BnmrOffice Documentation

Hassan Saadaoui
TRIUFM, 4004 Wesbrook mall, Vancouver, BC V6T 1Z4
saadaoui@triumf.ca

April 17, 2015

Abstract

This document gives a description and tutorial of the BnmrOffice program. The graphical user interface (GUI) of the program is designed using Qt technology. Several well-tested packages are needed to perform necessary tasks such as minimization, reading and plotting data, and scientific calculations. Instructions about installation, structure of the code, and models of the program will be covered.

Contents

1 Introduction

`BnmrOffice` is used to search, view, and analyze ASCII and β-NMR (.msr) data. It can be extended by the user to read any type of data. The program does many other tasks such as simulations, database interface, and converting units. This program is developed by Hassan Saadaoui and is maintained as needed. It is open source and released under the General Public License (GPL). No warranty or guarantee of the results is implied. Please acknowledge the author if you are using this program in an offline data analysis. It is the least you can do to encourage future developments and maintenance of the program. For any questions, please email at saadaoui@triumf.ca.

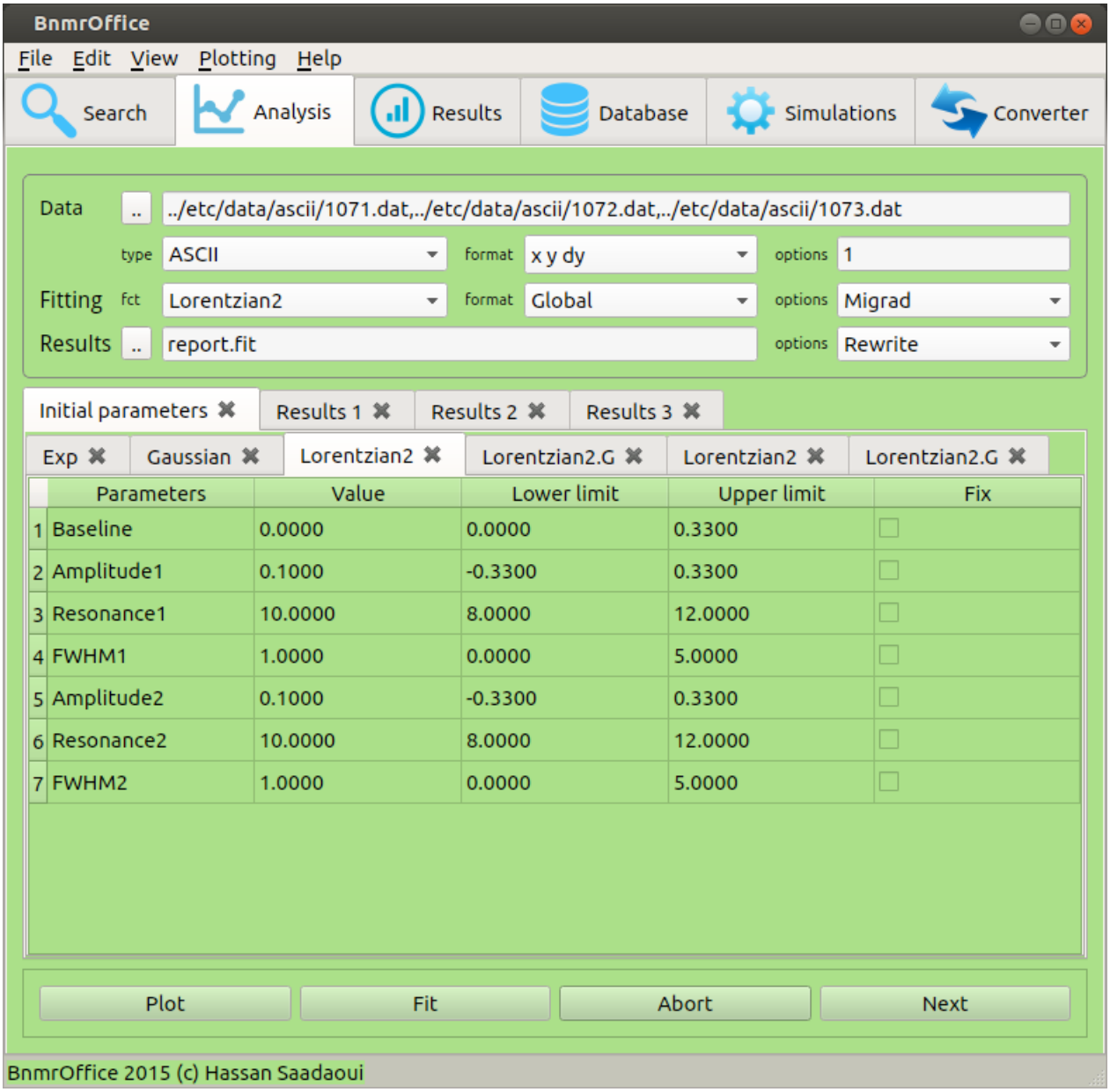

Figure 1: Regression page of the `BnmrOffice` graphical user interface.

2 Requirements

The program is developed in `C++` and `QT`. The latter provides many excellent libraries for programming and building the graphical user interface. Many desktops of Linux OS are build using `QT`, such as the KDE desktop. `QT` is cross-platform, modern and well-maintained. The `BnmrOffice`'s core components are as shown in Fig. 2

- `QT`: a `C++` framework for programming and developing GUI applications. Version 4.8.x or 5.x is needed.
- `MUD`: a library to read the μSR data format (.msr) developed at `CMMS` in TRIUMF.
- `MINUIT`: a minimization routine for fitting data.
- `XMGR`: an application to plot and visualize data.
- `QCustomPlot`: to plot data directly onto the GUI (included within the package).
- `GSL`: (optional) for compiling few fitting functions.

In addition to the above requirements, and depending on your system, you may also need some dependency packages. The main packages are:

- `gcc` compiler
- `automake`

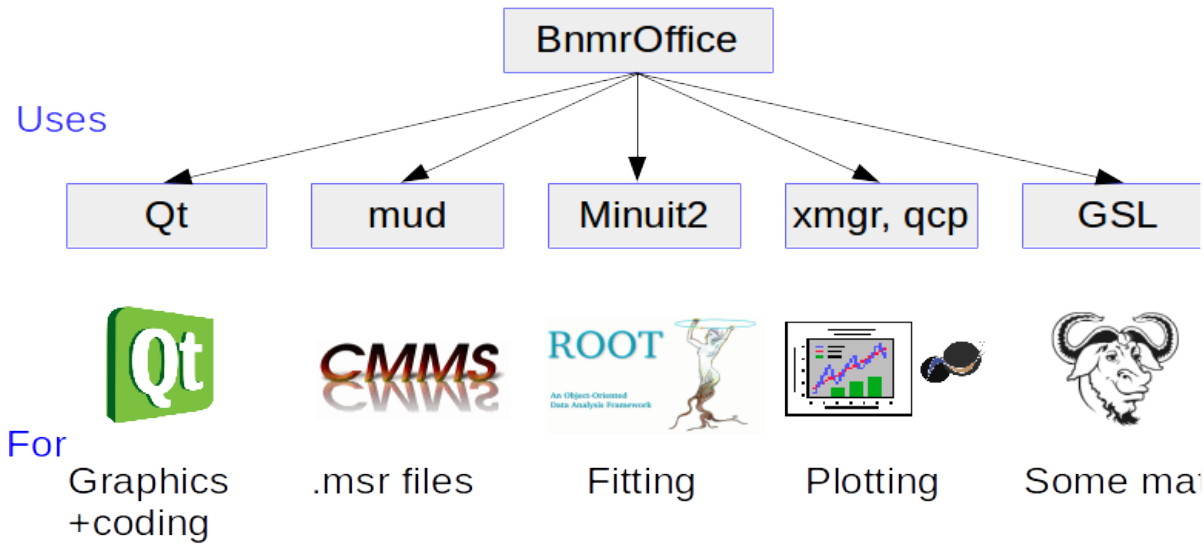

Figure 2: External packages required by `BnmrOffice`.

3 Package Structure

The package you download will have a structure as shown in Fig. 3. The main folder contains 4 sub-folders and 5 files.

- `src/`: contains the source code. This has also many sub-folders for each page and a `main.cpp`, `mainwindow.(cpp,ui,h)`, `plotting.(cpp,h)`, and `constants.h` for some shared constants.
- `fct/` contains the fitting functions and script `./compile` to execute the codes and create the libraries.
- `etc/` for documentation, images, data, scripts and templates, and the resources.qrc file needed by `QT`.
- `bin/` where the execution binary data is dumped.
- `bnmroffice.pro` used to generate the makefile.
- `AUTHOR` for authorship attributions.
- `LOG` for keeping a log of the package changes over time.
- `COPYING` supplies the GPL agreement.
- `README` for installation instructions.

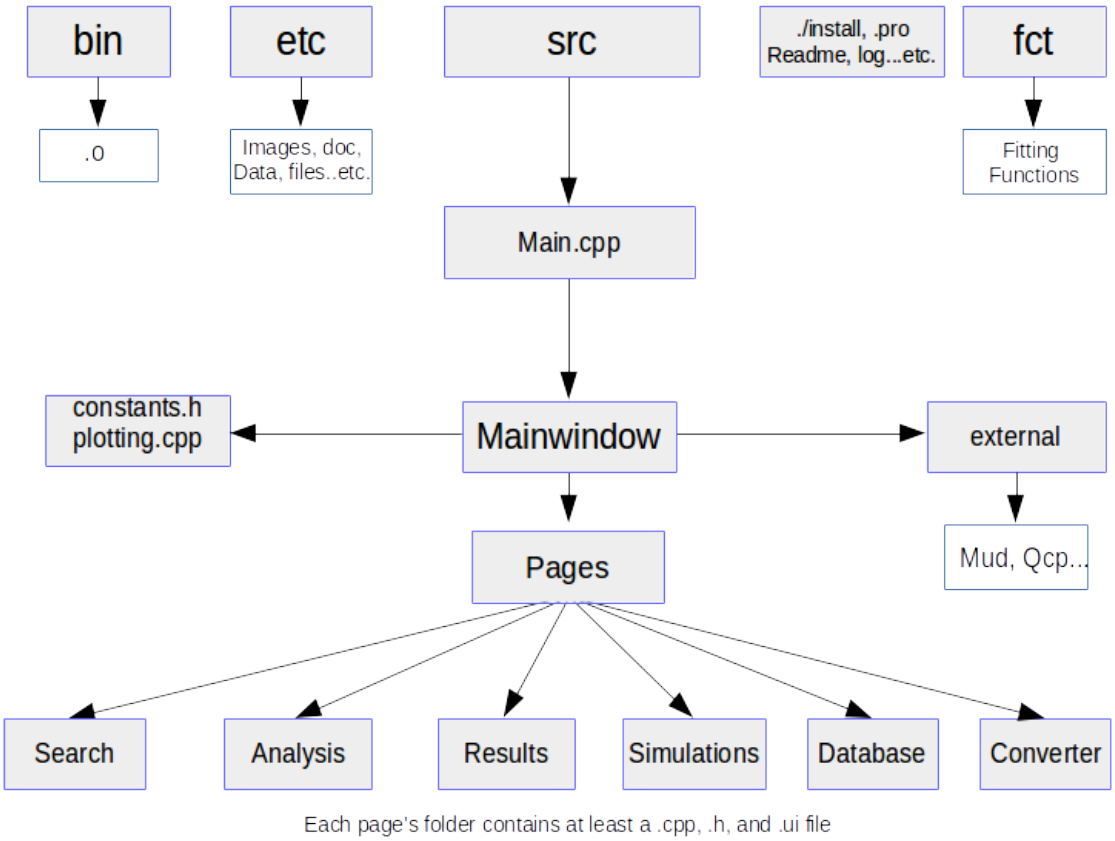

Figure 3: Structure of the program. Upon execution, `BnmrOffice` calls main which calls mainwindow, plotting, and constants. The mainwindow in turn calls pages and menu bar options. In each page, you find three files; .cpp, and .h for the programming part, and .ui for the GUI part.

4 Installation

In addition to bnmroffice, you also need: `QT`, `MINUIT`, `MUD`, `XMGR`, and `GSL`.

1. It is likely that your system has `QT`, and `XMGR` pre-installed. In this case, you only need to install `MINUIT` and `MUD` libraries.
2. These instructions may seem long, but they are meant to give as much details for the less-experienced users. In most cases, a linux-experienced user may be able to install all libraries without help, except the instructions for installing `BnmrOffice`.
3. It is assumed for clarity, in all that follows, that you unpack your downloads to the home folder `~/` (done using `-C ~/` or `-d ~/`). That is optional, as you may unpack somewhere else like; `~/programs`, `~/downloads`. Without `-C` or `-d` your unpacks will appear in your current directory.
4. Any line here preceded by the $ sign, is a command line that you may copy and paste to your terminal.
5. These instructions are meant for Linux users only. Mac users may find them useful, and Windows is not supported yet.
6. Download the latest bnmroffice from local computers, sourceforge, or github.
   `$ wget https://sourceforge.net/projects/bnmroffice/files/bnmroffice.tar.gz/download`
7. Unpack it
   `$ctar -xvf bnmroffice.tar.gz -C ~/`

4.1 QT

It is the backbone of the GUI and programming. Download the `QT` on-line installer from http://www.qt.io/download-open-source/. It is a light executable which downloads based on your system/selections. It provides all `QT` 5.x binary and source packages and latest `QT` creator.

4.2 MINUIT

This package is used for minimization. It is developed at CERN originally in Fortran and later converted to `C++`. It is very powerful and well tested. To compile, follow these steps.

1. Download latest Minuit2 located at
   http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/release/download.html
2. Unpack and cd

   ```
   $ tar -xvf minuit.tar.gz -C ~/
   $ cd ~/minuit
   ```

3. To install follow the instructions at
   http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/gettingStarted/autoconf.html
4. Make SURE that the tests in the tutorial are running as described in the link
   http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/minuit/gettingStarted/testOutput.html
5. Copy (as superuser) the miniut libraries from `minuit/src/.lib/liblcg_Minuit.*` to `/usr/lib/`

   ```
   $ sudo cp minuit/src/.lib/liblcg_Minuit.* /usr/lib/
   ```

6. Update ldconfig

   ```
   $ sudo ldconfig
   ```

Extra notes: It is somewhat a challenge to compile Minuit2. These extra notes maybe useful.

- Depending on your system, you may need to modify few codes namely src/MnUserTransformation.cpp to add `#include <cstdio>` or `#include <cstdio.h>` just below `#include <algorithm>` and re-compile.
- Locate where libraries and header files are, hopefully in `/usr/local/include/Minuit2`, and `/usr/local/lib/`
- Add the path `/usr/local/lib/` to `/etc/ld.so.conf` as described here http://stackoverflow.com/questions/1099981/why-cant-python-find-shared-objects-that-are-in-directories-in-sys-path

  ```
  $ export LD_LIBRARY_PATH=/usr/local/lib
  ```
  or
  ```
  $ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
  ```
- Run ldconfig
  ```
  $ sudo ldconfig
  ```

4.3 MUD

This package is needed to read the TRIUMF .msr files.

1. Download the MUD library source archive mud.tar.gz from http://musr.ca/mud
   ```
   $ wget http://musr.ca/mud/mud.tar.gz
   ```
2. Unpack and cd
   ```
   $ tar -zxvf mud.tar.gz -C ~/
   $ cd ~/mud
   ```
3. Run make with root access (read the install instructions withing the mud package)
   ```
   $ sudo make all
   ```
4. Copy the files mud.h and libmud.a into `/usr/lib/` and `/usr/include/`
   ```
   $ sudo cp ./lib/libmud.a /usr/lib/
   $ sudo cp ./src/mud.h /usr/include/
   ```

4.4 XMGR

This is needed for plotting. It is an old but quick and robust GUI, and makes publication quality figures. Subsequent versions have been developed, however most of the user community still prefers an outdated version dating back to 1994. Recently, a resurrected version of the outdated program appeared on Github almost 20 years later! Please contact for other options if you are unable to install `XMGR`.

1. downloand and Unpack xmgr from https://github.com/mlund/xmgr-resurrection
   ```
   $ wget https://github.com/mlund/xmgr-resurrection/archive/master.zip
   $ unzip master.zip -d ~/
   $ cd ~/xmgr-resurrection-master
   ```
2. XMGR has a feature of displaying the copyright each time it starts from a terminal. This can obscure the user from seen the error messages produced by bnmroffice. To avoid this, comment the lines 107-111 in main.c and save.
3. Typically the following packages are required: libice-dev libx11-dev lesstif2-dev libxmu-dev libxpm-dev. Install if not found in your OS. Try, on ubunto/related systems, get-all (or your system's alternative).
   ```
   $ sudo get-all install libice-dev libx11-dev lesstif2-dev libxmu-dev libxpm-dev
   ```

4. Read README.md to compile the code. Each system is different; try these given steps:
   `$ cmake . -DENABLE_NETCDF=on`
   `$ cmake . -DCMAKE_INSTALL_PREFIX=/usr/local`
   `$ make`
   `$ sudo make install`
5. If all went well, open the executable `xmgr` located likely in `src/` or somewhere else within your folder.
6. You must locate this file and copy it to `/usr/bin` and `/usr/local/bin/` if it is not already there.
   `$ sudo cp xmgr /usr/bin/`
   `$ sudo cp xmgr /usr/local/bin/`

4.5 GSL

This package is optional. It is only needed to compile few fitting functions in `fct/`

1. download latest gsl from http://gnu.mirror.iweb.com/gsl/
2. Unpack and cd
   `$ tar -xvf gsl-latest.tar.gz -C ~/`
   `$ cd ~/gsl-latest`
3. Run configure
   `$ ./configure`
4. Run make
   `$ make`
5. Run make install as root
   `$ sudo make install`
6. Update libraries cache
   `$ sudo ldconfig`

4.6 BnmrOffice

1. cd to the downloaded package
   `$ cd ~/bnmroffice`
2. change the path to bnmr Data, and bnqr Data as defined in constants.h
   `$ gedit src/constants.h`
3. If Qt binaries are not in your path, set the env (locate where qmake is)
   `$ PATH=/usr/Qt/5.4/gcc_64/bin:$PATH` (change `"/usr/Qt/5.4/gcc_64/bin"` as per your system)
   `$ export PATH`
4. Run qmake. [Optional: to modify the default install location use `$qmake PREFIX=/your.new.location/`]
   `$ qmake`
5. Run make and make install
   `$ make`
   `$ sudo make install` (needs root)
6. cd to directory `fct/` and compile all the libraries using the script compile
   `$ cd fct`
   `$ sudo ./compile`
7. To test the gui, invoke
   `$ bnmroffice` [or `$ ./bnmroffice` if not installed as root].

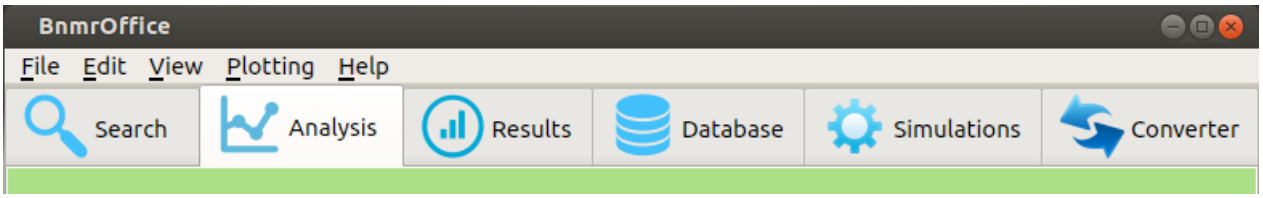

Figure 4: `BnmrOffice` menu bar and main pages.

5 Description of the GUI

The GUI has a menu bar at the very top and a tab widget below it. This tab widget contains 6 tabs (pages): `Search`, `Analysis`, `Results`, `Database`, `Simulations`, and `Converter`. Each of these contains widgets that the user may change and push buttons for issuing signals. The menu bar and the pages functionalities will be described next.

5.1 Menu bar

5.1.1 File

This has 4 options; (i) invoke a new window, (ii) open an old version of `BnmrOffice`, (iii) clean `temp.*` data which removes all files that the program creates for plotting purposes, and (iv) quit/close the window.

5.1.2 Edit

It has editing options. For now, the user can overwrite the path variables of β-NMR and β-NQR archive predefined in `constants.h`. Note that this is a temporary overwrite, and to make it permanent, one must modify the `constants.h` and re-compile.

5.1.3 View

It has the option of invoking a live/stream data window (see 5.2). Also, the user can change the view of the program widgets (default is `fuse`), and the color of the GUI (default is `"Green-white"`).

5.1.4 Plotting

Several check-boxes can be used for plotting purposes. The `XMGR` plots are closed by default after a new window is plotted to avoid the buildup of many `XMGR` windows. To keep the old plots active one must check the box `"Keep Plots"`. Also, in `XMGR`, the plots are by default separated, to combine them in one plot the user could check the box `"Combine Plots"`. At the moment, the GUI creates a lot of ASCII files in the background as needed by `XMGR`. These files are deleted by default after the user signals are processed. If the user wants to keep copies of the ASCII file, the box `"Keep ASCII files"` must be checked.

5.1.5 Help

This contains the `"About"` dialog for authorship and version of the current GUI, `"Tips"` dialog which does nothing but remind the user that by hovering the mouse index onto labels one can get the tool-tips for each widget. `"Tutorial"` invokes an HTML page with these instructions.

5.2 Streaming data

This window is invoked from `view/show streaming window`. It streams data during regular intervals of time as defined by the user. The user can choose the type of data (only `BNMR` is supported at the moment), run number, year, and settings for plotting options (bin, x min, x max). The user must specify the update interval.

Upon clicking on Start, the GUI starts a counter (in seconds) and then plots the data after each interval. The user can change the input variables without stopping the plotting as the GUI reads the input and plots the data again at the end of the end of each interval.

For `BNMR` data, at the moment, the GUI displays 4 plots and each contain 3 or 4 grouped curves. The first plot displays the asymmetries of the experiment counters of polarized $^{8}Li^{+}$ beam. These are; the positive helicity asymmetry (defined as $\frac{counter_1 - counter_2}{counter_1 + counter_2}$ during the + helicity of the laser), the negative helicity asymmetry, and the total asymmetry which is the difference of the first two. In the 2nd plot, similar asymmetries for the neutral beam are presented. In the 3rd plot, the counters of polarized $^{8}Li^{+}$ are shown, and the counts of beam monitors are shown in plot 4. An example is shown in Fig. 5

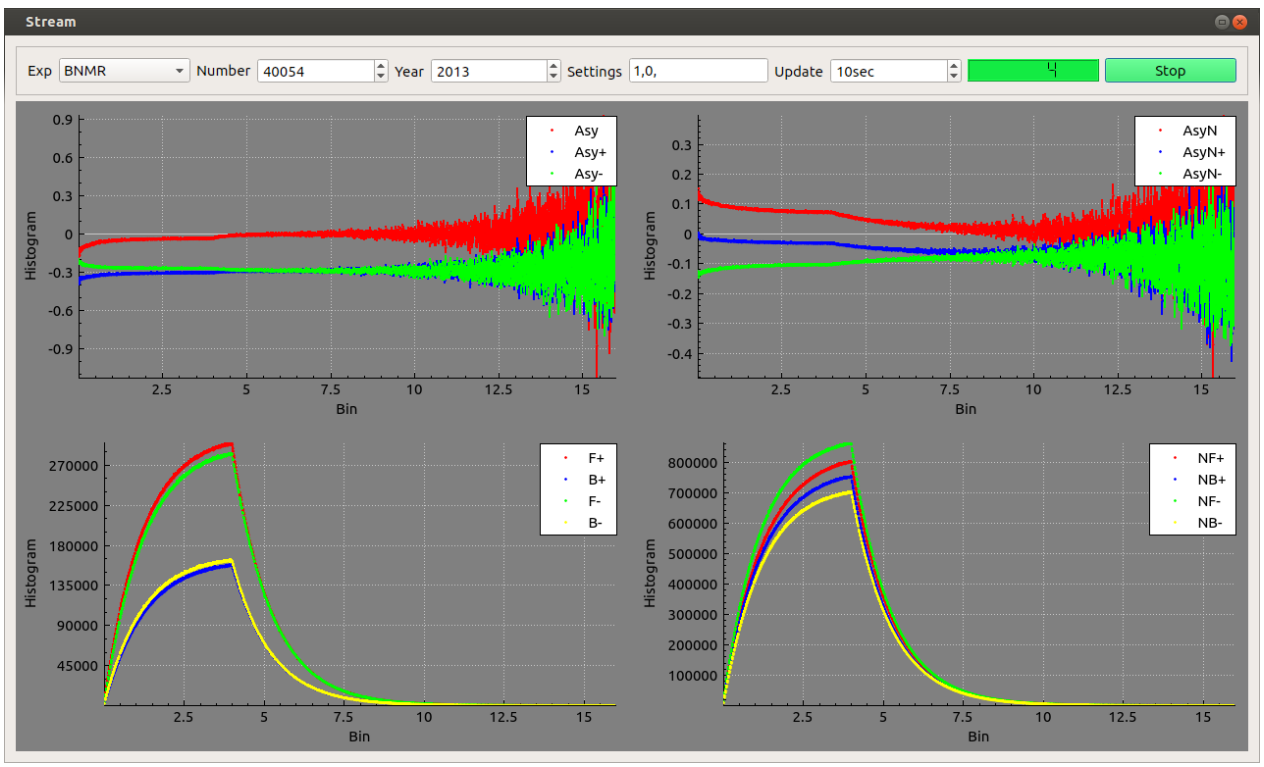

Figure 5: Streaming window: it streams data at regular intervals. This could be useful for on-line experiments.

5.3 Search

In this page, the user can find the data that corresponds to his search query. There are 9 fields that user can change. In the first line one finds, the title field (must be a string), experiment type (either β-NMR or β-NQR data at the moment), type of data (1f, 1n, 20, and 2e modes of `BNMR` data). In the 2nd line, the user can specify the intervals of year, run number and elapsed time. In the 3rd line, the user can specify the interval of independent variables of temperature, energy, and field. These are defined in `tab_search.cpp` code. The user can start the search by clicking on the pushbutton `Search`. After sometime, the search results will be returned in a table with 9 columns. These are (1) check-box columns to choose which runs to send to the analysis page, (2) run number, (3) year, (4) type of experiment mode, (5) elapsed time in minutes, (6) temperature in K, (7) energy in keV, (8) field in Gauss, and (9) the run title.

The user can also display more columns by clicking on the pushbutton `"More"` which displays a dialog that contains more fields to display. These fields are pre-defined in a template file called `bnmr-bnqr-logs.txt` which contains 5 columns, (i) Labels, (ii) Symbols, (iii) Exp, (iv) Path, and (v) Unit. The user can change this file to add more fields or to update the previous ones. The program will read the file saved in the current working directory, and if not found, it reads the default file from `resources.qrc`.

The user can select all runs by clicking on `"Select"` and send the selected runs to the next page for analysis by pushing the button `"Next"`.

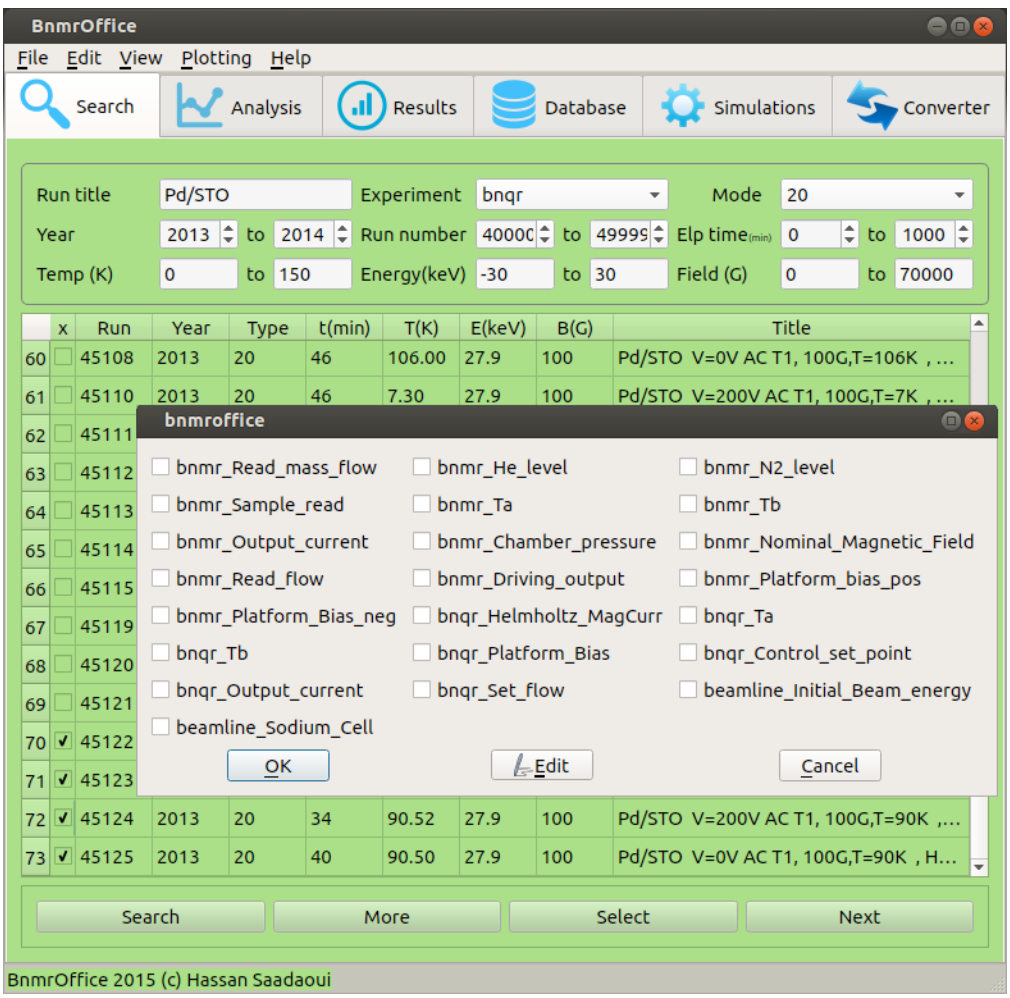

Figure 6: Searching the .msr files. More fields can be displayed after clicking on "More" and checking the desired fields. These fields are predefined in `"bnmr-bnqr-logs.txt"` which can be edited by pushing `"Edit"`.

5.4 Regression

This is the main page of the entire program and by far the most complicated to program. The network of background signals are large and may cause the program to crash unexpectedly. Please report these cases.

5.4.1 Data input

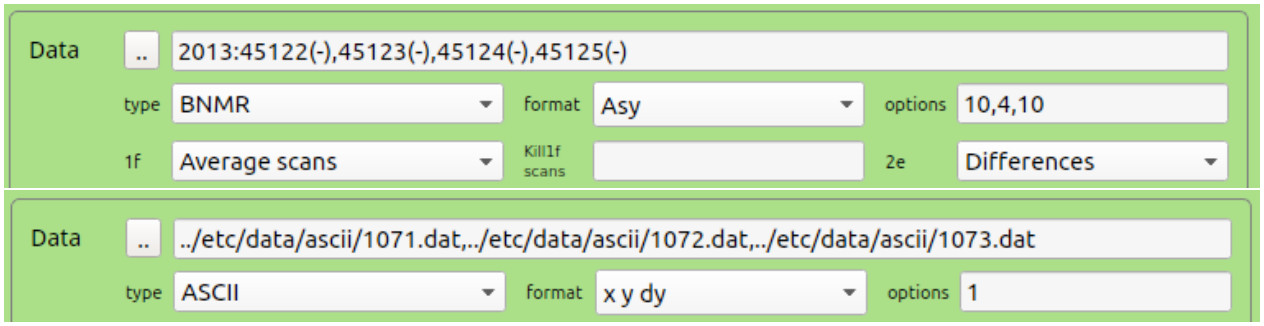

Figure 7: Options for data input

The user can choose to fit either ASCII text data or `.msr` files of the `CMMS` facility. The user can locate the data using the tool button next to Data for ASCII files. For `.msr` files, the user can only specify the run year and number and the program looks it up in the archive directory (defined in `constants.h` or `Edit`). For each of these types of data, there are three ways to specify the input data. For the `.msr` type follow these instructions;

1. Using run numbers directly:
   - A year must always be specified.
   - "|" is used to separate runs of different years.
   - "," (coma) separates between run numbers.
   - "-" separates between two numbers, which defines a range of runs.
   - "(-)" is used to flip the asymmtry opposite to the original one, (either up or down).
   - No other characters or space are allowed.
   - Example 1: "2011:45012,45672,42333|2013:45333(-),40123"
   - Example 2: "2011:40100-40110"
2. Using `.inf` files:
   - It is best to create `.inf` file out of `temp.inf` created by the program from a direct input method above.
   - The file `temp.inf` is found in the working directory.
   - Must follow the same template as of `temp.inf`
   - The user can change the file and update the independent variables.
3. Using `.list` files
   - The content of the file must follow the same instructions as run numbers.
   - Example: A file named `"myfavoriteruns.list"`, contains a single line `2010:45012,45672,41223,45012,40072,42313|2014:45123(-),40003`.

Reading/fitting ASCII files follows very similar instruction to above. Files to fit should be either written in the lineEdit (separated by commas) or; in files of `.list` or `.inf` extension as above. The file must reside in the working directory, otherwise its full name with path should be given. Examples;

- Direct input: "file1.txt,file2.txt,file3.txt"
- `.list` file: contains a single ascii line:
  `"file1.txt,file2.txt,file3.txt"`
- `.inf` file: `"myruns.inf"`, contains the columns
  files year temp
  file1.txt 2015 100
  file2.txt 2014 200
  file3.txt 2015 300

The format of `.msr` files to create are pre-defined as asymmetry or counts and are all in xydy format. The inner format of the ASCII file must be set in the field `format` and these must be columns with numbers and no other characters.

For the `.msr` files, there are several more options to tweak, such as the averaging of 1f runs, ignoring some bad 1f scans, and setting the type of 2e asymmetry. An experienced `BNMR` user must be familiar with these options.

The data can be binned and the limits of xmin and xmax values can be set in the options (settings) field. These must be numbers separated by commas.

5.4.2 Fitting selection

Figure 8: Fitting functions input.

The user can select the function to use, the mode of fitting (single or global) and type of errors. The functions are defined in the folder `fct/` and the user can add new ones by invoking the selection `"Create New"` in the functions comboBox. The user must follow the instructions in the pop-up window and then select `"Update"` from the comboBox. This will add the newly defined function to the list.

For the global method the user can choose to show all parameters for each run or not. These settings can be changed by double clicking on the initial parameters tab-widget.

The errors are defined by `MINUIT` routine, and are symmetric (Migrad) or asymmetric (Minos) errors. The latter are heavy to compute and the program may become unresponsive for sometime while the computation is going on. For further details read http://seal.web.cern.ch/seal/documents/minuit/mnerror.pdf.

5.4.3 Results output

Figure 9: Fitting results output.

The fitting results are written to this file. The user must specify a name, or browse for an old file. The results can be either appended (using `Append`) to the old file keeping its content (useful for doing run by run fitting), or the old file is overwritten using `Rewrite`.

5.4.4 Parameters input

Initial parameters ✖

Exp ✖

	Parameters	Value	Lower limit	Upper limit	Fix
1	Amplitude	0.2000	0.0000	1.0000	☐
2	Rate	0.1000	0.0000	1.0000	☐

Initial parameters ✖ Results 1 ✖

Exp ✖ Exp.G ✖

	Parameters	Value	Lower limit	Upper limit	Fix	Share
1	Amplitude	0.0848	0.0000	1.0000	☐	☑
2	Rate	0.0651	0.0000	1.0000	☐	☐
3	Amplitude2	0.0848	0.0000	1.0000	☐	☑
4	Rate2	0.0651	0.0000	1.0000	☐	☐
5	Amplitude3	0.0848	0.0000	1.0000	☐	☑
6	Rate3	0.0651	0.0000	1.0000	☐	☐
7	Amplitude4	0.0848	0.0000	1.0000	☐	☑
8	Rate4	0.0651	0.0000	1.0000	☐	☐

Figure 10: Input parameters for (a) single and (b) global fits. (a) The fit starts from this table for each file, or from the results of the last file in the sequence enabled by double clicking on the `"initial parameters"`. (b) If a parameter is shared between files, only the parameter of the first run is active and the same parameter for other files becomes inactive.

The initial parameters are read from the function library. The table contains 5 columns for the single method, and 6 columns for the global method. These columns are; (1) parameter name, (2) initial value of the parameter, (3) lower limit, (4) upper limit, (5) fix the parameter checkBox, and (6) share the parameter checkBox.

These parameters can be changed, and saved in a template for future use by right-clicking on the specific table and then choose `"save as a template"`. This creates a text file template with a prefix `".tab"`. The user can change this text file as required, and the template can be loaded later for a similar function.

5.4.5 Parameters output

Initial parameters ✖ Results 1 ✖ Results 2 ✖

	Files	Amplitude	Rate	Chisq
1	45122	0.1058±0.0185	0.1083±0.0392	1.0700
2	45123	0.1058±0.0185	0.1037±0.0368	1.0700
3	45124	0.1058±0.0185	0.1083±0.0364	1.0700
4	45125	0.1058±0.0185	0.0942±0.0373	1.0700

Figure 11: Results of a global fit where the shared parameter is Amplitude.

This prints out the output of the fit. The number of significant figures can be set by double-clicking on the results tab. One can also change the number of errors to show, and the way the filename is displayed.

5.5 Results

This page reads the files of fitting parameters created by the analysis page. It displayed a table with two columns, the left column represents the x-axis and the right column the y-axis. Each column contains all fields found in the specified file (as created during the fitting).

The user can check any of the fields, and a matrix of plots of y versus x will be displayed. The user can clear all choices using `"Clear"`, and kill/delete the active table using `"Purge"`.

The plots are made using `XMGR` plotting program. Horizontal or vertical error bars are displayed if specified in the chosen parameter. The user can edit the `XMGR` windows as he pleases and can make publication quality figures out of these results.

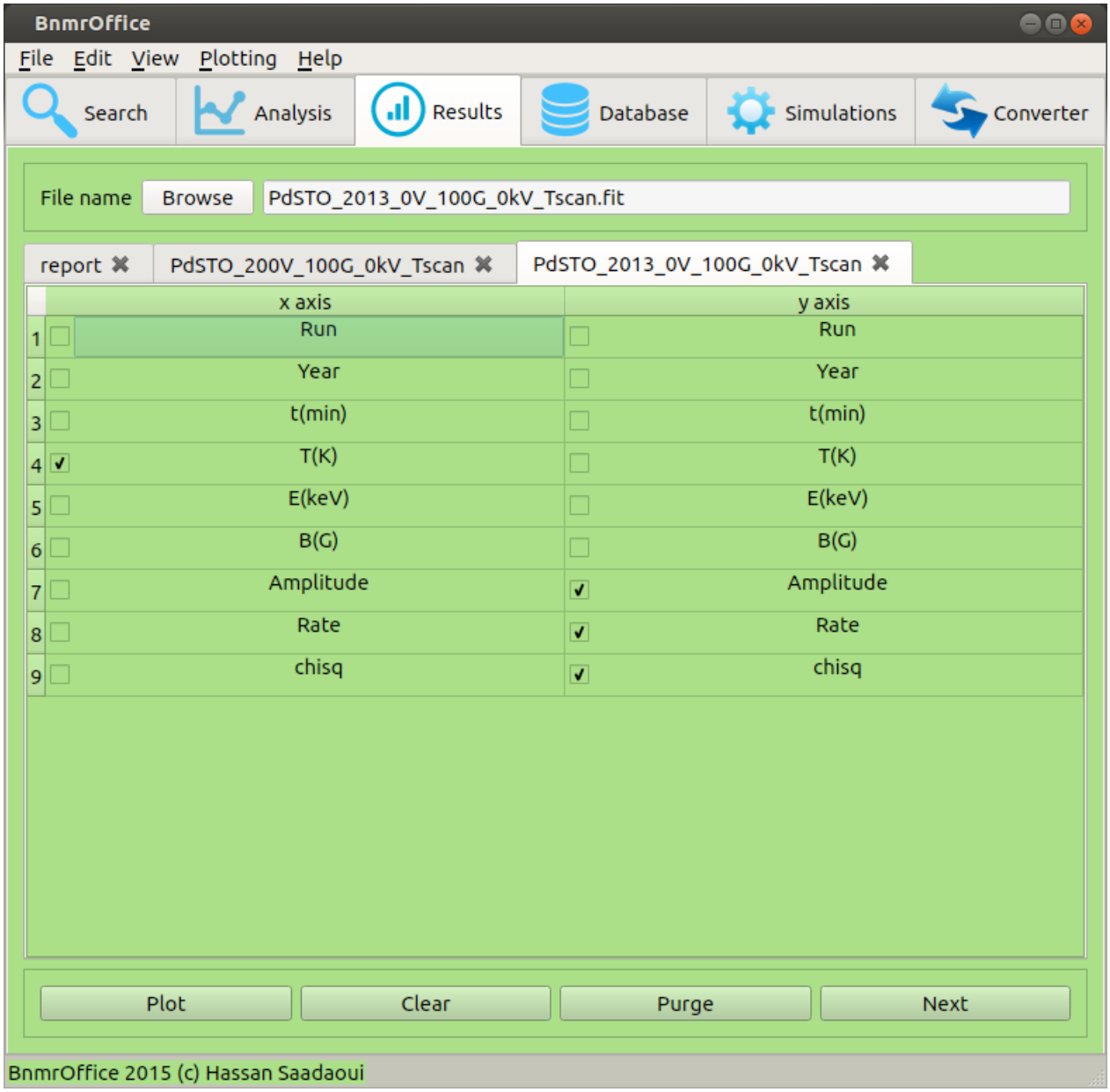

Figure 12: Results page.

5.6 Database

This page offers a user-friendly interface for databases, and uses SQLite language http://www.tutorialspoint.com/sqlite/sqlite_overview.htm. At the start, the user must select a database by clicking on the toolButton next to "Database", or create a new one from the "Querry" lineEdit using SQL commands and hitting "Execute". It is advised to use an SQL manager (like the friendly browser extension SQLite manager) to create databases and tables. Then, one can use this interface to add/delete rows and edit cells, interact with the content of the database. But an experienced user can do everything from this page as well by executing the "Querry" commands.

Te get familiar with the interface two databases are pre-made and come with the program and can be found in the folder "etc/sql/". These are called "bnmr.sqlite" and "physics.sqlite". Each contain several tables. The user can load any of these tables from the comboBox, and a model of the table will be displayed.

The user can execute any query to study the loaded table. Example;
`SELECT * FROM table_of_constants where Unit="kg"` will select all fields in the `table_of_constants` where the unit is in kg. The user must be familiar with SQL to execute from the Query field. Any table can be changed by adding or deleting rows. Also each cell can be edited, or displayed by clicking on "Open". This can be used to display a cell with a lot of text or view the cell as image if the full path of the image was given in that cell.

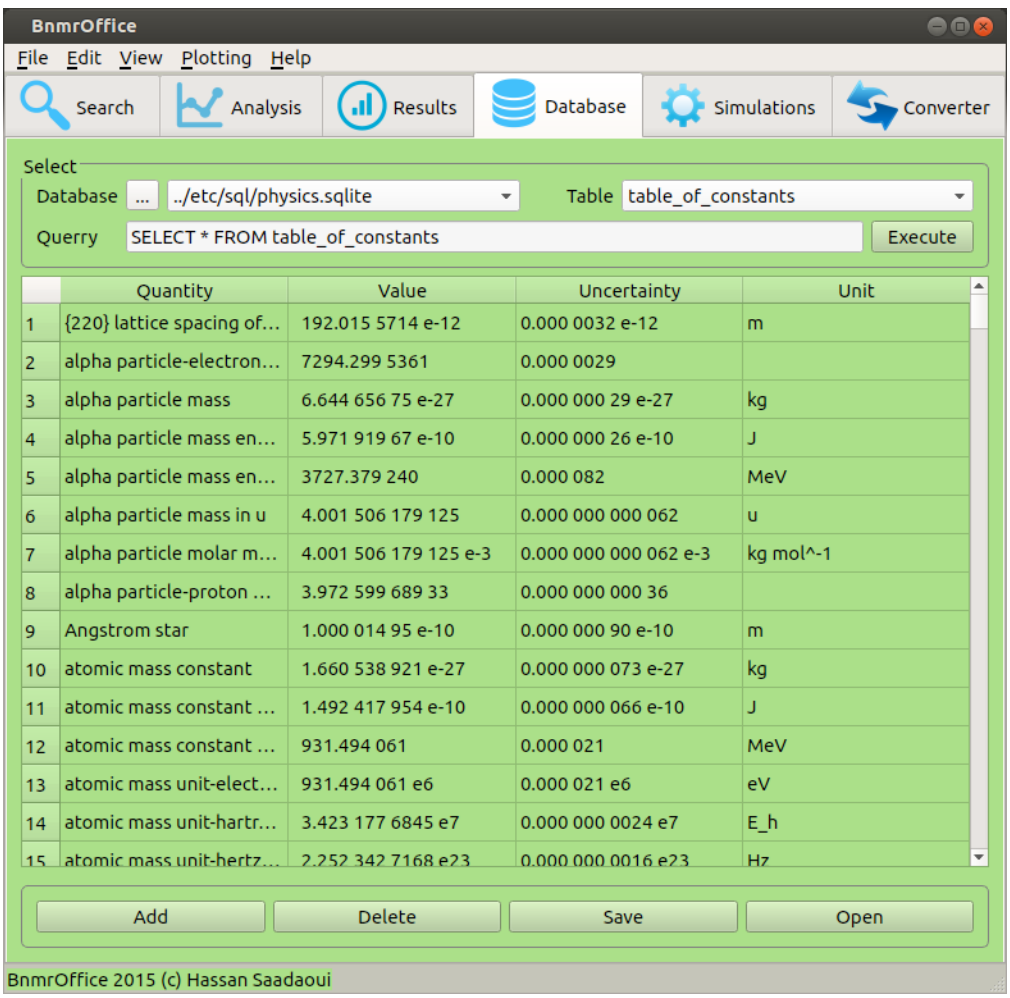

Figure 13: Database interface loaded with a table of physical constants.

5.7 Simulations

5.7.1 Van-Vleck second moment

In this page, one computes the dipolar second moment using the Van-Vleck method [J. H. Van Vleck, Phys. Rev. 74, 1168 (1948)]. The user specifies the implanted ions, target material, and type of lattice, and then the coordinates of each site. The results will be plotted on the same page using `QCustomPlot` libraries. The plots can be exported into .pdf, .png, and .bmp files. ASCII files of the computed are created in the working directory.

The simulation is tested against two published papers in (1) Lattice locations of 8Li in Cu, Hyperfine Interactions 120/121 (1999) 419-422. (2) Location of 12B in Al and Cu, Phys Rev B 13, 34 (1976). The source code of this calculation is originally written by Samir Tabara, and adapted for `BnmrOffice`.

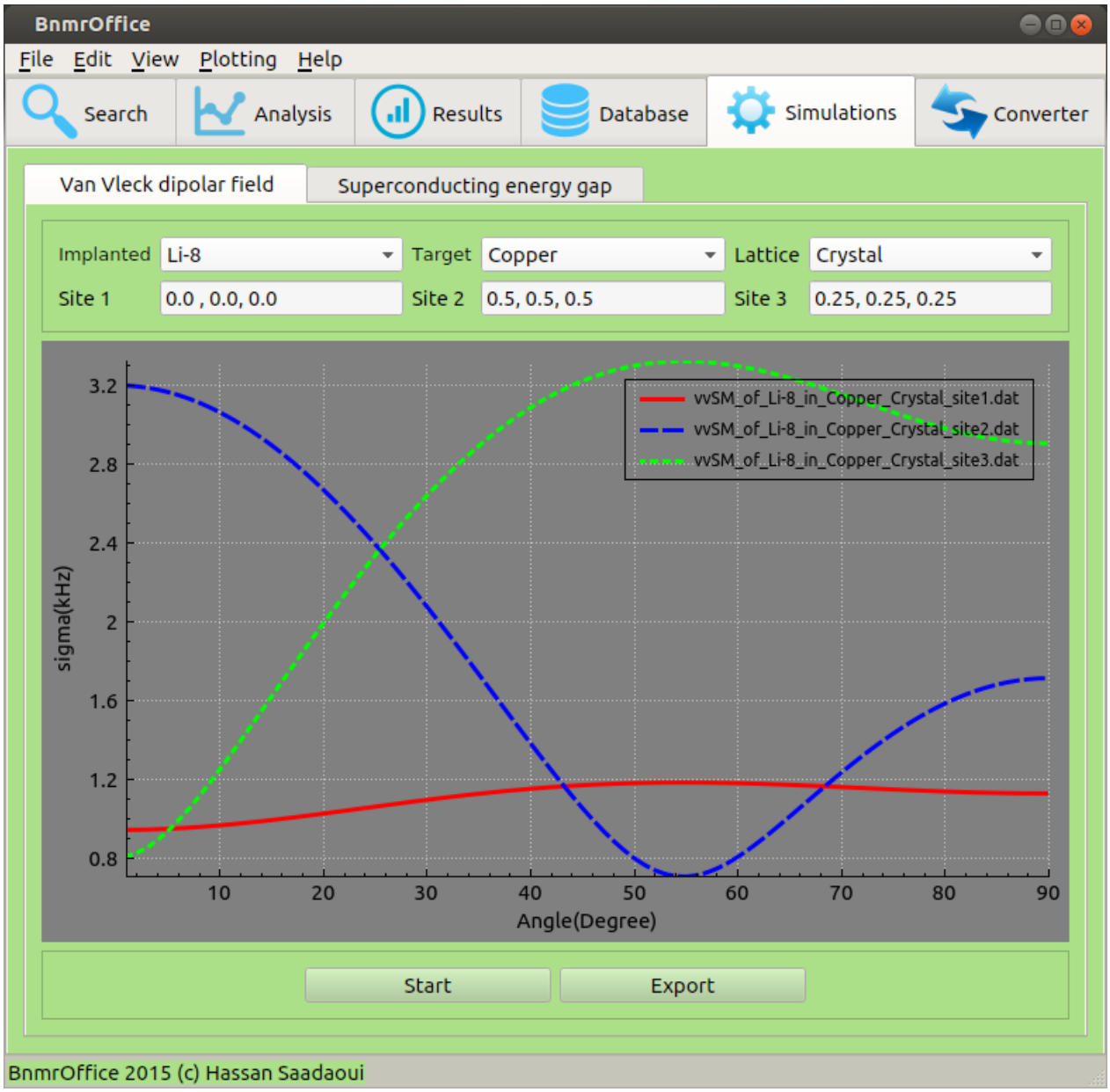

Figure 14: Van-Vleck calculation

5.7.2 Superconducting energy gap

This page performs calculation of the superconducting energy gap, for a given gap symmetry and parameters such as the penetration depth and coherence length. The definitions can be found in the papers H. Saadaoui *et al.*, Phys. Rev. B 88, 094518 (2013), and R. Prozorov and R. W. Giannetta, Supercond. Sci. Technol. 19, 41 (2006).

The simulation computes three functions, the second moment of the magnetic field distribution versus temperature (`sigma_vs_temp`), the second moment of the magnetic field distribution versus field (`sigma_vs_field`), and the penetration depth versus temperature (`lambda_vs_temp`).

Four forms of the energy gap $\Delta(T, \mathbf{k})$ are pre-defined; the s-wave, d-wave, s-wave-dirty and d-wave non-monotonic. These are defined in Eqs. 17 and 18 in the Prozorov paper. Two empirical forms of the second moment of the magnetic field distribution versus field are specified, the Modified-London and Ginsburg-Landau forms. Please refer to the paper of H. Saadaoui *et al.*, Phys. Rev. B 88, 094518 (2013) for further details about these functions.

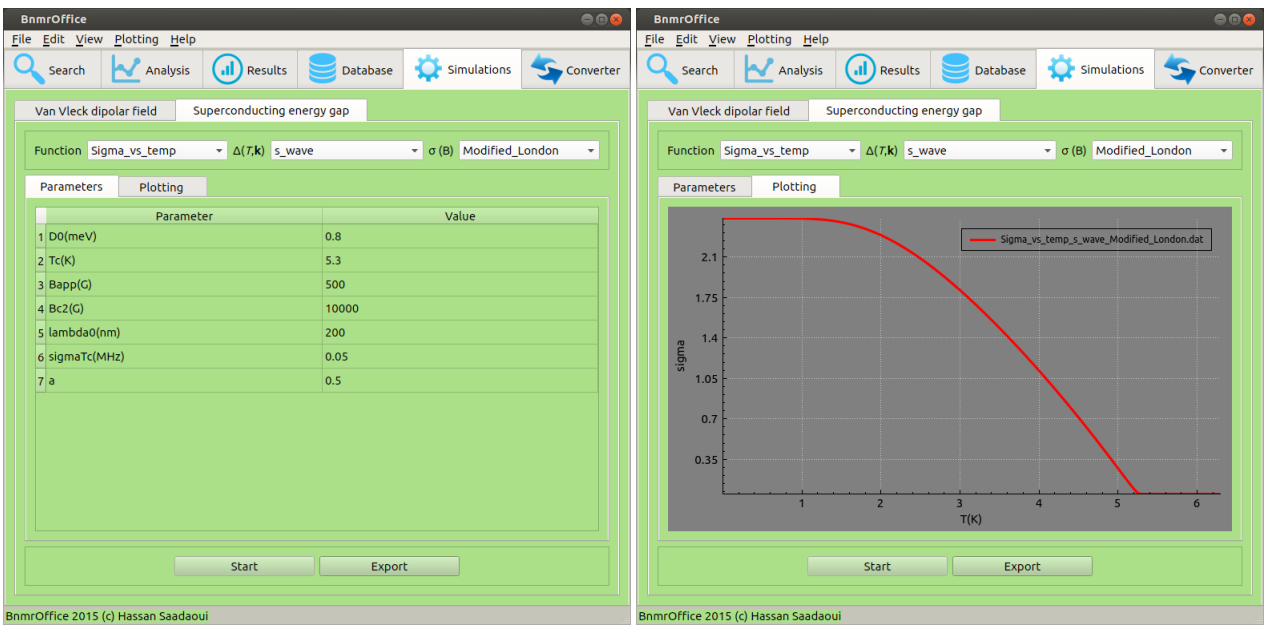

Figure 15: Input and output of the energy-gap interface

5.8 Converter

This page is used to convert between several units useful for a `BNMR` user. The definitions are given in `"tab_convert.cpp"` file. The user can change any field and hit Enter, and the associated field will update.

- Magnetic Field (G) = Frequency (kHz)/0.63015
- Magnetic Field (G) = 2.2131*Current(Amp) + 0.175
- Pulse duration (ms) = 5*10000/π/Bandwidth(Hz)
- Pulse duration (ms) = 1764.8/Bandwidth(Hz)
- Magnetic Field (G) = Frequency (kHz)/85.16

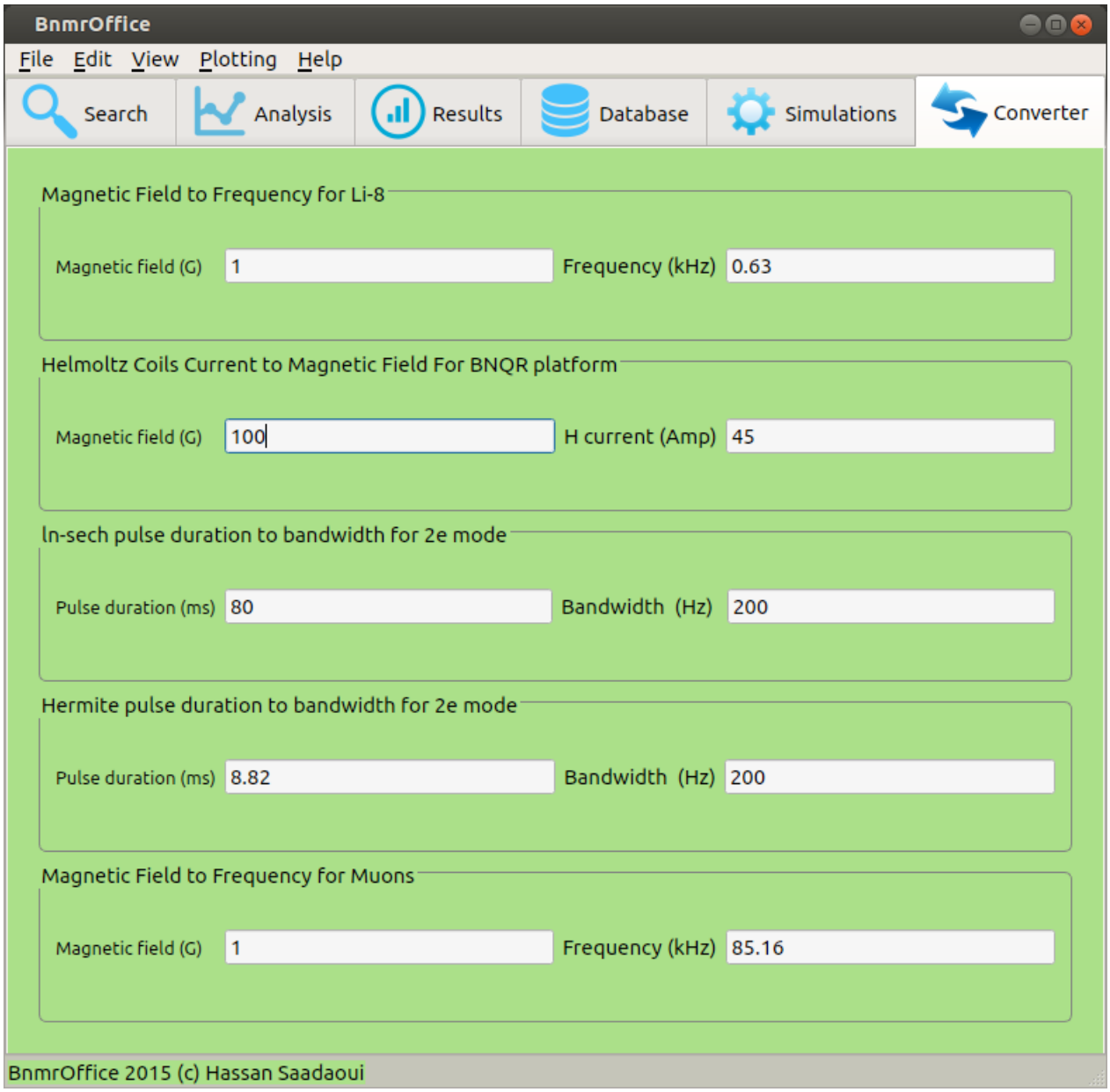

Figure 16: Converter page.

6 Fitting functions

The user can write his own fitting functions in the directory `fct/`. A new function must be written in `C++` but requires minimal programming knowledge of this language. At template of a typical function is as follows:

```
#include <iostream>
#include <fstream>
#include <math.h>
#include <stdio.h>
#include <vector>
#include <sstream>
#include <string.h>
#include <iostream>
using namespace std;
#include "parameters.h"

//Wrap in "C" for the compiler.
extern "C"
{
  //The defalt initial parameters loaded to the table.
  void defaultParameters(Parameters &defaults)
  {
    defaults.addParameter( "Amplitude" , 0.2, 0.001, 0.15, 1.0 );//par[0]
    defaults.addParameter( "Rate"      , 0.1, 0.001, 0.0,  1.0 );//par[1]
  }
  //This must have the same number of parameters as in the default parameters above.
  double function(double x, const std::vector<double> par)
  {
        return par[0]*exp( - par[1]*x);//par[0] is amplitude and par[1] is rate.
  }
}
```

The user must follow these instructions:

- Make a copy of the file `newFunction.cpp` found in `etc/files`.
- Rename the file (eg: newfct.cpp) and save it in the folder `fct/`.
- cd to the directory fct, and run the script `./compile` as root
- $ `sudo ./compile name` (eg: sudo ./compile newfct.cpp)
- This compiles the library and puts a copy in the library functions folder (`/opt/bnmroffice/lib/`).