Data Analysis Procedure

1. Proper Python environment setup is required to run the analysis modules (step by step procedure will be added here later to help new users, different for Windows and Unix systems).
2. To start the analysis, copy all the files from GitHub repository <https://github.com/netepenko/Analysis_GIT> to your computer and place them in one common folder.
3. Run the analysis\_GUI.py file which will start the users interface for data analysis
4. Enter Shot and Channel number and press Load data button. If there was a problem with loading the data you will see a notification in the status bar, if the data was loaded successfully the raw data file name will be displayed there.

Data loading routine creates an object of channel\_data class, defined in channel\_data\_class.py file. The instance of that class has embedded functions like plotting

Channel data class initialization routine reads the database file MainDB.db to get the file location for selected shot and other parameters. The database file should be in the same folder with the python scripts. The script which implements the database reading is database\_operations.py. To view the database one can use free database viewer from <http://sqlitebrowser.org/>.

If the raw data file for the specified shot exists in the folder mentioned in the database Shot\_List table, the initialization routine reads raw data from the file and stores it for further use.

To view the file structure of the raw data \*.hws file one can use the HDF viewer available at <https://support.hdfgroup.org/products/java/release/download.html>.

1. To plot the loaded data press Plot data button, it will open a plot frame with the raw data points displayed. One can navigate by zooming in and dragging through the data to specific region of interest. It is also possible to change the number of points displayed on the figure at the right upper corner of the frame and hit refresh button. This feature is useful to prevent plot from freezing when trying to plot a big data set (hundred million of points), especially on the slow machines. The modified plotting routine skips some data point and leaves only as many of them not to exceed the specified limit. At the same time when zooming is done the routine reloads more points from the data, which allows to see the details, which cannot be done with the standard python matplotlib plotting.
2. To perform the fitting of the signal we need to identify the regular peak shape of the pulses. The normalized pulse shape can be described with the following equation:

latex-image-1.pdf

where c1 and c2 are decay and rise constants, and V0 and t0 are normalization constant and time shift to have peak at t=0 which are determined by c1 and c2.

The good peak finding algorithm requires a voltage threshold to be specified and chi2 fitting parameter limit when the peak considered to be of a good shape. Click Find good peaks button and check the results of the peak shape fitting. This should be done before adding a pulser to tha data, since it will pick pulser peaks instead of the real peaks. The selected good peaks figures are place in the folder ../Analysis\_Results/\_shot\_number\_/Good\_peaks/. Check if the selected peaks are good and if not reduce chi2 and repeat the step. Function used for good peak search and shape determination is defined in peak\_sampling.py module and called peak\_shape. This function automatically writes found peak shape parameters to the database table Peak\_Sampling, such as decay time, rise time and position of maxima.

1. Once the peak shape for the channel data was found, the fitting of the data can be done using the linear fitting algorithm with polynomial background. The fitting routine is defined in raw\_fitting.py and uses lfitm1.f90 fortran code which should be compiled to python module.

Input the fitting interval and press Fit interval button.

The output is placed in ../Analysis\_Results/\_shot\_number\_/Raw\_fitting folder.