutine** function



The main layout of this Dialog Box is a VBox that contain the following elements:

separatorsFrame > holds the **columnSeparatorLbl**, **tabSeparatorOpt**, **commaSeparatorOpt**, **spaceSeparatorOpt**, **semicolonSeparatorOpt**. This option chosen here will be used in the **previewData** function.

channelFactorsFrame > holds the **numberOfChannelsLbl**, **numberOfChannelsSpin**, **divideTimeFactorLbl**, **timeFactorStr**, **timeUnitStr**, **divideChannelsFactorLbl**, **channelsFactorStr**, **channelsUnitStr**, **previewBtn**. These values will be used in the **previewData** function

previewTextBox > this box is filled with the previewDF after setting it.

acceptBtnImportDlg > calls the function **plotPreviewDF**

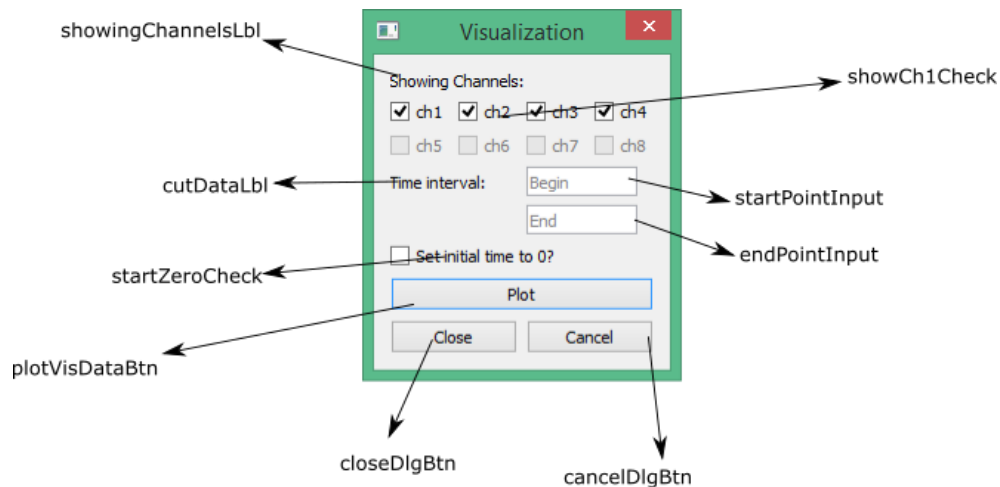
closeBtnImportDlg > accepts and closes the dialog

cancelBtnImportDlg > cancel any alteration

openVisualizationDialog

This is function builds the dialog in which the user defines the parameters for data visualization. It may be of interest to analyze only a defined number of channels or just a region. Also, for convenience, there is an option to set the initial time to zero.

importDlg > This is the main dialog



This box's layout is a grid with four columns, it contains the **showingChannelsLbl**, **showCh1Check**, (and this repeats to 8) **cutDataLbl**, **startPointInput**, **endPointInput**, **startZeroCheck**, **plotVisDataBtn**, **closeDlgBtn**, **cancelDlgBtn**

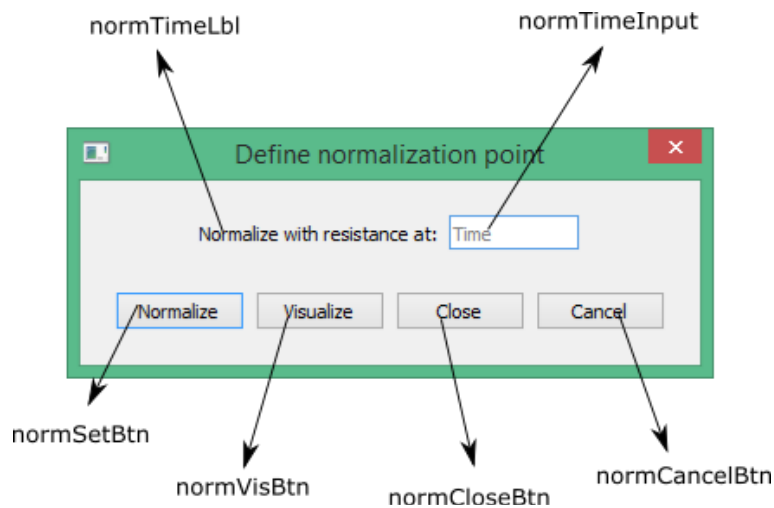
The **plotVisDataBtn** calls the **setVisualizationDF** function.

The checkboxes are linked to **showingChannelsControl** dictionary. This dictionary is used after the user clicks the Accept button in the **openFileDialog** linked to the function **plotPreviewDF**. Here, the algorithm checks the number of columns used and set each key of this dictionary to true or false.

openNormalizationDialog

This function opens the Normalization dialog. Normalization means that the system will divide each column by its own value at a given time. The resulting data table will be plotted together for comparison. In this box, the user can choose the normalization point, and it can go back to the visualization table. This option should be available only when there are more than 2 columns in the visualizationDF

normalizationDlg > This is the main dialog box. It contains the **normTimeLbl**, **normTimeInput**, **normSetBtn**, **normVisBtn**, **normCloseBtn**



The button **normSetBtn** will call the function **setNormalizationDF** to define the normalizationDF while the **normVisBtn** button will call **plotVisualizationData** to plot the visualizationDF that has already been set at this point of the analysis.

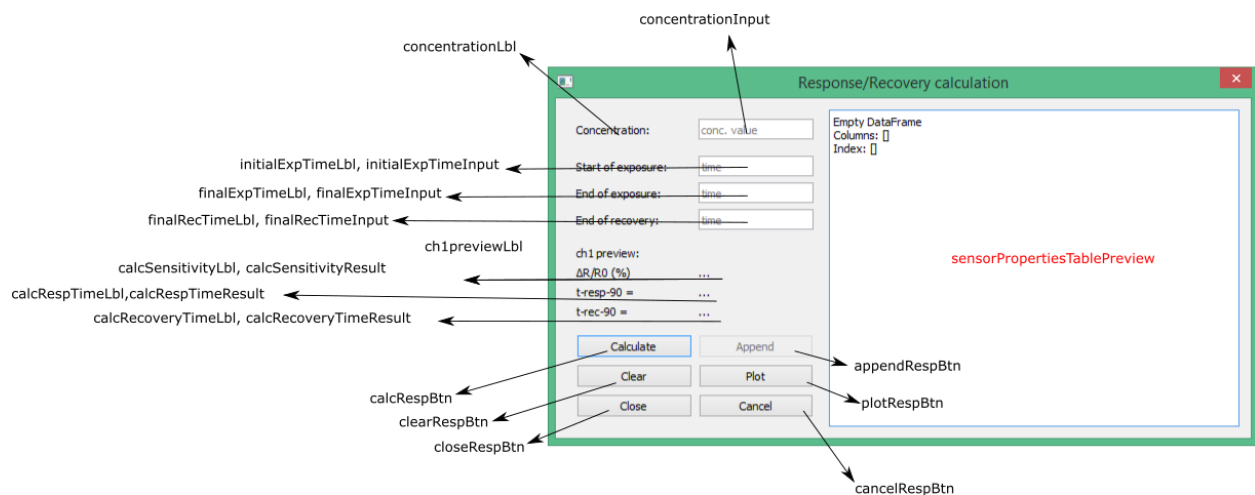
The layout is a Vbox with the **topNormDialogWidget** and **BottomNormDialogWidget** objects. It was designed this way to align both of these objects in the center.

openResponseDialog

This is the box to calculate the response, response time and recovery time of each exposure. The user has to input the concentration of a given exposure cycle, the start of exposure time, the end of the exposure time, and the time of the end of recovery. Based on these three values, the system will calculate the response in dR/R0 in percentage or Rgas/Rair according to the user's choice. It will also calculate the response time and recovery time will be given in time units

`responseDlg` > is the main dialog box and its layout is a Hbox that contains the `leftPanel` and the `sensorPropertiesTablePreview`.

The left panel contains the `concentrationLbl`, `concentrationInput`, `initialExpTimeLbl`, `initialExpTimeInput`, `finalExpTimeLbl`, `finalExpTimeInput`, `finalRecTimeLbl`, `finalRecTimeInput`, `ch1previewLbl`, `calcSensitivityLbl`, `calcSensitivityResult`, `calcRespTimeLbl`, `calcRespTimeResult`, `calcRecoveryTimeLbl`, `calcRecoveryTimeResult`, `calcRespBtn`, `appendRespBtn`, `clearRespBtn`, `plotRespBtn`, `closeRespBtn`, and `cancelRespBtn`.



The procedure here should be entering the values in the four input boxes (`concentrationInput`, `initialExpTimeInput`, `finalExpTimeInput`, `finalRecTimeInput`), hit the calculate button and then append to insert the data in a table that will be shown in the left white area. The clear button will delete the last datapoint, in case of any mistakes.

The `calcRespBtn` is linked to the `calcResponse`.

The `appendRespBtn` is linked to the `appendResponseToDF`

The `clearRespBtn` is linked to the `clearResponseDF`

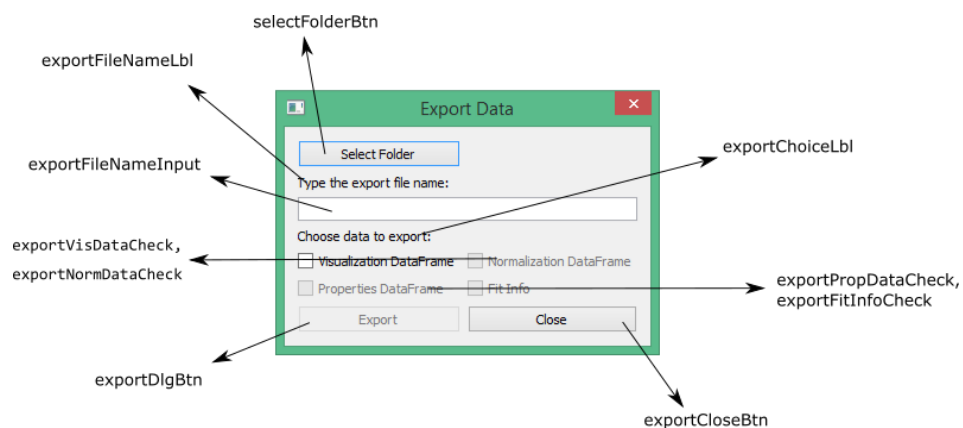
The `plotRespBtn` is linked to the `plotResponseData`

openExportDataDialog

This function opens the export Data dialog.

The user can choose to export to CSV files the data from each one of the DataFrames as long as they are holding data. The options to export are visualizationDF, normalizationDF, propertiesDF, fit Info.

exportDlg > This is the main box. It contains the **selectFolderBtn**, **exportFileNameLbl**, **exportFileNameInput**, **exportChoiceLbl**, **exportVisDataCheck**, **exportNormDataCheck**, **exportPropDataCheck**, and **exportFitInfoCheck**



The system will keep the checkBox disabled if there is no data in the data frames to export. It does it by checking the length of the index row of the DataFrames, if it is zero, then it disables the checkboxes.

The export files will be CSV type created with the same column separator used in the openFileDialogBox and it will hold the name of the columns.

openSettingsDialog

This box allows the user to choose the response type, number of fitting points, plotting style, and color palette.

The most important settings here are the response type and number of fitting points. The number of fitting points will be the parameter that will generate the fitted function. If the number is too big, it can make the fitting process slow.

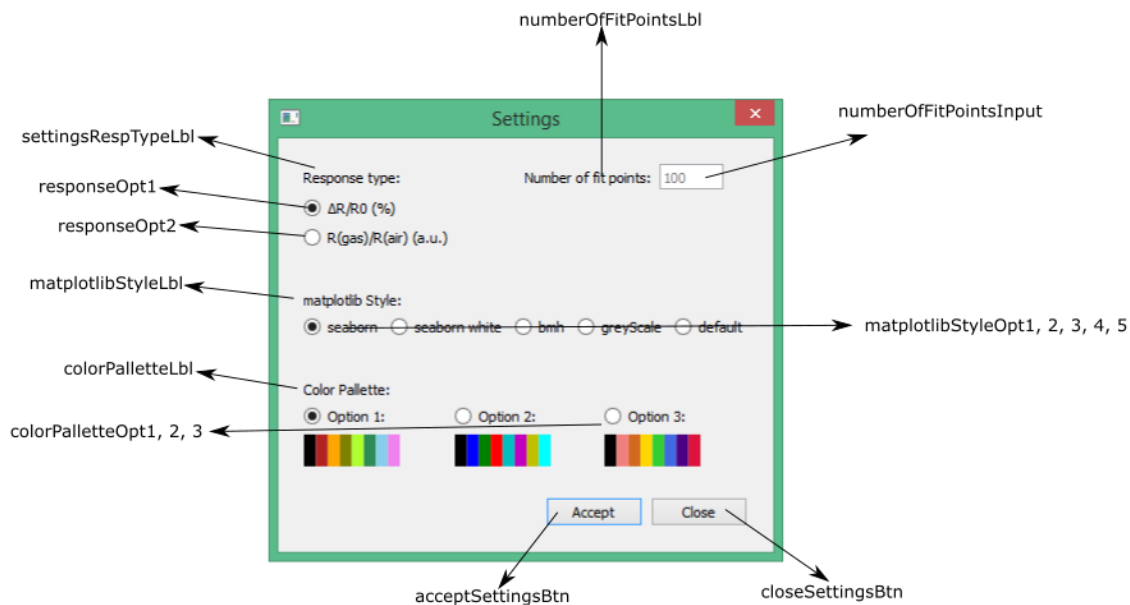
settingsDlg > This is the main box. The layout here is a VBox that contains four lines (**settingsDlgWidget1**, **settingsDlgWidget2**, **settingsDlgWidget3**, **settingsDlgWidget4**).

The first contains **settingsRespTypeLbl**, **responseOpt1**, **responseOpt2**, **numberOfFitPointsLbl**, **numberOfFitPointsInput**.

The second contains, **matplotlibStyleLbl**, and **matplotlibStyleOpt1** to **matplotlibStyleOpt5**

The third contains, **colorPaletteLbl**, and **colorPaletteOpt1** to **colorPaletteOpt3**

The fourth contains **acceptSettingsBtn** and **closeSettingsBtn**



openFileRoutine

This function will get the file path using the QFileDialog and set the fileName that will be used to insert the data into the rawDF

previewData

This function is activated when the preview button (**previewBtn**) in the open file dialog box is clicked. This function will run the following steps:

1. Check if there is a filePath in the variable **fileName**. If there is, it goes forward, else it does nothing;
2. The algorithm then empty the **rawDF** and **previewDF**. This is designed to make this function able to work multiple times after the software is running;
3. It checks what is the separator chosen and uses the **separatorList** to assign it to the **separator** variable;
4. Creates the rawDF using the read_csv from pandas;
5. If it does not have at least 2 columns, the separator is wrong;
6. Builds the previewDF by putting the name of ch1, ch2, ch3... up to the length of the number of columns in the rawDF. Also calculates and the new time values and channel values based on the users input. If these inputs are not floatable, it will return an error.
7. By the end of the routine it will put the data in the **previewTextBox** and enable the acceptButton and the visualization button in the dock widget;
8. The name of each column here is ch1, ch2... For each of these columns, it will make each variable stored in the dictionary **showingChannelsControl** True;

setVisualizationDF

This function is called by the plot button (**plotVisDataBtn**) in the visualization dialog box. It will run the following steps.

1. Empty the **visualizationDF**;
2. Check if the user has entered values to start and end time. If so, it will assign the variable **startVisualizationTime** and **endVisualizationTime** to the closest value that the user has entered. If there is none, it will get the first and/or last values from the **previewDF**;
3. Then, it makes the **visualizationDF** equals the interval between **startVisualizationTime** and **endVisualizationTime** of the **previewDF**;
4. It checks if the user wants to set the time to zero. If so, it will subtract the **startVisualizationTime** of the time data in the **visualizationDF**;
5. Creates and populate a list (**showingChannelsList**) with the channels selected. Each of the checkbox was already made enabled or disabled after values present in the **showingChannelsControl** that were set in step #8 of **previewData** function.

6. If the users do not select at least one channel, it returns an error, else, it makes the visualizationDF equals itself but with only these columns, make the buttons response, and export available and call `plotVisualizationData`.
7. Finally, it creates the `propertiesTableColNames`. This is based on what columns are inside the visualizationDF.

setNormalizationDF

This function will normalize the data inside the visualizationDF as long as it has at least two columns by dividing each column by its own value at the chosen time. It runs the following steps:

1. Empty the `normalizationDF`;
2. If there is text in the `normTimeInput`, it makes the `normalizationPoint` as the closest value from the user's input.
3. It makes the normalizationDF indexes the same as the visualizationDF;
4. Then it inserts the visualization data divided by the `normalizationPoint`;
5. Calls `plotNormalizationData`;

calcResponse

This function calculates the response, response Time and the recovery time of a given exposure-recovery cycle for all channels present in the `visualizationDF`. This function is called by pressing the `calcRespBtn` on the `responseDlg`. The system does that by running the following steps:

1. After entering the concentration, the start of exposure time, end of exposure time, and end of recovery time, it will look into the `visualizationDF` for the closest values present in the data table. It will assign the variables `startExposureTime`, `endExposureTime`, `endRecoveryTime` with these values;
2. Creates a list called properties list. For each column in the `visualizationDF` it will find the initial resistance `r0`, final exposure resistance `rf`, and final recovery resistance `rf2`. These values correspond to the resistance values of `startExposureTime`, `endExposureTime`, `endRecoveryTime`, respectively.
3. Calculates the absolute variations in the adsorption and desorption cycle;
4. It calculates the response according to the user's settings (in the settings dialog). The default is $dR/R0$ in %.
5. Calculate the values corresponding to 90% response and recovery variation, and finds the corresponding values in the visualizationDF. The routine is different if the resistance variation is positive or negative. The variation depends on the nature of the semiconductor material/gas interaction;
6. Find the time in the time table corresponding to these values, and subtracts the `startExposureTime`, `endExposureTime` to calculate the response time and recovery time, respectively;

7. Append the values of response, response time and recovery time to the properties list, set the append button enabled,
8. Update the channel 1 preview panel and if the propertiesDF has at least three rows, it opens the enables the button in the dock widget for the power-law fit.

appendResponseToDF

This function gets the [propertiesList](#) and added to the [propertiesDF](#). It does that by #1 converting this list into a pandas' series with the [propertiesTableColNames](#) and the name being the concentration value. #2 This series becomes the row is then appended to the [propertiesDF](#) and #3 it updates the text shown in the [sensorPropertiesTablePreview](#)

clearResponseDF

The user can clear the last line of the [propertiesDF](#) to start over;

powerLawFunc

Function used in the fitting has the form of

$$y = ax^b$$

fitRespData

This function will fit the response data to the [powerLawFunc](#) described previously. It does it by using the [curve_fit](#) from the [scipy.optimize](#). The algorithm run as follows:

1. Empty the [fitDF](#);
2. Calculate the fitting step by dividing the last data point from the [propertiesDF](#) concentration column by the [numberOfFitPoints](#) that can be entered in the settings menu. If the last concentration is 5, and the number of points is 100, each step will be of 0.05;
3. Enter the values in the [x_fit_values](#) that represent the concentration values according to the step value, and starting at zero;
4. Insert these values in the [fitDF](#);
5. For each column in the [propertiesDF](#) it will fit the curve using the function [curve_fit](#) and it uses the [visualizationDF.columns](#) to get the proper names of each column. This is important because the user can choose any sort of combination of channels to fit;
6. The [curve_fit](#) function returns the coefficients that are added to the [coef1_list](#) and [coef2_list](#). Each of these lists will hold a number of values equivalent to the number of columns analyzed.
7. Creates a str with this information that will be used in the legend when plotting. These strings are stored in the variable [fitListLabel](#)
8. It calculates the [y_fit_values](#) and add to the [fitDF](#);

9. Call the function `plotFittedData` that plots both the `fitDF` and the response data from `propertiesDF`;

`getExportFileDirectory`

This function creates the string to the path to export the data

`setSettings`

This function will set the values chosen in the settings dialog box.

1. sets type of response type by assigning the to `responseType` dic
2. sets the fitting number points
3. sets the matplotlib style
4. sets color pallet
5. plot with new settings

`plotDataFrame`

This is a generic plot function that was designed to plot a dataframe. It also asks for the units of the axis and it puts the name of the axis as necessary. It creates up to eight plots separated.

`plotPreviewDF`

This function first adjusts the plotting control to the `previewDW` and then it plots it using the `plotDataFrame` function

`plotVisualizationData`

This function first adjusts the plotting control to the `visualizationDF` and then it plots it using the `plotDataFrame` function

`plotNormalizationData`

This function plots the `normalizedDF` in one axis

`plotResponseData`

This function will plot the response, response time and `recoveryTime` in three plots separated

plotFittedData

This function will plot the response and the fitted data in one plot

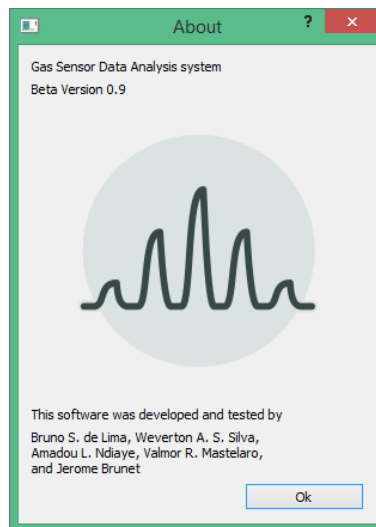
exportData

This function will export the data frames into CSV files according to the folder that the user has chosen

1. Gets the export file names from the input box
2. Create paths according to the [exportDirectory](#) from the [getExportFileDirectory](#) function
3. Export the data that is checked. The check boxes are disabled if there is no data in the respective dataframe. The [fitDF](#) generates two tables, one with the data, another with the fit info

showAboutDialog

Opens up the following About Dialog



invalidParametersWarning

Warning box for error handling.

closeEvent

Close event with a message to confirm