# muSRFlow User Manual

*Release 1.0.0-rc.6*

**Zaher Salman**

**Nov 02, 2026**

# Contents


**1 Overview** **3**
- 1.1 What you get . . . 3
- 1.2 Under the hood . . . 5
- 1.3 Two editions . . . 5
- 1.4 A first fit . . . 5

**2 Getting started** **7**
- 2.1 Choose your edition . . . 7
- 2.2 Desktop installation . . . 7
- 2.3 Windows installation . . . 7
- 2.4 Running the desktop app . . . 8
- 2.5 Opening the web edition . . . 8
- 2.6 The Quick Start guide . . . 8
- 2.7 Support . . . 9

**3 Windows installation** **11**
- 3.1 Before you start . . . 11
- 3.2 Install step by step . . . 12
- 3.3 Existing installations . . . 13
- 3.4 Troubleshooting . . . 13

**4 Linux installation** **15**
- 4.1 Install the application . . . 15
- 4.2 Set up ROOT and musrfit . . . 16
- 4.3 Choose folders and test . . . 17
- 4.4 Updating and troubleshooting . . . 17

**5 Runs and fitting** **19**
- 5.1 Run selection . . . 19
- 5.2 Fit range . . . 20
- 5.3 Fit model . . . 20
- 5.4 Fit options . . . 21
- 5.5 Series fit actions . . . 21

5.6 View settings . . . 22
5.7 Fit and Plot . . . 22

**6 Parameters and parameter plots** **25**
6.1 The parameter table . . . 26
6.2 Selecting parameters for plotting . . . 26
6.3 The Choose X axis dialog . . . 27
6.4 The parameter-plot window . . . 27

**7 LEM single-histogram example** **33**
7.1 Data and model . . . 33
7.2 Inspect the fitted run . . . 34
7.3 Compare the energy scans . . . 34
7.4 Reproduce and inspect . . . 35

**8 Constraints** **39**
8.1 Custom parameters . . . 39
8.2 Constraints . . . 39
8.3 Example: couple the magnetic field to the Larmor frequency . . . 40
8.4 Preview . . . 40

**9 t0 / background** **41**
9.1 Multi-group asymmetry fits . . . 41

**10 MSR editor** **43**
10.1 Ownership: GUI or editor . . . 44
10.2 Workflow . . . 44
10.3 File handling . . . 44

**11 Results and diagnostics** **45**
11.1 Result files . . . 45
11.2 Execution details . . . 45
11.3 Command log . . . 46
11.4 The status bar . . . 46
11.5 Run log window . . . 46
11.6 Outcome dialog . . . 46

**12 Configuration sharing** **47**
12.1 What is shared . . . 47
12.2 On the web . . . 47
12.3 On the desktop . . . 48
12.4 Notes . . . 48

**13 Preferences** **49**

**14 Web and desktop editions** **51**
14.1 Bundled ROOT and musrfit . . . 52
14.2 Downloading missing runs . . . 52

**15 Keyboard shortcuts** **55**

15.1 Parameter selection (Parameters tab) . . . 55
15.2 Parameter-plot window . . . 55
15.3 MSR editor . . . 55
15.4 General notes . . . 56

**16 Troubleshooting** **57**
16.1 Fit or plot will not start . . . 57
16.2 “Nothing to fit” . . . 57
16.3 Fitted parameters look wrong or unchanged . . . 57
16.4 Parameter plot is empty or missing data . . . 58
16.5 Parameter plot hangs on “Preparing plot files” . . . 58
16.6 Web page cannot load data . . . 58
16.7 Restored configuration differs from the original machine . . . 58
16.8 Some links start with a different prefix . . . 58
16.9 Reporting problems . . . 59

**17 Glossary** **61**

**Index** **63**

muSRFlow is a graphical workflow for creating, editing, fitting, plotting, and inspecting `musrfit` MSR files. It is available as an Electron desktop application and as a browser application, and both editions wrap the same set of command-line tools (`musrfit`, `musrview`, `msr2data`, and `dump_header`).

Start with *Windows installation* or *Linux installation* for desktop setup. For the browser edition, see *Getting started*.

These pages describe the end-user workflow. They are organised around the main window tabs, mirroring the Quick Start help that is built into the application (the `?` button).

# CHAPTER 1

# Overview

muSRFlow provides a modern interface over the classic μSR analysis tools. A typical session selects the data, chooses a fit model, fits it, inspects the result, and finally inspects how the fitted parameters vary run by run.

## 1.1 What you get

- A single window with the full workflow in tabs:
  - **Runs** – run selection, factorial fit range, fit model, fit options, and the Plot / FFT / RRF view settings.
  - **Parameters** – the parameter table with starting values, steps, limits, and plot selection.
  - **Constraints** – custom parameters and parameter constraints.
  - **t0 / Bg** – expert timing and background controls.
  - **MSR Editor** – the generated MSR document, with GUI-to-MSR and MSR-to-GUI synchronisation.
  - **Results** – generated files, execution details, and the full command log.
- Two one-click workflow actions in the main toolbar:
  - **Fit** – generate the MSR document and run the fit.
  - **Plot** – generate the MSR document and display the spectra.
- Interactive parameter plots that show how every fitted parameter evolves across the selected runs.
- Configuration sharing: the current analysis settings can be encoded into a link and restored on another machine.

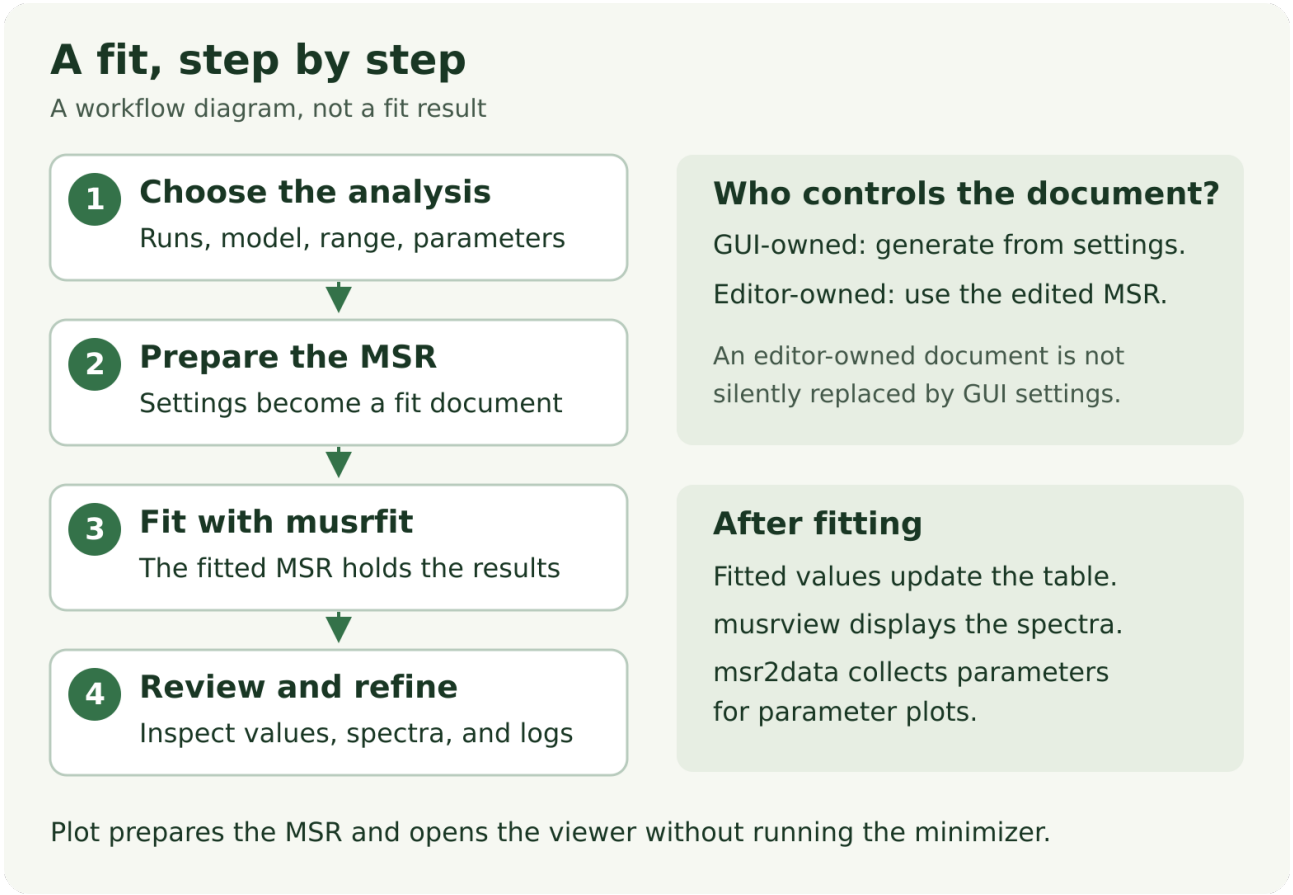



Fig. 1: The basic fit workflow. See *Runs and fitting* for independent series and shared-parameter fits, and *MSR editor* for document ownership.

## 1.2 Under the hood

muSRFlow does not reimplement the physics. It writes MSR files that are fed to the musrfit package, and renders spectra with `musrview`:

- `musrfit` – the muon-spectroscopy fitting engine. muSRFlow generates an MSR file from the GUI and runs `musrfit` on it. The fitted values (a `par.dat` parameter table) feed the parameter plots.
- `musrview` – the reference data viewer. On the desktop it opens its own native window; in the browser edition the server produces a ROOT canvas that is drawn with JSROOT inside the page.
- `msr2data` – converts a fitted MSR file into a parameter table (`par.dat`) that the parameter plot can read.
- `dump_header` – reads run metadata used to resolve portable run names and to guess initial parameters.

## 1.3 Two editions

- **Desktop (Electron)** runs the tools directly on your machine, manages its own data and working directories (set under **File / Preferences**), and can open and save MSR files from the File menu.
- **Web** runs the same interface in a browser. The data tree and the command-line tools live on the server, so file selection is limited to what the server exposes, and there is no local Preferences dialog.

Both editions share the same workflow and produce the same MSR documents. The differences that matter day to day are collected in *Web and desktop editions*.

## 1.4 A first fit

1. In the **Runs** tab, choose the instrument and year and enter one or more run numbers.
2. Adjust the fit range and the fit model as needed.
3. Press **Fit**.
4. Open the **Parameters** tab to verify and adjust starting values, then press **Fit** again.
5. Use **Plot** any time to look at the spectra.
6. Select parameters and choose **Plot parameters…** to inspect how they vary across the run set (see *Parameters and parameter plots*).

The Quick Start button (`?`) at the right of the tab bar opens a short guide that highlights the section relevant to the active tab. Its sections are expanded into the chapters of this manual.

CHAPTER 2

# Getting started

## 2.1 Choose your edition

- **Windows desktop:** follow *Windows installation*. ROOT and musrfit are included in a WSL2 runtime. The guide covers first installation, existing runtime updates, and Windows-specific errors.
- **Linux desktop:** follow *Linux installation*. Use an existing musrfit installation or install the separate runtime through Preferences.
- **Web:** open your site's muSRFlow URL, for example `http://musruser.psi.ch/muSRFlow/`. The server runs the tools; there is nothing to install on your computer.

Packaged desktop applications do not require Node.js or npm.

## 2.2 Desktop installation

The complete instructions are in *Windows installation* and *Linux installation*. Choose work and data folders in **File / Preferences** before your first analysis. Keep work files outside the application installation directory.

## 2.3 Windows installation

The Windows instructions have their own chapter: *Windows installation*. It also contains the update steps for older WSL runtimes.

## 2.4 Running the desktop app

When the window opens, the **Runs** tab is active. Choose the instrument, year, and run numbers, check the model, and press **Plot** to inspect the spectrum. Press **Fit** to fit it and refresh the parameter table. Generated files go to the selected work directory; the **Results** tab and run log show the outcome.

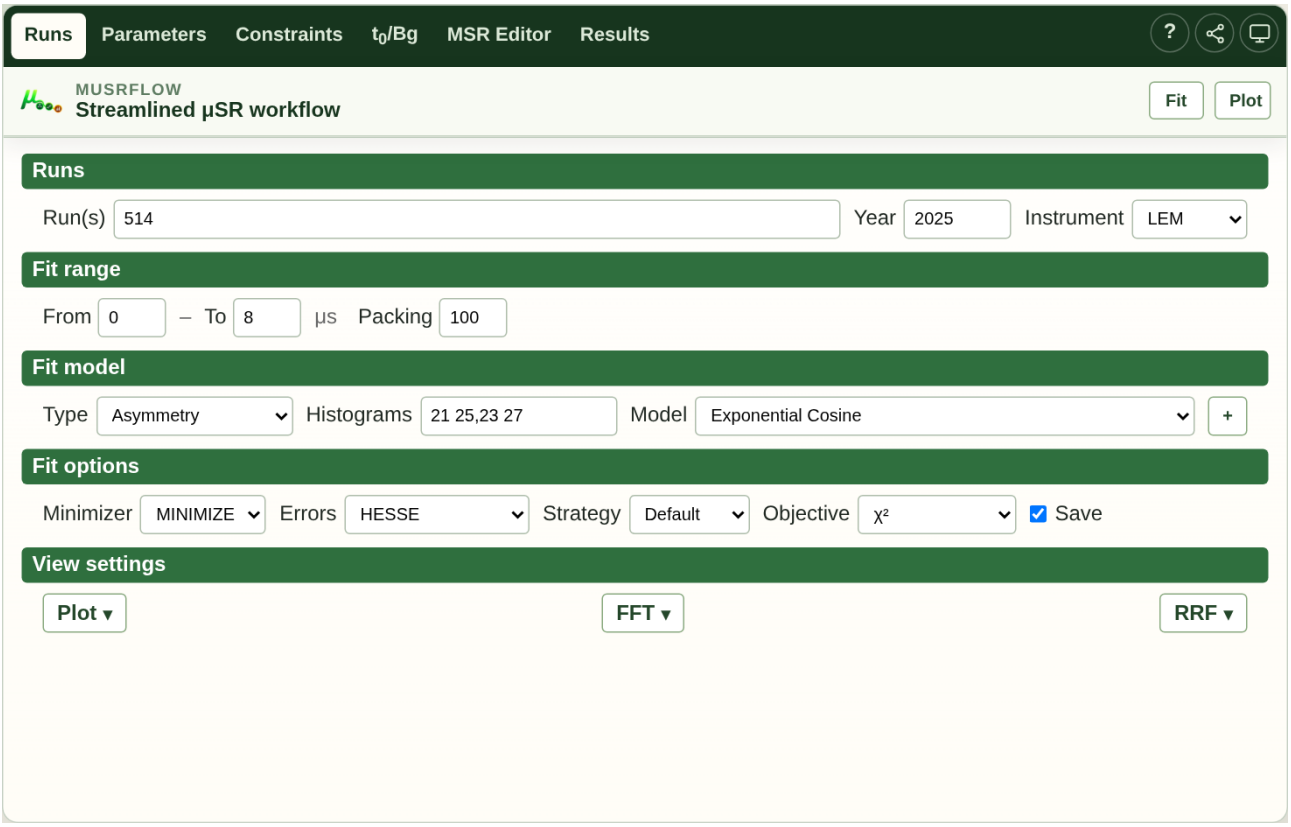



Fig. 1: The browser edition before fitting. Run 514 (LEM, 2025) is an example selection, not a validated fit configuration. Choose your runs and model here; the Fit and Plot buttons remain available across tabs.

## 2.5 Opening the web edition

Open the deployment URL in your browser. Instrument options and available run data depend on that site's server configuration. Fit and Plot run on the server; output files can be opened from the result links.

## 2.6 The Quick Start guide

The ? button at the right of the tab bar opens a short, tab-aware guide. On the desktop it opens in an independent window; in the browser it opens a modeless dialog next to the main window. The full workflow chapters follow the installation guides in this manual.

## 2.7 Support

For questions and bug reports, include the version and run log and contact Zaher Salman. See *Troubleshooting* for common workflow problems.

For a complete example using real fitted LEM data and application screenshots, see *LEM single-histogram example*.

CHAPTER 3

# Windows installation

The Windows installer includes muSRFlow, ROOT, and musrfit. Install WSL2 first; muSRFlow prepares its own Linux runtime on first launch. You do not need Node.js, npm, Ubuntu, or a separate ROOT/musrfit installation.

Already installed muSRFlow? Read *Existing installations* as well: updating the application does not update an already imported WSL runtime.

## 3.1 Before you start

- Use a **64-bit Intel/AMD PC**. The current package is x64.
- Windows 11 is recommended. Windows 10 requires build 19044 or newer for WSLg, which displays the musrview spectrum windows.
- Administrator access is needed to enable the Windows virtualization features.
- Obtain the Windows installer, named `muSRFlow-<version>-x64.exe`, from the project's Releases or the **musrflow-windows** package in Packages. Verify its SHA-256 checksum against the accompanying `.sha256` file. The Windows installer includes the bundled runtime (`musrfit`, `musrt0`, `musrview`, `msr2data`, `dump_header`, `root-config`) for use inside WSL2; only markers are placed in the app resources.
- For a test build that has not been published yet, download the **musrflow-windows-x64** artifact from a successful **Build Windows installer** workflow run and extract the installer.

**Avoid old rc.5 installers.** Early builds lack the WSL fixes. The current runner-built installer includes them and has passed Windows installation, runtime setup, data download, Fit/Plot, and multi-run smoke tests.

See Microsoft's WSL GUI requirements for the Windows prerequisites.

## 3.2 Install step by step

1. **Install WSL.** Open **PowerShell as Administrator** and run:

   ```
   wsl --install --no-distribution
   ```

   **Restart Windows afterwards.** Skip this step if WSL is already installed. muSRFlow supplies its own distribution, so Ubuntu is not required.

2. **Update and check WSL.** In administrator PowerShell, run:

   ```
   wsl --update
   wsl --version
   ```

   Follow any restart instructions. The output should include WSL and WSLg. These commands are described in the Microsoft WSL command reference.

3. **Install muSRFlow.** Double-click the Windows installer and complete setup. The runtime archive `muSRFlow-runtime-rootfs.tar.gz` is included; do not extract or import it manually for a normal installation.

4. **Launch muSRFlow** from the Start menu using your normal Windows account. First launch imports the runtime as `muSRFlow-runtime` and can take several minutes. Fit and Plot wait until preparation finishes. Afterwards, verify in PowerShell:

   ```
   wsl --list --verbose
   ```

   The list should contain `muSRFlow-runtime` with **VERSION 2**. `Running` and `Stopped` are both normal when idle. The runtime is stored under `%LOCALAPPDATA%\muSRFlow\muSRFlow-runtime` for this account.

5. **Choose work and data folders before fitting.** Create folders such as:

   ```
   C:\Users\<your-name>\muSRFlow\work
   C:\Users\<your-name>\muSRFlow\data
   ```

   In **File / Preferences**, select them as **Work directory** and **Data directory**. Leave **musrfit binaries** and **musrfit startup XML** on automatic detection, then save. Enter normal Windows paths; muSRFlow translates them for WSL.

   Use a work folder outside the application installation directory. The default `.` can resolve to the installation folder and place your results among files managed by the installer.

6. **Try Plot, then Fit.** Select a known instrument, year, and run. Click **Plot** and confirm that the spectrum window opens. Missing run data can be downloaded from `musruser.psi.ch` after confirmation when the server is reachable. Then click **Fit**, check that the parameter table updates, and inspect the generated files in your work directory.

## 3.3 Existing installations

1. Preserve any MSR/data/result files currently stored inside the muSRFlow installation folder: move them to a separate work folder before upgrading.
2. Close muSRFlow and install the refreshed Windows application.
3. If your runtime was imported from an older bundle, add the missing Boost headers from PowerShell:

   ```
   wsl -d muSRFlow-runtime -- dnf -y install boost-devel
   ```

   This needs access to the distribution's package repositories. ROOT uses these headers when loading musrview plotting classes. Fresh runtimes from the corrected installer already include them.
4. Reopen muSRFlow, select your work directory in Preferences, and retry Plot and Fit.

Reinstalling the application preserves an existing `muSRFlow-runtime` distribution, so the application update alone does not add those headers. The command above repairs it in place; do not unregister the distribution as part of this update. The Linux **Install runtime…** preference is not used on Windows.

## 3.4 Troubleshooting

### 3.4.1 Required feature is not installed

For `HCS_E_SERVICE_NOT_AVAILABLE` during runtime preparation, Windows could not create the WSL virtual machine. In **administrator PowerShell**:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all␣
↪/norestart
```

Restart Windows. Check **Task Manager / Performance / CPU** for **Virtualization: Enabled**; if disabled, enable hardware virtualization in BIOS/UEFI. If Windows itself is a virtual machine, its host must expose nested virtualization. If the error persists, check the boot setting:

```
bcdedit /enum | findstr -i hypervisorlaunchtype
```

Only if it reports `Off`, enable it and restart:

```
bcdedit /set hypervisorlaunchtype Auto
```

See Microsoft's WSL troubleshooting guide.

### 3.4.2 Cannot find wsl.exe

For `spawn wsl.exe ENOENT`, first check that `wsl --version` works in PowerShell. If it does, install the refreshed muSRFlow build: early rc.5 builds incorrectly replaced the Windows executable search path. Also check that the selected work directory exists and is writable.

### 3.4.3 Cannot load libCore.so

Install the refreshed muSRFlow build. It passes the ROOT library directory inside WSL for every bundled command. Setting a Windows `ROOTSYS` or `LD_LIBRARY_PATH` is not a substitute for this launcher fix.

### 3.4.4 Missing boost/version.hpp or plotting crash

Follow *Existing installations* to install `boost-devel` in an older runtime. The corrected runtime includes it. If plotting still crashes, include the full command output when reporting the problem.

### 3.4.5 No spectrum window appears

Confirm that `wsl --version` lists WSLg, update WSL, and inspect the muSRFlow run log. Successful runtime import or a successful fit alone does not verify interactive graphics support.

### 3.4.6 Reporting a problem

Include the muSRFlow version, full run log, and the output of:

```
wsl --version
wsl --status
wsl --list --verbose
```

The corrected runtime has completed a real fit and generated a PNG in a Linux container. Interactive Windows/WSLg plotting still needs validation on the target PC.

# CHAPTER 4

# Linux installation

Install the muSRFlow application, then select a working ROOT + musrfit installation. **The Linux application package does not include the scientific runtime.** The application bundles no musrfit binaries; use an existing ROOT + musrfit installation, install the separate runtime (AppImage/RPM or the offline bundle in Preferences), or provide tools via the configured binaries directory. Node.js and npm are not needed on the target PC.

The current application packages are for x86_64 Linux. Examples below use `1.0.0-rc.6`; substitute the version of the package you obtained. Run the commands from the folder containing the downloaded files.

## 4.1 Install the application

Choose one application format.

### 4.1.1 RPM

On an RPM-based distribution with dnf:

```
sudo dnf install ./musrflow-1.0.0-rc.6-x86_64.rpm
musrflow
```

On other RPM systems, use the distribution's package manager. The application RPM filename starts with lowercase `musrflow`.

### 4.1.2 AppImage

```
chmod +x muSRFlow-1.0.0-rc.6-x86_64.AppImage
./muSRFlow-1.0.0-rc.6-x86_64.AppImage
```

The application AppImage filename starts with `muSRFlow`. It requires a compatible Linux desktop and AppImage/FUSE support. If it cannot start, check the terminal output or use the RPM on a supported RPM distribution.

## 4.2 Set up ROOT and musrfit

### 4.2.1 Use an existing installation

Open **File / Preferences**. Leave **musrfit binaries** on automatic detection for a standard installation, or choose the directory containing `musrfit`, `musrview`, `msr2data`, `musrt0`, and `dump_header`. Save your settings, then use **Check tools** to verify the saved selection.

### 4.2.2 Install the separate runtime from a local bundle

1. Obtain the runtime bundle for your architecture. Keep `runtime-download.json` and its matching `musrflow-runtime-*.tar.gz` archive in the same folder. You do not need to unpack it.
2. Open **File / Preferences / Install runtime…** and select `runtime-download.json`.
3. Review the versions and destination, then select **Install and use**. muSRFlow validates the archive, installs it in your personal application data directory, and selects its tools.
4. Use **Check tools** to confirm the selected runtime starts.

This installation does not require administrator access. Runtime downloads are not automatic: obtain the bundle separately before opening the installer. A compatible Linux system and desktop/display services are still required.

### 4.2.3 Alternative: system runtime RPM

```
sudo dnf install ./musrflow-runtime-1.0.0-rc.6-x86_64.rpm
```

This installs the runtime into `/opt/musrflow/runtime`, which muSRFlow recognizes automatically when no explicit tool directory takes precedence.

### 4.2.4 Alternative: runtime AppImage

For a user-writable runtime location:

```
chmod +x musrflow-runtime-1.0.0-rc.6-x86_64.AppImage
./musrflow-runtime-1.0.0-rc.6-x86_64.AppImage --install "$HOME/.local/
↪share/musrflow/runtime"
```

Launch the application with the installed runtime selected:

```
MUSRFLOW_RUNTIME_ROOT="$HOME/.local/share/musrflow/runtime" musrflow
```

For the application AppImage, replace `musrflow` in that command with its path. Leave **musrfit binaries** on automatic detection so the selected runtime is used. Without a custom destination, the runtime AppImage installs to `/opt/musrflow/runtime` and needs permission to write there. Its filename is lowercase `musrflow-runtime`, unlike the application AppImage.

## 4.3 Choose folders and test

1. In **File / Preferences**, choose a writable **Work directory** outside the application installation folder, for example `~/muSRFlow/work`. Use **Browse…** to select it.
2. Set **Data directory** to the root of your instrument data tree, or to a writable folder such as `~/muSRFlow/data` for downloaded runs.
3. Leave **musrfit startup XML** on automatic detection unless you have a specific installation configuration, then save.
4. Select a known instrument, year, and run. Click **Plot**, then **Fit**. Confirm the spectrum opens, fitted parameters update, and output files appear in your work directory. Missing runs can be downloaded from `musruser.psi.ch` after confirmation when the server is reachable.

## 4.4 Updating and troubleshooting

- Upgrade the application RPM with your package manager, or replace the application AppImage. The separate runtime is not updated by an app upgrade.
- Preserve work files outside the application/runtime installation directories.
- For missing tools or shared libraries, run **Check tools** and inspect the reported executable paths. A configured binaries directory overrides automatic detection; clear an obsolete selection or choose the correct one.
- If fitting works but no spectrum window opens, inspect the run log and confirm your desktop session can display native musrview windows.
- If data are missing, check the instrument, year, run number, and data root. Downloaded files use the selected writable data root or the personal fallback folder reported in Preferences.

Include the application version and run log when reporting a problem.

# CHAPTER 5

# Runs and fitting

The **Runs** tab holds everything that selects and describes the data: the run list, the fit range, the fit model, the fit options, and the view settings used for Plot, FFT, and RRF.

## 5.1 Run selection

**Run(s)**
: One or more runs, separated by commas. A dash selects a contiguous range and a colon-with-step `from:to:step` produces a regular sequence, for example `520:540:2`.

  Each comma- or range-separated data set is fit independently and merged into one fit document, so `514, 516, 520-525` prepares eight data sets.

  A `+` between runs adds the run to the previous data set via the MSR `ADDRUN` directive rather than starting a new data set. `514+515` is a single data set composed of two runs.

  Examples:

  - `514` – one run.
  - `514, 520-525` – seven data sets: 514 and each of 520..525.
  - `514+515, 520:540:2` – one ADDRUN data set and a stepped range.

**Year**
: The data-taking year. The year range starts at 1998 and cannot exceed the current year.

**Instrument**
: The spectrometer. The exact choice (for example LTF, FLAME, LEM, GPS, GPD, Dolly, VMS, HAL) depends on the data tree your installation exposes.

The path shown in the resolution warning below the row confirms that the chosen runs resolve to real data files; a successful resolution is silent.

## 5.2 Fit range

- **From / To –** the time window in microseconds that is actually fit (default `0` to `8`).
- **Packing –** the time-binning factor used for the fit (default `100`).

## 5.3 Fit model

**Type**

The fit type. **Asymmetry** is the normal choice for a summed forward plus backward count rate. **Single hist** fits a single histogram. The legacy **Asymmetry (Legacy)** variant is kept for old MSR files that used the classic syntax but is not offered for new documents.

**Histograms**

The detector histograms, or a packed detector name, that make up the asymmetry signal.

An **Asymmetry** fit normally takes one forward,backward pair (for example `21 25`). Several *groups* of one physical run can be fit at the same time by entering slash-separated forward,backward pairs, for example `1,2/3,4` for two groups (detectors 1 and 2 form the first group, 3 and 4 the second). Each group becomes its own RUN block with its own `alpha` and phase, and the groups scale a shared free amplitude by individual ratios (the first group keeps the fixed identity factor). The parameter table carries a per-group `Alpha`, `Phi`, and ratio; the physical model parameters stay shared. Multi-component models and per-run pairs of a multi-run (global) fit are not eligible for the multi-group route.

**Components**

The theory model is a sum of components. The first component is labelled **Model** and is required; further components are added with + and can be removed with the `x` on their row. Each component has its own parameter set, shown in the **Parameters** tab.

The component list is grouped by physics:

- **Relaxation –** `Asymmetry`, `Exponential`, `Gaussian`, `Stretch`.
- **Static KT –** `SLKT`, `SGKT`, `LGKT`, `STRKT`, `GBGZF`.
- **Oscillating –** `ExponentialCos`, `GaussianCos`, `StretchCos`, `AbragamCos`, `GBGCos`, `ExponentialBessel`, `GaussianBessel`.
- **KT x relaxation –** combined Kubo-Toyabe and relaxation functions, e.g. `LLFExp`, `GLFExp`, `LGKTExp`, `GBGExp`, `STRKTExp` and their spin-flip variants `LLFSExp`, `GLFSExp`, `LGKTSExp`, `GBGSExp`, `STRKTSExp`.
- **Dynamic KT –** `LDKTLF`, `GDKTLF`.

Changing a component rebuilds the parameter table and the constraint preview so their contents follow the new model.

## 5.4 Fit options

- **Minimizer** – Minuit2 strategy variant used by the fit: `MINIMIZE` (default), `MIGRAD`, or `SIMPLEX`.
- **Errors** – the error calculation run after the minimisation: `HESSE` (default), `MINOS`, `HESSE + MINOS`, or `None`.
- **Strategy** – Minuit2 strategy `0`, `1`, `2`, or **Default** (leave unspecified).
- **Objective** – $\chi^2$ (default) or **Max likelihood**.
- **Save** – write the detailed Minuit2 output including covariance information (on by default).

## 5.5 Series fit actions

When two or more independent data sets are fit together **without shared parameters**, the fit follows the independent-series route and the **Fit options** row gains a per-run action chip for every data set. This is how a multi-run series is (re)fitted efficiently without re-fitting runs that have not changed.

Clicking a chip cycles through three actions:

- **reuse** – keep the existing fit result for that run.
- **refit** – rebuild the MSR from the live GUI parameters and fit again.
- **skip** – leave the run untouched.

The **All** dropdown applies one action to every chip at once (**Reuse all / Refit all / Skip all**), which is the convenient way to set a uniform policy and then override single runs by clicking their chip.

A series where every run is skipped reports **Nothing to fit**. Sharing any parameter (the **Shared** column in the parameter table) disables this route and the chips disappear; see *Parameters and parameter plots*. The parameter plot is unaffected and still works.

### 5.5.1 Per-run statistics

For an independent-series fit the status-bar statistic button summarises the series – either **Series did not converge** (some runs failed to converge) or a worst/best aggregate such as $\chi^2$`/NDF worst 1.24 / best 1.01 · 5 runs`. Clicking the button opens a per-run statistics table that lists every data set in run order with:

- the run identifier;
- the action that produced it (**reuse**, **refit**, or **skip**);
- the run's statistic text ($\chi^2$/NDF or max likelihood/NDF, or `Fit did not converge`);
- the converged state;
- the fitted file the value came from.

Reused or skipped runs show the values read from the MSRs already used for plotting. Rows are highlighted for contrast: runs whose value is more than twice the series median stand out as outliers (the spread a worst/best summary might hide), failed runs are marked as not converged, and reused/skipped rows that still carry the on-disk value are visually distinct from freshly fitted ones.

## 5.6 View settings

**Plot**, **FFT**, and **RRF** are one-row disclosure buttons; each expands its settings inline, and only one group is open at a time. The dots next to the buttons mark a group whose settings differ from the defaults.

**Plot**

- **Time range –** independent start and end of the displayed time axis; leaving a field empty selects automatic limits.
- **Y range –** independent displayed Y limits; empty means automatic.
- **View packing –** histogram packaging for the view; empty means automatic.
- **Log X / Log Y –** switch the respective displayed axis to a logarithmic scale.
- **Lifetime correction –** apply the lifetime correction to the displayed spectra (on by default).

**FFT**

The Fourier-transform display settings:

- **Type –** `Power`, `Real`, `Imaginary`, `Real & imaginary`, or `Phase`.
- **Apodization –** `Strong`, `Medium`, `Weak`, or `None`.
- **Phase –** the phase value or a musrfit phase expression, for example `8.5`, `par7`, `par(7,5, 16)`, `parR(...)`.
- **DC correct –** remove the time-domain DC offset before the transform.
- **Range –** the frequency window used for the transform; empty means automatic.
- **Units –** `MHz`, `Gauss` (default), `Tesla`, or `Mc/s`.
- **Power –** the `power` record for the FFT (default `12`).
- **Phase correction –** automatic phase correction `Off`, over the **FFT range**, or over a **Custom range** you type below.

**RRF**

Rotating-reference-frame display settings. They are strictly opt-in: opening the settings does not change the generated plot block.

- **Enable –** add rotating-reference-frame directives to the PLOT block.
- **Frequency –** the reference frequency, with units `MHz`, `kHz`, `Mc/s`, `Gauss` (default variety shown as Gauss), or `Tesla`.
- **Packing –** the packing for the RRF view (required when enabled).
- **Phase –** optional phase in degrees, or a parameter reference such as `par4`.

## 5.7 Fit and Plot

The two toolbar actions work on the current window state:

- **Fit –** validate the run selection, prepare the MSR document from the GUI, and run `musrfit`. The fitted values flow back into the parameter table and the results.

- **Plot** – validate the run selection, prepare the same MSR document, and run `musrview` to display the spectra. On the web the resulting ROOT canvas is drawn inside the page window with JSROOT; on the desktop `musrview` opens its own native window.

While either action runs, the status bar shows progress and the Results tab accumulates the command log. Plot-compatible values are cached so a plot does not lose the numerical fit quality already returned by a previous fit.

On the desktop, a local **Fit** uses every available CPU core by default. Set **musrfit threads** in *Preferences* to cap the OpenMP thread count (`musrfit -u`) when a machine is shared or a fit would otherwise saturate the login node; leaving it blank keeps the all-cores behaviour. The web edition always fits on the server and ignores the setting.

If the MSR editor owns the document (a hand-edited MSR is authoritative), the editor block is used for Fit and Plot as-is and GUI changes do not silently replace it – see *MSR editor*.

# CHAPTER 6

# Parameters and parameter plots

The **Parameters** tab shows the numerical model: one row per fitted parameter, per data set, with starting values, steps, and limits. Fitting feeds the measured values back, and the same table drives the interactive parameter plots.

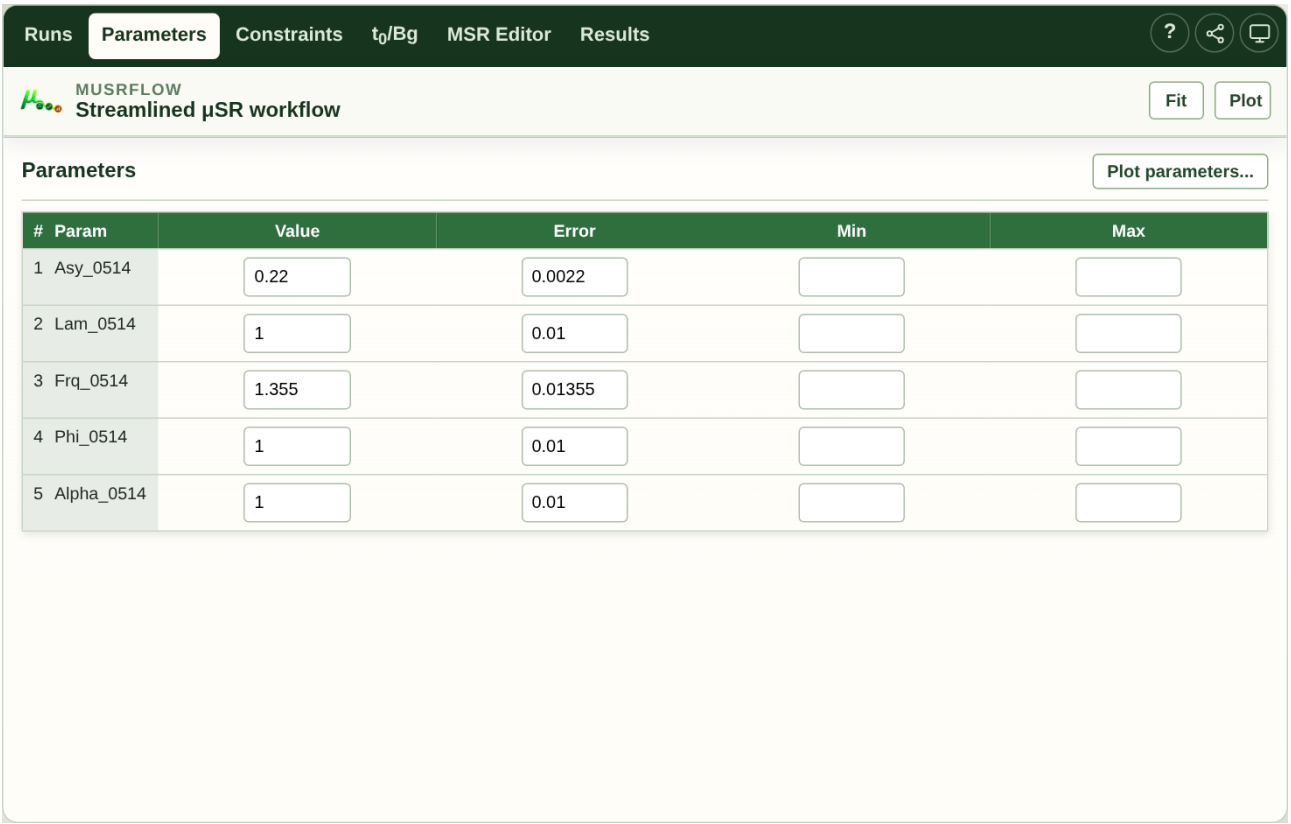



Fig. 1: Example parameter table before fitting, for the same run and model as *Getting started*. These are initial values, not measured results; the Error column contains starting steps at this stage.

## 6.1 The parameter table

The table is rebuilt automatically whenever the model changes (a new component, a new run, a constraint). Rows are keyed to the model:

- Single-run fit: one row per parameter of the selected model.
- Multi-run fit: a row per parameter per data set, each named by its run (for example `Lam_2255`), plus the parameters you choose to share.

Rows follow the physical role of each parameter: the initial asymmetry first, then the relaxation/lineshape parameters, then the precession frequency, then the phase, then the asymmetry alphas and group ratios, and finally the detector normalization/background parameters. This ordering is what ends up in the generated `FITPARAMETER` block and therefore in the `*_par.dat` columns from `msr2data`. A parameter introduced through a constraint (for example `Bfield = gamma_mu * Frq`) is placed with the built-in parameter it is linked to; when it is linked to several parameters it takes the highest-priority (lowest) rank among them. An order imported from an existing MSR file always overrides this physical sequence.

For each parameter the columns are the **starting value**, the **step**, and the **lower/upper limits**. Edit any of them directly. Run-scoped parameters are matched to their data set by the run suffix, so a per-run starting value only affects that run. The values in this table are the starting values used for the next fit; after a fit they are replaced by the fitted results, and the fit uncertainties are available in the parameter plots as error bars.

### 6.1.1 The Shared column

In a multi-run fit the table also has a **Shared** column in the first row of each parameter. Tick it to keep a single shared starting value, step and limits for all runs instead of a separate per-run copy. This is equivalent to the musrfit `GLOBAL` parameter mechanism.

Sharing any parameter takes the fit out of the independent-series route, so the per-run refit/skip action chips disappear (see *Runs and fitting*). The parameter plot still works.

## 6.2 Selecting parameters for plotting

Click a parameter name to select or deselect it for plotting:

- **Click** – toggle that single parameter.
- **Ctrl / Cmd-click** – add an individual parameter without changing the rest of the selection.
- **Shift-click** – select a contiguous range of parameters.

The selection state is shown in the row. Then use the **Plot parameters…** button at the top of the parameter table to open the plot window. If the parameter table is not yet populated (no fit run yet and no file loaded), the plot can still build the parameter data inline from the table – the button stays enabled whenever the table is built.

## 6.3 The Choose X axis dialog

The dialog lists the parameter data columns and asks which one to use as the common X axis. Each selected parameter becomes a Y series.

The dialog also offers the **Layout** of the plot window:

- **Auto –** one parameter per panel. Up to three parameters share a single column; beyond that the panels grow toward a square grid (two parameters in two columns, three in three, and four or more arranged as the closest square grid).
- **Single row –** all parameters side by side in one row.
- **Single column –** all parameters stacked in one column.

The layout choice is remembered for the next plot. The window opens with the chosen arrangement, and the **Plot options** panel lets you re-arrange the panels afterwards.

## 6.4 The parameter-plot window

The window shows one panel per data set / parameter as arranged by the layout choice. Panels share the chosen X axis. The toolbar holds:

- **Refresh –** reread the current parameter-data file and redraw every visible panel. Use this after a new fit to update the numbers.
- **Data… –** change the data set loaded in the *active* panel.
- **Add data set… –** choose one or more parameter-data files and overlay them onto *every* panel at once. Each file keeps one colour from the **Palette**, and every panel plots the parameters it already shows for each newly added file, so a scan can be compared against the current one across the whole window with a distinct colour. The Y-axis label stays on the active panel's parameter rather than reverting to a generic label. Adding files this way is the same overlay mechanism as **Overlay data sets** in *Plot options*; the toolbar shortcut just applies the additions to all panels instead of the active one.

Overlaid curves get a *smart legend* label that says which physical state each scan represents instead of only a file name. The label is derived from the data set's own scan variable and its constant coordinate: an energy scan (`dataE` varying) is tagged with its fixed temperature (`(T = 300 K)`), a temperature scan with its fixed energy for LE-μSR (`(E = 12 keV)`) or with its field for bulk μSR (`(B = 100 G)`), and a field scan with its fixed temperature. When no fixed coordinate is present the plain file label is used. You can also rename a legend entry directly in the plot; a Refresh then keeps your name instead of rebuilding the default.

- **Plot options –** expand the layout, X, parameter, style, and displayed series controls for the active panel.

### 6.4.1 Plot options

- **Layout** – the concrete panel grid (for example `1 x 1` … `3 x 3` and larger grids as the panel count needs). Changing it adds or hides panels; new panels start empty and clone the sync settings of the current panel.
- **Panel indicator** – which panel is active (`Panel 1/3`). Click a panel to activate it before changing its panel-specific settings.
- **Data set** – the data source shown in the active panel.
- **X** – the X column for the active panel.
- **Parameter** – add a Y parameter to the active panel. Serial chips underneath list the current Y series; click the `x` on a chip to remove that series.
- **Style** – `Scatter`, `Lines + markers`, or `Lines`.
- **Palette** – the base colour palette (`Default`, `Bold`, `Pastel`, `Monochrome`) used to assign each overlaid data set its default colour. Changing it restyles every panel.
- **X title / Y title** – override the automatic axis label of the active panel; leave empty to keep the automatic label.

### 6.4.2 Remembered plot configurations

The full window state is remembered per data source: the panel arrangement, the X column, the template, the palette, the per-series styles (including the error-bar caps setting), and the overlay files. A configuration is keyed by the data source *and* the parameter set it plots, so each parameter selection has its own remembered configuration.

Restoring a configuration is always an explicit choice, never something that happens behind your back:

- The **Choose X axis** dialog appears every time you plot parameters.
- If a remembered configuration matches the parameters you selected (the same set, in any order), a **Configuration** dropdown appears in the dialog. The most recent matching configuration is preselected; the dialog shows exactly what will be restored, and you press **Plot** to confirm.
- Choosing **From scratch** in the dropdown (or having no matching configuration) starts a normal new window from the X axis and layout choices on the dialog.

Matching is exact: a saved configuration for three parameters never shows up for a one-parameter request, even if that one parameter is part of the three.

- State is keyed by the data source, so `scanA_par.dat` keeps its own styles and layout while other files (and their overlay sets) remember theirs independently.
- Up to eight data sources are remembered; adding a new one forgets the oldest. Each source keeps at most one configuration per parameter set (the newest wins) and a handful of parameter sets in total.
- **Forget plot configuration** in **Plot options** clears the remembered state for the data source shown in the active panel, so the **Choose X axis** dialog offers no configuration next time.
- The remembered configuration of the current data source is carried in configuration share links, so a colleague restoring your link gets the same plot window – see *Configuration sharing*. When

a real fit produced the parameter table, the table itself is embedded in the link (inline text) and a web restoration renders the identical plot immediately, without needing a local or server-side fit.

- Uploaded files keep only their identity, not their contents: if the original file cannot be read again (for example it was deleted), the restored plot degrades gracefully and the dialog is offered instead.

### 6.4.3 Overlay data sets (multiple files in one panel)

Panels can plot several `*.dat` files at once so scans can be compared on one set of axes. Open **Plot options** and use **Overlay data sets > Add file…** to choose extra parameter-data files (on the desktop this is a native multi-select dialog; on the web it is the browser's multi-file picker). Each file becomes a row with a checkbox:

- checked rows are drawn on top of the panel's main data set;
- the colour swatch shows the default trace colour for that file, assigned from the **Palette**;
- files are identified by their resolved path or blob identity, so picking the same file twice keeps one entry – and one user-adjusted style;
- the X axis is aligned across files **by column name** – a file that does not contain the panel's X column is skipped and named in a striped note under the panel;
- in overlay mode each file's traces share its colour, and legend entries are prefixed with the file name (`scanB_par.dat: λ₁`).

Growing the layout clones the active panel, so expanding a `1 x 1` grid to `2 x 1` carries the overlay files into the new panel. Unchecking a row removes that file from the active panel only; the file stays available in the list for other panels.

### 6.4.4 Worked example: energy scan at two temperatures

See *LEM single-histogram example* for the complete illustrated walkthrough and reproducible results. It uses **new single-histogram fits** of LEM 2025 runs 1014-1027 (blue, 3.8 K) and 1033-1045 (red, 10 K), not the older asymmetry-fit reference tables.

### 6.4.5 Editing trace styles

Click any marker directly in a plot panel to open a small style editor near the pointer:

- **This trace / This data set** – whether the change applies to the clicked (file, parameter) series only, or to every trace of that data set (all panels showing it). Choosing **This data set** unifies the whole file and clears any per-panel trace styling left behind, so those panels no longer shadow the new style.
- colour swatches, marker **symbol**, **size**, and **opacity**;
- **Error-bar caps** – error bars are plain vertical lines unless the checkbox is ticked, which draws the cross-bars at both ends;
- **Apply** – commit the staged choices and redraw the affected panels; **Close** hides the editor, and ESC (or `Q`) dismisses it;
- **Reset** – restore the file's default colouring for that scope.

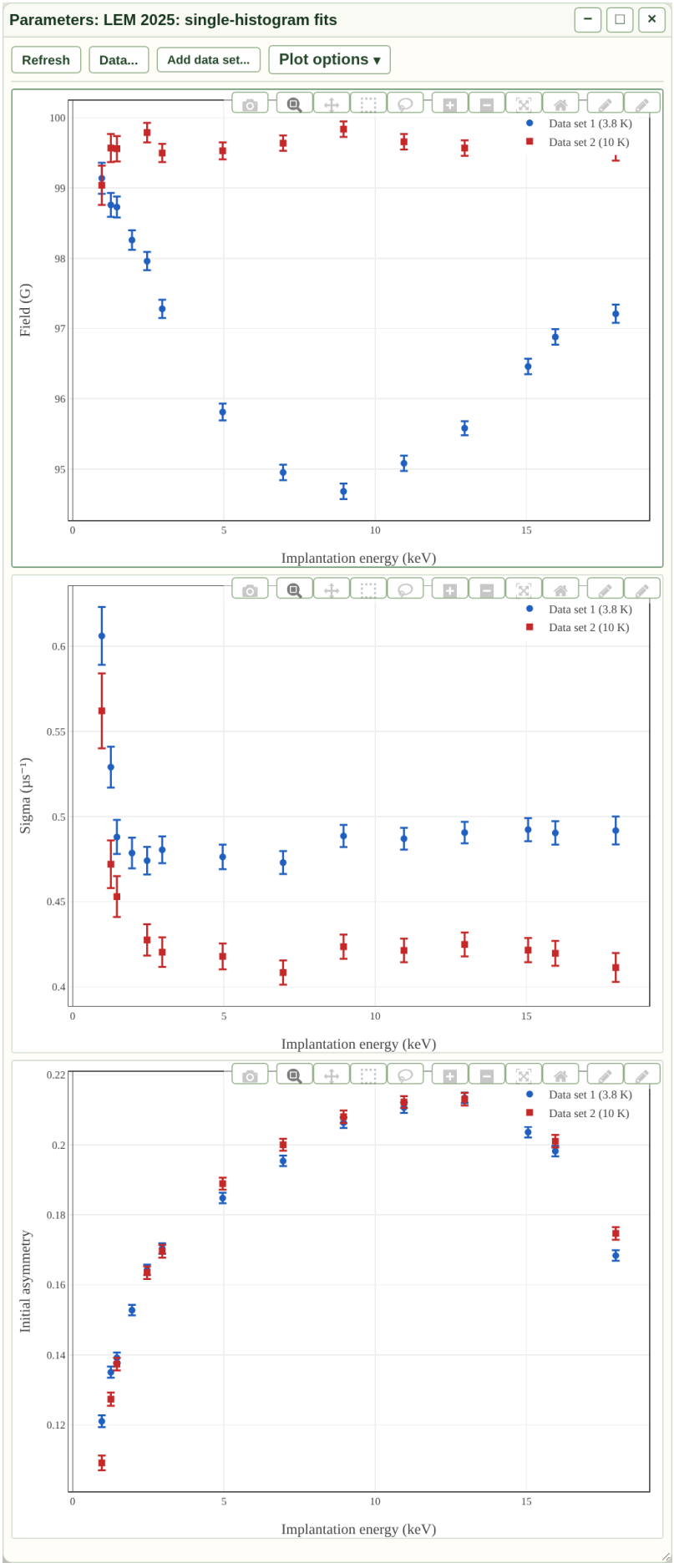



Fig. 2: Three vertically stacked panels from the application, using the newly fitted single-histogram tables. Both data sets are overlaid in each panel. The example chapter explains the model, histogram grouping, uncertainties and limitations; results are in `docs/examples/lem2025-singlehist-energy-scan`.

Trace-level edits are keyed by (data set, panel, parameter), so they survive panel re-layouts but never leak onto other panels. Data-set-level edits are shared: recolour `scanA_par.dat` once, and every panel that shows it updates. `Esc` (or `Q`) closes the editor; the title bar is a drag handle so the panel can be repositioned within the parameter-plot window.

### 6.4.6 Template and series presets

The **Template** dropdown applies a predefined multi-panel arrangement that maps recognised parameter names to roles. For example the `tf-scan` template lays out three panels – amplitude, damping, and frequency – above one another. Parameters that cannot be matched to the data are omitted rather than guessed; unmatched panels stay empty.

### 6.4.7 Keyboard shortcuts

With a panel focused (and not while typing in a field), the shortcut keys are:

- `u` – reset the active panel to its automatic X and Y ranges.
- `Q` or `Esc` – close the plot window.

The plotly mode bar at the top-right of each panel provides zoom, pan, box select, and download, and a custom button toggles each axis between linear and logarithmic. Trace colours are assigned automatically per series – clicking a marker opens the style editor described above to change them.

# CHAPTER 7

# LEM single-histogram example

This worked example repeats the LEM 2025 energy scans using **single-histogram fits**, not asymmetry fits. The original files in `/tmp/lemfit` were asymmetry fits and were used only for initial parameter estimates. They were not overwritten.

## 7.1 Data and model

- **Data set 1:** runs **1014-1027**, 3.8 K, blue circles.
- **Data set 2:** runs **1033-1045**, 10 K, red squares.
- Sample: 85 nm Nb/Al2O3; nominal field 100 G; energy approximately 1-18 keV.
- Fit type: **Single histogram** (`fittype 0`), groups `21 25` and `23 27`; nominal fit interval **0-8 microseconds**, packing **100**.

Both count-histogram groups are fitted together for each physical run. The polarization model is

$$P_j(t) = A_0 \exp[-(\sigma t)^2/2] \cos(2\pi\gamma_\mu B t + \phi_j).$$

The MSR theory is `asymmetry`, `simpleGss`, `TFieldCos` with `fun1 = gamma_mu * par3`, where parameter 3 is `Bfield`. Thus field is fitted directly in G, rather than converted from a plotted frequency later. `Sgm` is in inverse microseconds and `Asy` is dimensionless.

Each group has its own phase, normalization and constant background. `Asy`, `Sgm` and `Bfield` are common to the two groups **within one physical run**. No parameters are shared **between physical runs**: these are 27 independent fits, not one global fit across the energy scan.

**Note**

No `musrt0 -g` correction or explicit t0/data override was applied. musrfit reads the LEM timing information and reports its automatic data range fallback (start t0 + 51 bins, end 66601). The nominal interval is 0-8 microseconds; the fallback excludes approximately the first 10 ns. The retained command logs record these warnings.

## 7.2 Inspect the fitted run

The screenshots below show the actual application after importing the fitted `1014_his.msr`. The fits themselves were executed with the command-line script described below; the screenshots do not imply a browser fit job was executed. Importing this file also restores its recognized GUI settings.

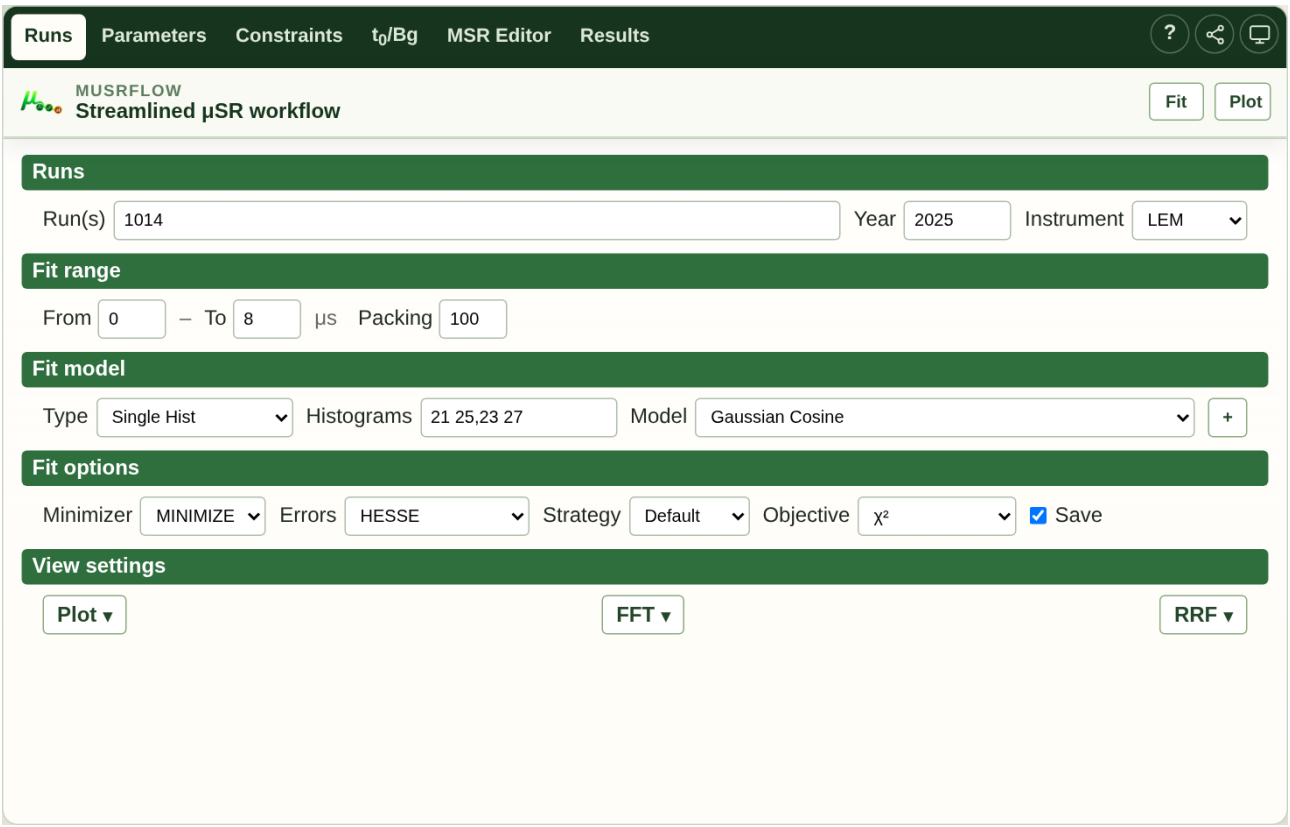



Fig. 1: Representative run 1014. Use this fit type, histogram grouping and model for the two run ranges. Leave cross-run Shared parameters unchecked.

## 7.3 Compare the energy scans

1. Fit each run range independently and export one parameter table per scan. The supplied `dataset1.dat` and `dataset2.dat` contain the new fits.
2. Open **Parameter plots**, load the first table and add the second as an overlay. Select `dataE` (implantation energy in keV) for X.
3. Use a **3 x 1** layout, with `Bfield`, `Sgm` and `Asy` in separate panels. Overlay both data sets in every panel.

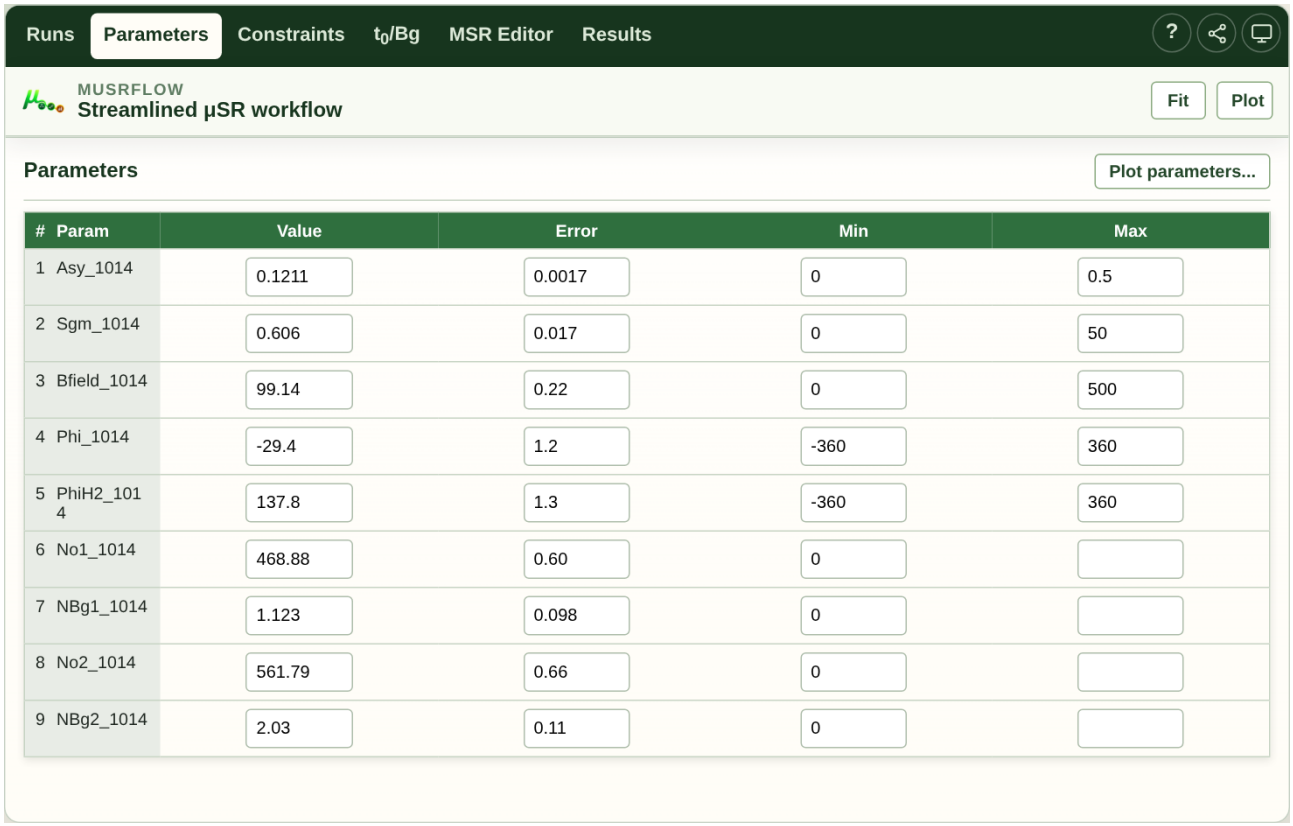



Fig. 2: Fitted parameters imported from run 1014, not illustrative starting values. The two groups have independent nuisance parameters.

4. Set data set 1 blue and data set 2 red using the marker style editor's **This data set** scope. Keep error bars enabled. Resize the plot window to give each panel approximately 4:3 proportions.

All 27 fits produced finite parameters and positive uncertainties for the three plotted quantities. Reduced chi-square ranges from **0.933 to 2.003** (809 degrees of freedom per fit). In particular, the lowest-energy fit is not statistically ideal: successful minimization does not establish that one Gaussian cosine fully describes the data. Inspect residuals and fit range dependence before drawing quantitative physical conclusions.

## 7.4 Reproduce and inspect

The repository directory `docs/examples/lem2025-singlehist-energy-scan` contains all 27 fitted MSRs, their starting-file backups and logs, raw and normalized msr2data tables, metadata, a fit summary and the fit script. Raw detector data are not included. The summary records their SHA256 hashes. The documented run used musrfit 1.12.1 with ROOT 6.38.00.

From the repository root, with musrfit, msr2data and dump_header on PATH:

```
python3 docs/examples/lem2025-singlehist-energy-scan/fit_all.py \
  --data-dir /mnt/data/nemu/his/2025 \
  --work-dir /tmp/lem-singlehist-new \
  --reference-dir /tmp/lemfit
```

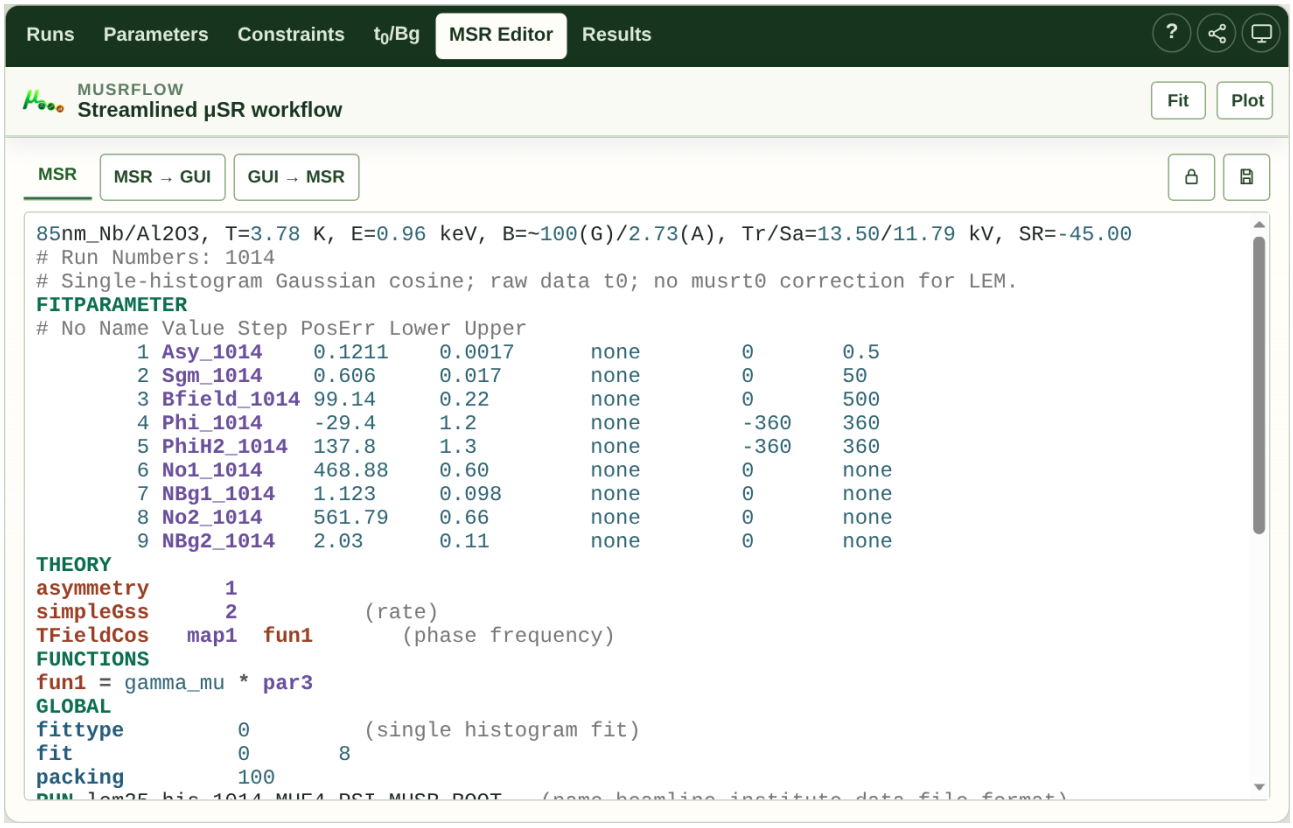



Fig. 3: The fitted MSR is the authoritative record. Editor ownership preserves the imported text rather than regenerating it from an approximate model.

Use your own data directory containing `lem25_his_1014.root` etc. Choose an empty work directory; existing results are not overwritten. The optional reference directory supplies starting values only. Without it, the script uses documented generic initial values, which need not reproduce the same minimum in every case.

The fit commands are `musrfit RUN_his.msr -e -t -u 2 -k`; each MSR contains `MINIMIZE`, `HESSE` and `SAVE`. The export commands use `msr2data "[RUN ...]" _his data new -o datasetN.raw.dat`. Only run suffixes in column headings are removed in `datasetN.dat` so both scans have matching parameter names; numerical values and uncertainties are unchanged.

Download the tables used in the comparison: `data set 1` and `data set 2`.

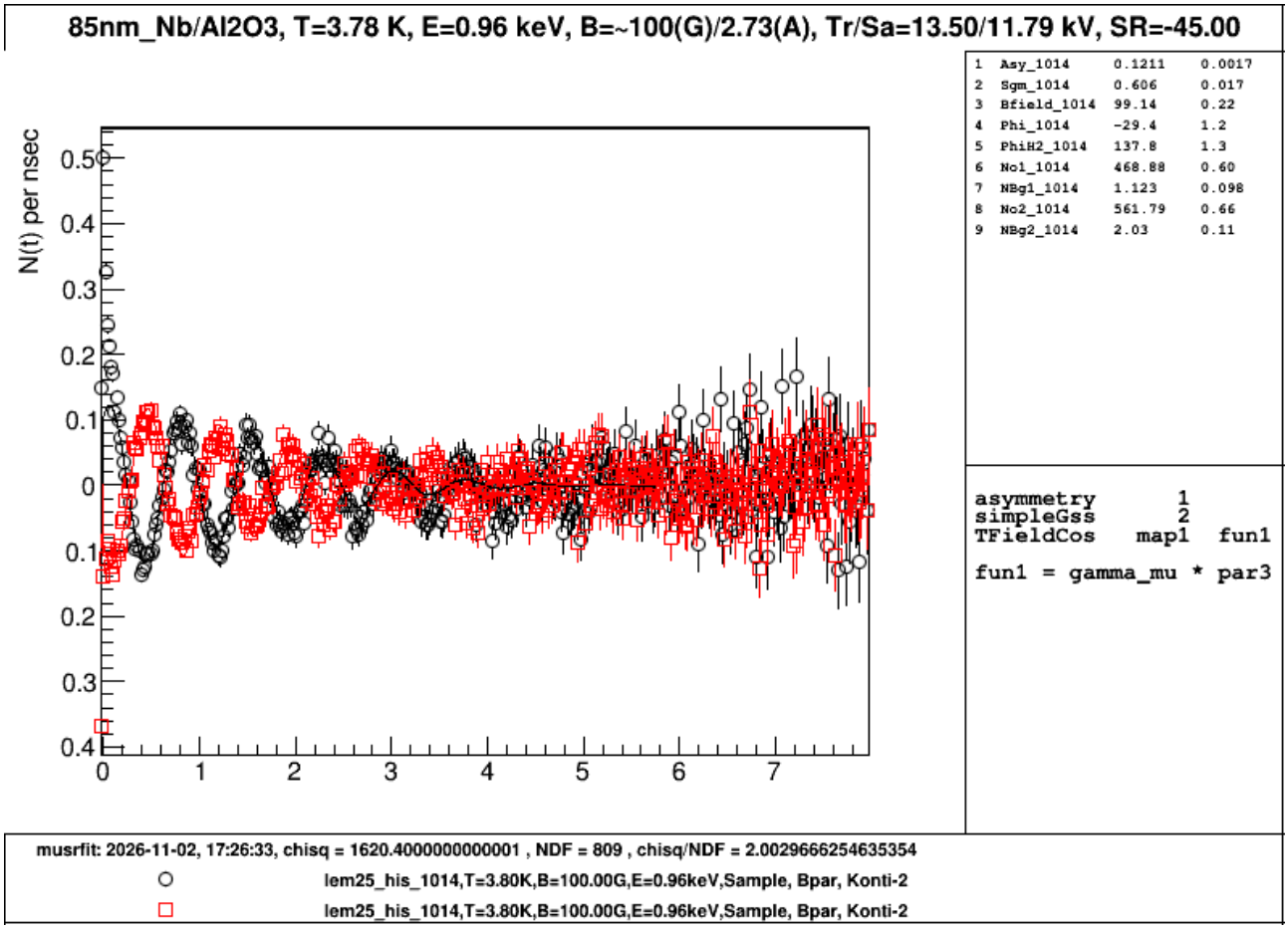



Fig. 4: Actual `musrview 1014_his.msr --png` output. `PLOT 0` contains `runs 1 2` and `lifetimecorrection`; both histogram groups are shown.

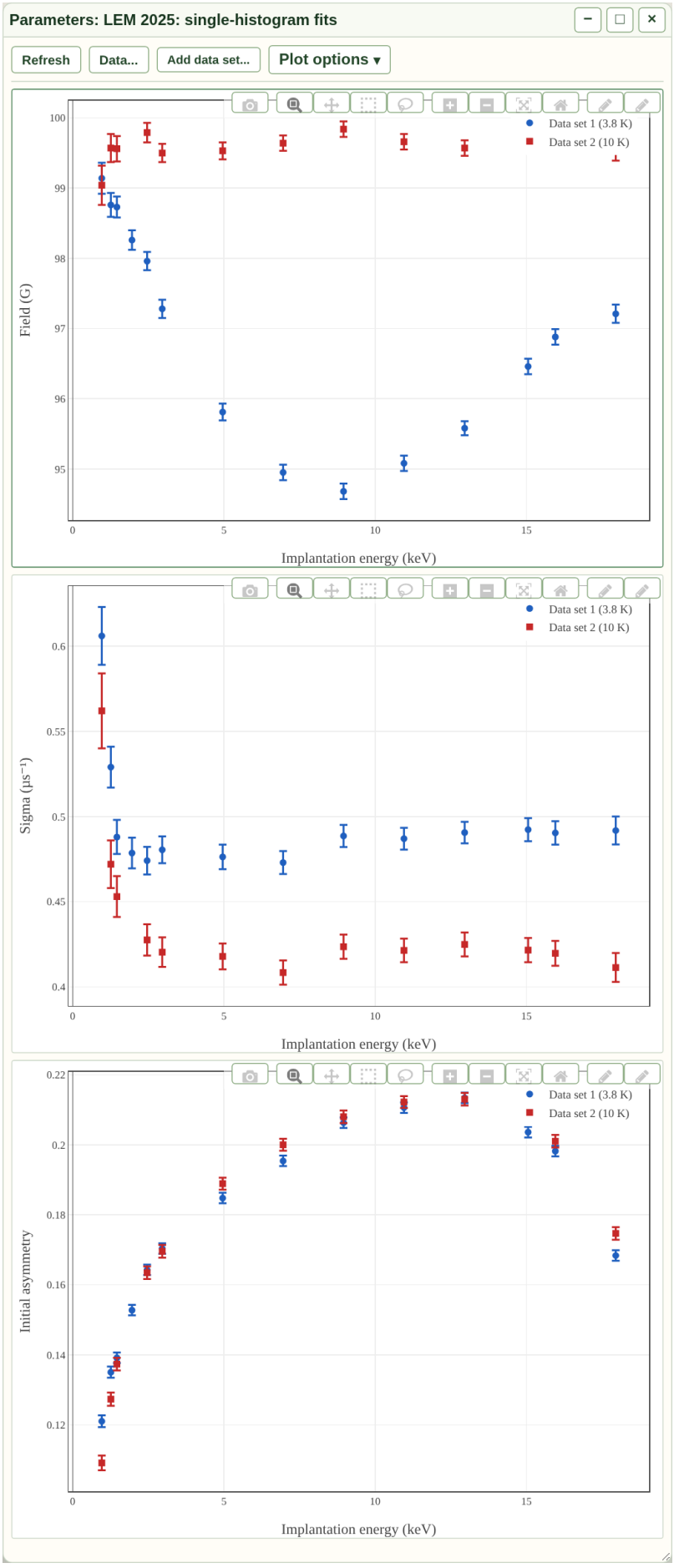



Fig. 5: Actual muSRFlow parameter-plot window, using the newly exported tables. Error bars are the HESSE parameter uncertainties exported by msr2data.

# CHAPTER 8

# Constraints

The **Constraints** tab adds fit functions and parameter relations that are not represented by the main theory controls. It is split into three parts:

- **Custom parameters –** the named constants you define once and reference from constraints.
- **Constraints –** one row per constraint: a parameter, a function expression, and an Add button.
- **Preview –** a live read-only rendering of the `THEORY` and `FUNCTIONS` blocks that the MSR will contain.

Everything on this tab is included the next time the MSR is generated, so you can refine the preview without running musrfit.

## 8.1 Custom parameters

Type a name (for example `Bfield`) and press **Add**. Custom parameters become available on the parameters list and can be referenced by constraints like any ordinary fit parameter. They are removed by the same workflow that removes a normal model parameter.

## 8.2 Constraints

A constraint binds the value of one parameter to an expression of others. To add one:

1. Choose the parameter to constrain from the **Parameter** dropdown.
2. Type the expression into the **Function** field.
3. Press **Add**.

The expression uses musrfit `FUNCTIONS` syntax: arithmetic on parameter names and literals.

## 8.3 Example: couple the magnetic field to the Larmor frequency

A typical use is to avoid fitting a frequency directly and instead fit an applied field:

1. Add a custom parameter named `Bfield` under **Custom parameters**.
2. Under **Constraints**, choose `Frq` from the **Parameter** dropdown.
3. Type `gamma_mu * Bfield` into the **Function** field and press **Add**.

The constraint reads "the Larmor frequency `Frq` equals the muon gyromagnetic ratio multiplied by `Bfield`". The theory and functions preview updates immediately so you can verify the generated `FUNCTIONS` text before fitting.

## 8.4 Preview

The **Theory** and **Functions** panes show exactly what the MSR will contain. The Theory pane reflects the model components from the Runs tab; the Functions pane reflects the constraints. Both are read-only – use the GUI controls to change them.

# CHAPTER 9

# t0 / background

The **t0 / bg** tab hosts expert controls used when detector timing or background behaviour must differ from the normal defaults.

Leave these controls unchanged for standard fits. When you do adjust them, they are written into the generated MSR like any other setting and take effect on the next Fit or Plot.

The tab loads its controls from the current MSR document, so editing an existing document and returning to this tab keeps showing the values that document actually uses.

## 9.1 Multi-group asymmetry fits

For a multi-group asymmetry fit (`1,2/3,4` in the **Histograms** field, see *Runs and fitting*) the groups share the single row of their one physical run, and a field holds one value per group in group order. When importing such an MSR, values that agree across the groups collapse into the tab's common fields and distinct values stay per group (for example `t0 forward 101 103` for two groups). A file whose groups disagree in an inconsistent way - some groups with a `t0` and others without, mixed background modes (a group cannot have both a background range and a fixed background), or a good-bin data range present for only some groups - is rejected without modifying the previously stored values, since a guessing read-back would shift every later group's token.

# CHAPTER 10

# MSR editor

The **MSR Editor** tab displays the complete MSR file that muSRFlow prepares. It is a real editor, not just a preview: you can edit the text by hand, and the edited document becomes the source of truth for the next Fit or Plot.

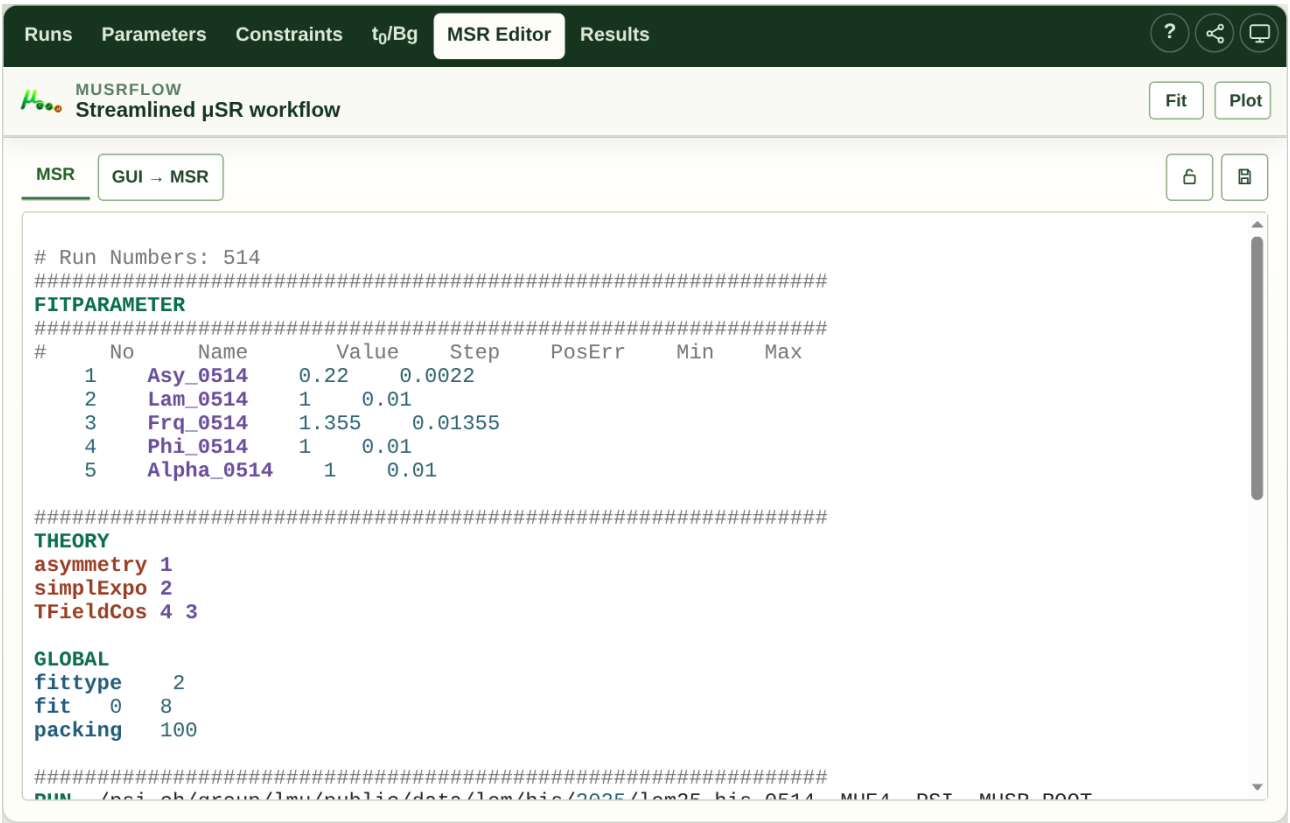



Fig. 1: A document generated with **GUI -> MSR**, before any fit. Syntax colours separate keywords and comments. The open padlock shows that the GUI still owns this document; editing it by hand changes that state.

## 10.1 Ownership: GUI or editor

Two actors can own the document:

- the **GUI**, which regenerates the MSR from the current tab settings; and
- the **editor**, once you edit the text by hand.

The lock icon in the editor toolbar shows which one owns the document. An open padlock means the GUI owns it and the next GUI generation or fit can replace the text; a closed padlock means the editor owns it and Fit and Plot use the text as-is.

## 10.2 Workflow

**GUI to MSR (the `GUI -> MSR` button)**
: Replaces the visible document with one generated fresh from the analysis settings. The GUI is the source of the text.

**Edit by hand**
: Entering text hands ownership to the editor: the lock icon becomes a closed padlock. While locked, **Fit** and **Plot** use the document as-is and GUI changes do not silently replace it. Any attempt that would overwrite the edited text asks for confirmation first.

**MSR to GUI (the `MSR -> GUI` button)**
: Copies the recognised settings of the current document back into the GUI fields (run list, fit range, model, parameters, and constraints). This does not surrender editor ownership; the editor keeps the document until you unlock it.

**Unlock (click the closed padlock)**
: Returns control of the document to the GUI, so the next GUI generation or fit may replace it.

## 10.3 File handling

On the desktop (Electron), the **File** menu opens and saves MSR files directly. The **Save** button in the editor toolbar writes the current document to disk. On the web, saving is handled through the server's result artifacts.

The editor also shows which source document it currently displays. For a multiple-run fit it can expose the generated document and the template it was built from; the active tab (**MSR** or **Template**) selects which one is shown.

For a complete example using real fitted LEM data and application screenshots, see *LEM single-histogram example*.

CHAPTER 11

# Results and diagnostics

The **Results** tab collects what each Fit or Plot produced: the generated files, the execution details, and the full command log. The status bar at the bottom of the main window gives the short-lived state without opening the tab.

## 11.1 Result files

The primary artifact is the **Result MSR** – the fitted document that contains the converged parameter block. Under **Other files** the tab lists:

- **ROOT plot** – the ROOT canvas produced for the spectra display.
- **Template MSR** – the template the document was generated from.
- **Output log** – the full output of the run.
- **Parameter data** – the `par.dat` parameter table that feeds the parameter plots.
- **All files** – a link to the whole artifact package.

On the desktop the links open the files on your machine; on the web they download or display the server-side artifact.

## 11.2 Execution details

A collapsible **Execution details** block records when the action started and finished, the process exit code, and whether the GUI parameter table and the fitted result were synchronised afterwards.

## 11.3 Command log

The command log is the authoritative record: every command line muSRFlow invoked, in order, with its arguments. It appears in the tab and in the pop-up run log explained below. Use it to reproduce an analysis by hand or to diagnose unexpected behaviour.

## 11.4 The status bar

The status bar under the main window keeps a terse permanent status:

- the **state** button (for example `Idle`, `Running musrfit`) – click it to open the run log immediately;
- the **fit statistic** (for example $\chi^2$`/NDF 1.012`), kept from the fit so a later plot does not erase it. For an independent-series fit the button instead shows the worst/best aggregate of the per-run statistics (or `Series did not converge` when any run failed); clicking it opens the per-run statistics table – see *Runs and fitting*;
- the **message** line for transient notices (click it to open the run log);
- the **file** and **directory** the last action operated on.

Hovering over a shortened value shows the full text. The state, statistic, message, file, and directory elements appear only when they add context.

## 11.5 Run log window

The run log dialog shows the same command log as the Results tab, as a large readable pane. It is opened from the status bar state or message, from the outcome dialog, or by clicking the log area in Results. It is modeless, so the main window stays usable while it is open.

## 11.6 Outcome dialog

If an action cannot be completed (a failed fit, an invalid run selection), an outcome dialog explains what went wrong. It offers **Close** and, when relevant, **Open run log** to inspect the commands that ran. A successful action does not pop this dialog – it stays out of the way.

# CHAPTER 12

# Configuration sharing

muSRFlow can encode the current analysis settings into a single link that another instance can restore. This is useful for reproducing a session or for passing a fit setup to a colleague.

## 12.1 What is shared

The link carries the run selection, instrument, year, fit range, model components, fit options, parameters and constraints, plot/FFT/RRF view settings, the analysis-related preferences, and – when a parameter plot configuration was remembered for the current data source – the parameter-plot window state (panel arrangement, palette, trace styles, overlay files). When the sender's parameter data is available on the machine that produces the link (a completed fit), the actual msr2data table is embedded as well, so the recipient can draw the identical plot – fitted values, scan variable, and error bars – without reproducing the sender's `*_par.dat` on its own file system. It does **not** carry the raw data files or your local working directories. A restored session therefore recreates the analysis on top of whatever data the target machine can see.

## 12.2 On the web

The share button in the tab bar (the chain-link icon) copies a link such as

```
http://musruser.psi.ch/muSRFlow/#c=...
```

Anyone who opens the link in a browser gets the same analysis settings, as long as the server exposes the same data.

A visitor who has the desktop build installed can also use the second button next to the share button (the monitor icon). It copies the same payload as a `musrflow://` link instead:

```
musrflow://open/#c=...
```

A browser cannot navigate to its own `https` page and then hand the payload to an installed application, so this dedicated scheme lets the web page open the settings directly in the desktop app. Only the base URL differs; the encoded fragment is byte-for-byte the same, so either link restores the same analysis.

## 12.3 On the desktop

Electron cannot handle its own browser links, so it ships the same encoding backwards as well: open a muSRFlow configuration link (from the web or from a colleague) and choose **Restore configuration link** to paste it into the restore dialog. The analysis settings are applied while your local data and working directories remain untouched.

The installed application also registers the `musrflow://` scheme, so clicking such a link (in a mail client, a document, or the web page above) starts muSRFlow and restores the embedded settings directly. The canonical `http://musruser.psi.ch/muSRFlow/#c=...` link still works: paste it into the restore dialog. See the desktop deployment notes for how the scheme is installed, including the AppImage caveat.

## 12.4 Notes

- The shared payload is a compact, optionally compressed fragment in the URL hash; nothing is sent to a third party.
- A restored configuration may reference data that was never fit on the target machine. The parameter table is rebuilt from the restored settings and can be populated by running a fit.
- The parameter-plot stanza is optional and keyed by the data source's identity. When the sender's parameter table could be read at share time it is carried inline as the main source, so on the web the plot opens immediately after restore (the browser's `Plot parameters...` action applies it the first time it is used). Otherwise it is applied when the same source exists on the target machine – a matching on-disk parameter-data path or an identical inline parameter table (see *Remembered plot configurations*). If neither applies it is ignored harmlessly and the plot opens with the normal dialog.

# CHAPTER 13

# Preferences

The **Preferences** dialog is a desktop (Electron) feature: it opens from the **File** menu. The web edition derives the same settings from server configuration and never exposes this dialog.

**Work directory**
: Where generated files, MSR documents, and result artifacts are written. Defaults to the current directory (`.`), which can be the application installation folder. Choose a separate writable work folder before fitting so application upgrades do not affect your results.

**Data directory**
: The root of the instrument data tree (for example `/psi.ch/group/lmu/public/data`). Run paths are resolved under it.

  When Fit or Plot needs runs that are missing locally, the desktop can download them from `musruser.psi.ch` after confirmation. Downloads go into this directory when the instrument/year folder is writable. Otherwise they go into `run-data` under your personal muSRFlow application data directory. The hint under the Data directory shows that fallback path, and the confirmation lists the actual destination and total size.

  Both locations remain available to subsequent fits, including offline. Cancelling a transfer discards its incomplete file. Your selected Data directory and startup XML are preserved; muSRFlow supplies the additional search paths through `MUSRFULLDATAPATH` when launching its tools.

**musrfit startup XML**
: The `musrfit_startup.xml` used to resolve portable run names. Leave it as **auto-detect** and muSRFlow looks for the standard installation file; choose a specific file when you run a specialised setup. A hint under the field shows the naming mode the detected XML enables.

**musrfit binaries**
: The directory that holds the musrfit tools (`musrfit`, `musrview`, `msr2data`, `musrt0`, and `dump_header`). On Linux, the application packages do not include these binaries (the runtime is distributed separately); leave empty for **auto-detect** and muSRFlow searches, in order,

the runtime shipped with the packaged app, then a separately installed one (`/opt/musrflow/runtime`), then the usual installation locations; choose a directory when the tools live somewhere non-standard (for example a developer build tree). The configured directory wins over every auto-detected location for the tool commands muSRFlow runs, and is put first on the `PATH` given to them. The setting is a desktop feature: it is remembered across sessions, while the web edition takes its tools from the server and never exposes this field.

**Runtime installation and checks (Linux)**

**Check tools** checks the saved tool selection. **Install runtime…** installs a local runtime bundle after confirmation and selects its tools. See *Linux installation* for the required files and installation steps. Windows uses its bundled WSL runtime instead; see *Windows installation*.

**musrfit threads**

The number of OpenMP threads passed to a local `musrfit` run (`-u`). Leave it blank to let musrfit use every available core. Set a lower number to keep a fit from saturating a shared workstation or login node. The value is validated by musrfit (`1 ... number of cores`) and has no effect on the web edition, which runs fits on the server.

**Initial parameters**

**Guess initial parameters from first-run header** (on by default): when a run set is added, muSRFlow reads the first run's metadata to seed sensible starting values (for example an approximate frequency or asymmetry) instead of generic defaults.

**Estimate t0 from the prompt peak (except LEM/HAL)** (on by default): before a local fit, muSRFlow runs `musrt0 --getT0FromPromptPeak` to write explicit `t0` lines into the generated MSR. This forces musrfit to use the histogram peak as the time-zero for instruments whose data-file header t0 is unreliable; the step is never applied to LEM or HAL/HIFI data, regardless of this preference. RUN and ADDRUN entries are also checked so an expert MSR cannot bypass the exclusion through a different GUI selection. Uncheck it to preserve the MSR as generated for other instruments too.

**Output file suffixes**

The suffixes appended to generated artifacts: **Asymmetry** (default `_asy`) and **Single histogram** (default `_singleHist`). Change them to avoid clashing with existing files or to match a naming convention.

Save stores the values for the next session; **Cancel** closes the dialog without changing anything. Directory changes take effect from the next action that resolves or writes files.

CHAPTER 14

# Web and desktop editions

Both editions share the tabs, fit engine, model library, parameter plots, and configuration-sharing protocol. The differences come from where the data and the command-line tools live.

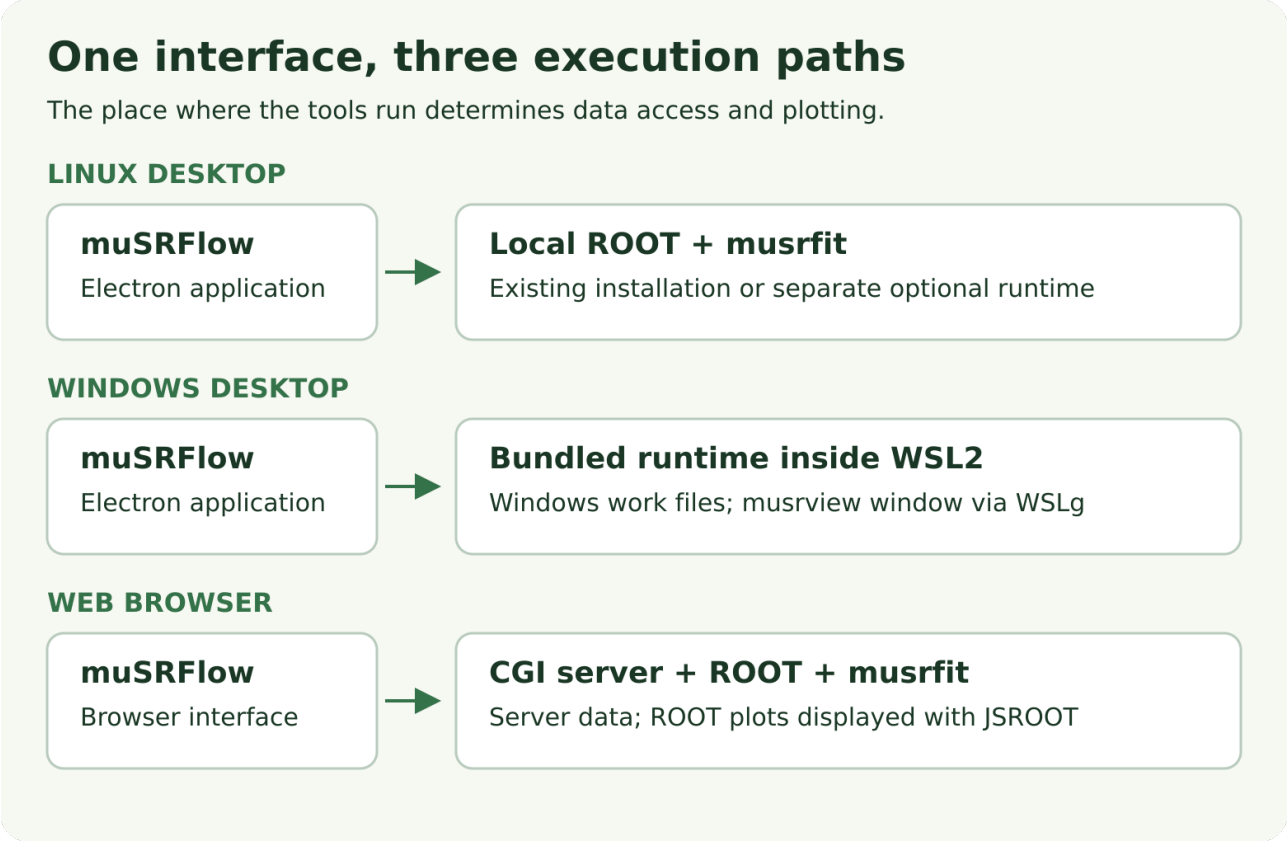



Fig. 1: Standard execution paths. Windows can also use explicitly configured native binaries instead of the bundled WSL runtime.

| Concern | Desktop (Electron) | Web (browser) |
|---|---|---|
| Where the tools run | Locally using configured or detected tools; through WSL2 on Windows. | On the server behind the CGI endpoint. |
| Data access | Local data plus optional downloads of missing selected runs. | What the server data tree exposes. |
| Preferences | Per-user, through **File / Preferences**. | Fixed by server configuration; no dialog. |
| Opening / saving MSR files | Native **File** menu and editor Save button. | Server-side artifacts; no local file dialog. |
| Spectra display | `musrview` in its own native window. | ROOT canvas produced by `musrview` on the server, drawn with JSROOT in the page. |
| Quick Start help | Independent auxiliary window. | Modeless dialog beside the main window. |
| Configuration links | Share webpage links; **Restore configuration link** to paste one. | Share button copies a link; opening it restores the settings. |

## 14.1 Bundled ROOT and musrfit

On Linux, the application and scientific runtime are separate. The Linux application packages contain no musrfit/ROOT binaries—only the muSRFlow frontend. Use an existing ROOT + musrfit installation, install an offline runtime bundle in Preferences, or install the system/runtime RPM or AppImage (containing `musrfit`, `musrt0`, `musrview`, `msr2data`, `dump_header`, `root-config` and bundled libraries). An explicitly selected binaries directory takes precedence over automatic detection. See *Linux installation* for installation and tool checks.

On Windows, the application installer includes the runtime. It runs inside a WSL2 distribution named `muSRFlow-runtime`. Tools are invoked through `wsl.exe`, with paths translated so inputs and outputs stay on the Windows side. The separate Linux runtime installer does not apply to Windows. See *Windows installation* for WSL setup, installation, updates, and fixes.

The workflow chapters of this manual apply to both editions. Where a step is specific to one edition (handling files, Preferences), the text says so explicitly.

## 14.2 Downloading missing runs

Fit and Plot check selected main runs and ADDRUN members before preparing MSR files. If files are missing, the desktop offers to download them over HTTPS from `musruser.psi.ch` and continue. Independent series fits request data only for entries that need fitting; reused results and explicit skips do not trigger downloads. Runs absent from the server are still reported as missing.

The writable Data directory is preferred. When it is read-only, a persistent folder under the user's application data directory is used instead. Exact filenames and formats are remembered across sessions, and incomplete downloads are never published as run files. Expert MSRs with manually specified RUN paths continue to use their existing file-resolution rules.

Generated download RUN paths are relative to the stored data tree. muSRFlow passes the matching search roots to its tools through `MUSRFULLDATAPATH`, including translated paths for Windows/WSL2. To run such an MSR from a terminal outside muSRFlow, set that variable to the download root or add the root to your own musrfit startup XML.

CHAPTER 15

# Keyboard shortcuts

Most of muSRFlow is mouse-driven, but a few shortcuts speed up the everyday tasks.

## 15.1 Parameter selection (Parameters tab)

- **Click a parameter name** – select or deselect it for plotting.
- **Ctrl / Cmd-click** – add just that parameter to the selection.
- **Shift-click** – select every parameter between the anchor and the one clicked (holding Ctrl as well adds the range).

The same actions work from the keyboard: focus a parameter name and press **Enter** or **Space** to toggle it.

## 15.2 Parameter-plot window

With a panel focused (and not while typing in a field):

- `u` – reset the active panel's X and Y axes to automatic ranges.
- `Q` or `Esc` – close the plot window.

## 15.3 MSR editor

The editor is a standard editable document: the usual cursor, selection, and clipboard keys apply. Its ownership state depends on whether the text has been edited by hand; see *MSR editor*.

## 15.4 General notes

- Shortcut keys ignore events typed into input fields, selects, and text areas so they never collide with normal text entry.
- In the parameter-plot window, plotly's mode bar provides the full set of interactive chart tools (zoom, pan, box and lasso select, downloads).

# CHAPTER 16

# Troubleshooting

For Windows setup, WSL startup errors, missing ROOT libraries, and older runtime updates, see *Windows installation*. For Linux package/runtime setup, see *Linux installation*. The sections below cover analysis workflows.

## 16.1 Fit or plot will not start

Check the resolution warning under the run row: it reports runs that cannot be resolved against the current data tree. In the browser edition, confirm the server exposes those runs; on the desktop, verify the data directory preference points at the real data root. If a model has no valid run, the action reports the reason in the outcome dialog and the run log.

## 16.2 “Nothing to fit”

An independent-series fit (two or more data sets, no shared parameters) where every run is set to **skip** reports this instead of running. Set at least one chip to **reuse** or **refit**, or use **Refit all** from the All dropdown.

## 16.3 Fitted parameters look wrong or unchanged

- The parameter table is the source of the *starting* values. After a fit it holds the fitted results; run the fit again after editing values.
- Hand-edited MSR documents are authoritative: if the editor lock is closed, Fit and Plot use the editor text and ignore GUI changes until you unlock (see *MSR editor*).
- A model change rebuilds the parameter table and may reset values; re-check the rows for every component you use.

## 16.4 Parameter plot is empty or missing data

- Click **Refresh** after a new fit – panels reread the parameter file.
- Use **Data…** to change the data set of the active panel.
- A panel that shares the X axis with no plottable column shows its empty message; activate it and add a parameter or load another file.
- Overlaid files that lack the panel's X column (or could not be re-read) are skipped and listed in a striped note under the panel. Fix the X column (**X** control) or uncheck the file under **Plot options > Overlay data sets**.

## 16.5 Parameter plot hangs on "Preparing plot files"

"Preparing plot files" covers the setup performed before the tools run. If the step never finishes, check the server/work-bench side:

- The command-line tools must be reachable on the machine that runs them (local install on the desktop, server-side install for the CGI).
- On the desktop, confirm the work directory is writable.

The command log records every command line that was attempted; it is the first place to look.

## 16.6 Web page cannot load data

The browser edition fetches data and runs fits on the server. If the page loads but actions fail, verify the CGI endpoint is reachable (see `cgi/README.md`) and that the server data root matches what the page was deployed with. The server health endpoint reports the configured data root; muSRFlow's resolution warning should agree with it.

## 16.7 Restored configuration differs from the original machine

Configuration links carry analysis settings only, not data or working directories. On a different machine the same settings resolve against that machine's data tree. If runs do not resolve, fix the data/startup-XML configuration (desktop Preferences; server configuration on the web) and refit.

## 16.8 Some links start with a different prefix

muSRFlow shares both a compact compressed form (`#c=`) and a readable legacy form (`#cfg=`). Both are accepted by the web and the desktop restore dialog.

## 16.9 Reporting problems

Include the command log (open from the status bar or the outcome dialog) and the versions involved. Contact Zaher Salman mailto:zaher.salman@psi.ch with those details for the fastest help.

# CHAPTER 17

# Glossary

**MSR**
: The `musrfit` input document. A text file made of blocks – `FITPARAMETER`, `THEORY`, `FUNCTIONS`, `GLOBAL`, `RUN`, `COMMANDS`, `PLOT`, `FOURIER`, `STATISTIC`. muSRFlow generates, edits, and fits this file.

**fit document**
: The prepared MSR plus the parameters, constraints, and settings that describe one fit. The MSR Editor shows it as text.

**RUN**
: A data set record inside an MSR which names one measured run or a single `ADDRUN`-joined set of runs.

**ADDRUN**
: An MSR directive that appends another run to the current data set, so `514+515` forms one data set from two runs.

**independent-series fit**
: A multi-run fit in which every data set has its own parameters (no `Shared` entries). muSRFlow treats it specially: it can reuse, refit, or skip each run independently through the action chips.

**global parameter**
: A parameter marked **Shared** in the parameter table. One starting value, step, and limits apply to all runs, matching musrfit's `GLOBAL` mechanism.

**par.dat**
: The parameter table produced from a fitted MSR (by `msr2data`). Each line is one run and each column one parameter (plus its uncertainty); the parameter-plot window reads these files.

**musrfit**
: The muon-spin-spectroscopy fitting engine. See the musrfit project pages for its manual.

**musrview**
: The reference viewer for muon-spin data and fits. muSRFlow runs it for the spectra display and for ROOT canvases of fitted data.

**msr2data**
: A companion tool that writes the `par.dat` parameter table from a fitted MSR.

**dump_header**
: A companion tool that prints run metadata. muSRFlow uses it to resolve portable run names and to seed initial parameter guesses.

**ROOT plot**
: A plot canvas produced by `musrview --root`. On the web it is drawn with JSROOT in the browser; on the desktop it is part of the native viewer's output.

**RRF**
: Rotating reference frame. An optional display transform that subtracts a reference frequency from the measured signal; configured in the RRF view settings.

**FFT**
: Fast Fourier transform. The frequency-domain view of the signal, configured in the FFT view settings (type, apodization, phase, range, units, and automatic phase correction).

**configuration link**
: A URL carrying the current analysis settings as a compressed fragment. Both the web share button and the desktop restore dialog use the same format.

If anything in this manual does not match the behaviour you see, contact Zaher Salman.

Separate repository documents cover desktop builds and packaging (`docs/DESKTOP_DEPLOYMENT.md`) and the web-server/CGI deployment (`cgi/README.md`); this manual concentrates on using the application.

# Index


## A

ADDRUN, 61

## C

configuration link, 62

## D

dump_header, 62

## F

FFT, 62
fit document, 61

## G

global parameter, 61

## I

independent-series fit, 61

## M

MSR, 61
msr2data, 62
musrfit, 61
musrview, 62

## P

par.dat, 61

## R

ROOT plot, 62
RRF, 62
RUN, 61