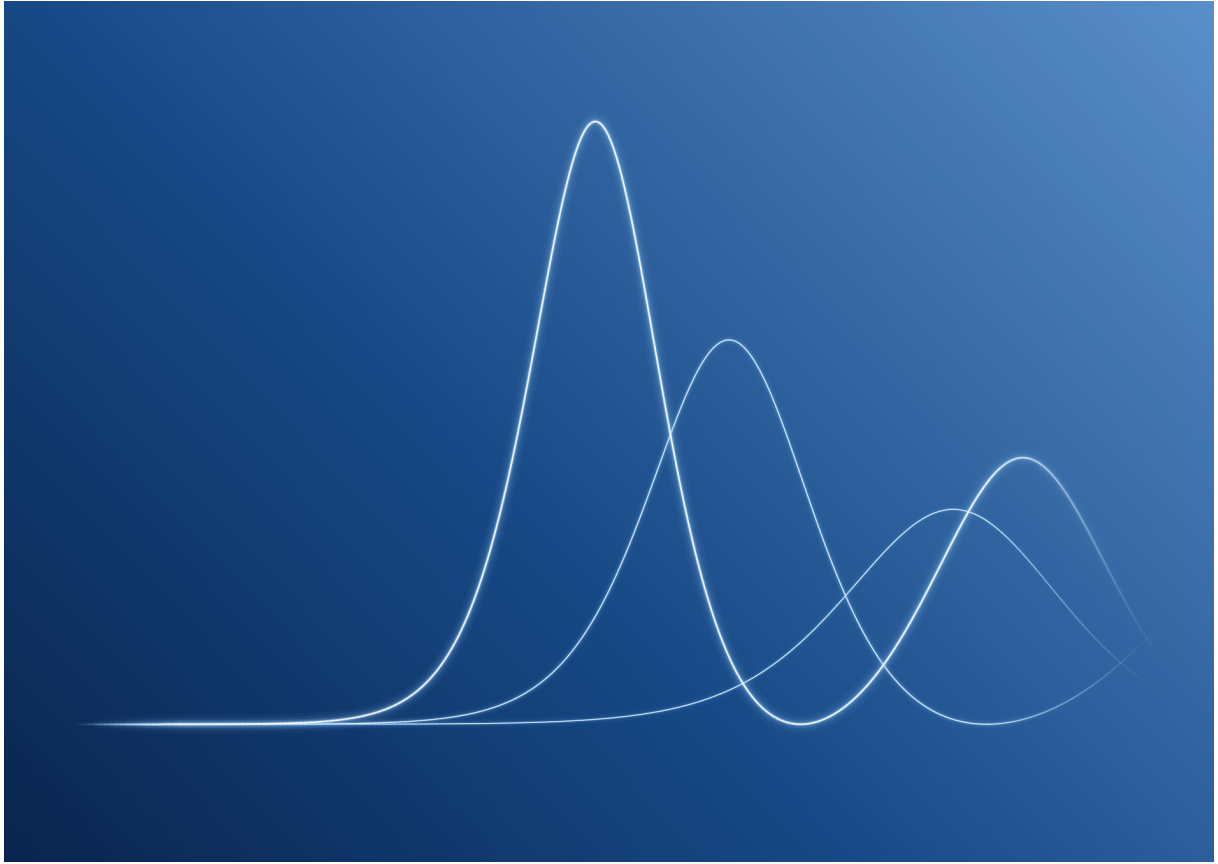




Maxwell Bloch Studio

User Guide

Version 10.8.0

Vahid Anari

vahid.anari8@gmail.com

This document describes the interactive desktop application for solving the Maxwell–Bloch equations, exploring driven coherent radiative regimes, and comparing model results with astronomical light curves.

Contents

1	Overview	3
2	Physical Model	3
2.1	Numerical Method	4
2.2	Input Profiles	4
3	Getting Started	5
3.1	Launching the Application	5
3.2	Command Line	5
3.3	Operating Modes	6
3.4	Main Window Layout	6
4	Parameter Tabs	6
4.1	Sample Tab	7
4.2	Solver Tab	7
4.3	Slice Tab	8
4.4	T ₁ and T ₂ Tabs	8
4.5	ICs Tab	8
4.6	BCs Tab	8
4.7	Pump 1, Pump 2, and Pump 3 Tabs	9
4.8	Fit Mode Tab	9
4.9	Range Tab	9
5	The Plot Area	9
5.1	Top Panel	10
5.2	Bottom Panel	10
5.3	Current Span Marker	10
5.4	Plot Style	10
6	Fit Mode	11
6.1	Loading a Data Folder	11
6.2	Velocity Navigation	12
6.3	Save and Unsave Behaviour	13
6.4	Mismatch Readout	13
6.5	Closing Data	13
7	File Menu Reference	14
8	View Menu Reference	14
9	Info and Help Menus	14
10	Exporting Plots	15
11	Saving and Loading Parameters	15
11.1	Saved JSON Structure	15
11.2	App Defaults	16

12 Using the Solver from Python	16
13 Typical Workflows	16
13.1 Standalone Simulation Workflow	16
13.2 Observational Fitting Workflow	17
14 Troubleshooting	18

1 Overview

Maxwell Bloch Studio is a desktop GUI application for numerically solving the Maxwell–Bloch equations and visualizing the resulting intensity, inversion, pump, boundary-field, and relaxation-time profiles. The app is intended for coherent-radiation problems in which a driven resonant medium is modeled with a one-dimensional Maxwell–Bloch system.

In astrophysical work, the same framework can be used to study several regimes, including:

- steady or slowly varying **maser** amplification,
- transient **superradiant** bursts,
- weakly radiative or suppressed-response regimes that may be interpreted as **subradiant** behaviour depending on the chosen initial conditions and relaxation times.

The application provides:

- interactive parameter editing with sliders, line edits, and combo boxes,
- automatic re-solving when physical solver parameters change,
- two operating modes: **standalone simulation** and **fit mode**,
- overlay of simulation results with observational light curves,
- JSON-based parameter/configuration saving,
- export of the current figure to PDF, PNG, or SVG,
- a PDF export report containing metadata and the current parameter state.

Note

The app is designed around fast interactive exploration. In normal use, there is no separate “Run” button: changing a solver parameter triggers a new solve automatically.

2 Physical Model

The solver evolves a dimensionless one-dimensional Maxwell–Bloch system for a driven two-level medium. The equations solved by the application are

$$\frac{\partial w}{\partial \tau'} = A' R - \frac{w - w_0}{T_1'} + \Lambda_n', \quad (1)$$

$$\frac{\partial R}{\partial \tau'} = -A' w - \frac{R - R_0}{T_2'}, \quad (2)$$

$$\frac{\partial A'}{\partial z'} = -\eta R, \quad (3)$$

where w is the population inversion, R is the medium coherence, and A' is the dimensionless field envelope. The solver works in normalised variables, with $\tau' = \tau/T_0$ and $z' = z/L$ a propagation coordinate normalised to the sample length L .

Symbol	Meaning
$w(\tau', z')$	Population inversion $\rho_{22} - \rho_{11}$
$R(\tau', z')$	Coherence envelope ($\rho_{21} = \frac{1}{2}R e^{i(kz - \omega t)}$)
$A'(\tau', z')$	Dimensionless field envelope
$\tau' = \tau/T_0$	Normalised retarded time
$z' = z/L$	Normalised propagation distance
$T'_1 = T_1/T_0$	Longitudinal relaxation time (may vary with τ')
$T'_2 = T_2/T_0$	Transverse dephasing time (may vary with τ')
$\eta = T_0/T_R$	Dimensionless coupling parameter
$\Lambda'_n = T_0\Lambda_n$	Dimensionless pump rate
w_0, R_0	Equilibrium inversion and coherence

The physical reference time T_0 is set in the **Sample** tab. By default, $T_0 = 86400$ s (one day), but any positive value may be used. The derivation of these equations from first principles is given in the companion Equations Reference document.

2.1 Numerical Method

The application solves the system numerically on a time-space grid. The matter variables are advanced in time with a fourth-order Runge-Kutta scheme, while the field is propagated through the spatial grid. Performance-critical loops are accelerated with Numba.

2.2 Input Profiles

The boundary field, the pump, and the relaxation times are entered as time profiles. Two profile types are used.

Pulse profiles (boundary field and pump)

The boundary field $A'|_{z'=0}$ and each pump contribution are sums of N sech^2 pulses,

$$f(\tau') = \sum_{i=0}^{N-1} a^i \text{sech}^2\left(\frac{\tau' - \tau_0^i}{w^i}\right), \quad (4)$$

with amplitude a^i , center τ_0^i , and width w^i .

Step profiles (relaxation times)

Each of T'_1 and T'_2 is either a constant or a sequence of smoothed steps between N levels $y^0, \dots, y^{N-1}$,

$$y(\tau') = y^0 + \sum_{i=1}^{N-1} \frac{y^i - y^{i-1}}{2} \left[1 + \tanh\left(\frac{\tau' - \tau_0^i}{w^i}\right) \right]. \quad (5)$$

Step i switches from y^{i-1} to y^i around τ_0^i , over a time of order $2w^i$. With $N = 1$ the profile is the constant y^0 .

Common options

In asymmetric mode, both profile types use separate widths on each side of the center,

$$w^i = \begin{cases} w_l^i, & \tau' < \tau_0^i, \\ w_r^i, & \tau' > \tau_0^i. \end{cases} \quad (6)$$

An optional spacing control places the centers at regular intervals of τ_p :

$$\tau_0^i = \tau_0^0 + i \tau_p \quad (\text{pulses}), \quad \tau_0^i = \tau_0^1 + (i - 1) \tau_p \quad (\text{steps}). \quad (7)$$

This is useful for repeated pulses or regularly spaced changes.

3 Getting Started

3.1 Launching the Application

Install the application once with

```
pipx install maxwell-bloch-studio
```

(or `pip install maxwell-bloch-studio` inside a conda or virtual environment), then start it with

```
maxwell-bloch-studio
```

or the shorter `mbstudio`. To start it with a double-click instead, run once

```
mbstudio install-app
```

which adds Maxwell Bloch Studio to the Applications folder on macOS (and so to Launchpad and Spotlight; drag it to the Dock to keep it there), to the Start menu on Windows, or to the applications menu on Linux (Section 3.2). Until version 9 the application was called Maxwell Bloch Solver; saved settings and parameter files carry over. A splash screen appears during startup, followed by the main window.

3.2 Command Line

The `mbstudio` command also works from a terminal:

Command	Effect
<code>mbstudio</code>	start the application
<code>mbstudio PATH</code>	start it and open a parameter file or a data folder
<code>mbstudio --version</code>	print the versions of the app, solver, widgets and Python
<code>mbstudio update --check</code>	show whether a newer version exists, and what is new
<code>mbstudio update</code>	update the app (asks first)
<code>mbstudio solve FILE</code>	solve a parameter file without the window
<code>mbstudio example [PATH]</code>	save the example script of Section 12
<code>mbstudio install-app</code>	start the app with a double-click (see below)
<code>mbstudio --help</code>	list all of this

`mbstudio` `PATH` opens a folder as a data folder (Section 6); a parameter file saved with a data folder opens that folder, and any other parameter file is loaded as with **File** → **Open Parameters**....

`mbstudio` `update` finds out how the app was installed and runs `pipx upgrade maxwell-bloch-studio` or `pip install -U maxwell-bloch-studio` in the same Python; add `--yes` to skip the question. A copy installed from a source folder (`pip install -e`) is updated there with `git pull` instead.

`mbstudio` `solve` `FILE` computes the same numbers as the app and saves them to `FILE_results.npz` next to the file (`-o NAME.npz` or `--output-dir DIR` to choose). The file holds the arrays `time`, `intensity`, `w`, `R`, `lambda_n`, `A0`, `t1`, `t2` (as in Section 12) and the parameters as JSON text, `params`; read it with `numpy.load`. For the parameter file of a data folder, choose a velocity with `--velocity 12.3`, or solve every velocity with `--all`, one file each. This runs on machines without a display, such as a computing cluster.

`mbstudio` `install-app` makes a launcher: `Maxwell Bloch Studio.app` in `~/Applications` on macOS, a Start-menu shortcut on Windows, and an entry in the applications menu on Linux. It starts the installation it was made from, so it keeps working after `mbstudio` `update`, and the app refreshes it by itself on the first start after an update; run it again only if you install the app in another Python. `mbstudio` `install-app --remove` removes it.

3.3 Operating Modes

The application has two main modes.

1. **Standalone simulation mode:** used when no data folder is loaded. The app solves the Maxwell–Bloch system directly from the current parameters.
2. **Fit mode:** entered after loading a light-curve folder. In this mode, the top panel overlays observational data and the model curve for one selected velocity component.

3.4 Main Window Layout

The main window is divided into three user-facing areas.

- **Top toolbar (fit mode only):** velocity navigation, save/unsave controls, an “Only saved velocities” filter, and a χ^2 -style mismatch readout.
- **Central plot area:** two vertically stacked panels that share the time axis. The top panel shows intensity or flux; the bottom panel shows one selected auxiliary curve.
- **Parameter tabs:** organized controls for the sample, solver grid, initial conditions, driving terms, fit alignment, and display ranges.

The **Bottom Plot** selectors appear beside the velocity controls: **Left** chooses which auxiliary quantity is shown on the lower panel’s left axis, and **Right** chooses between the population inversion w , the coherence R and None on its right axis. Both show each curve’s colour.

Figure 1 shows the window in standalone mode.

4 Parameter Tabs

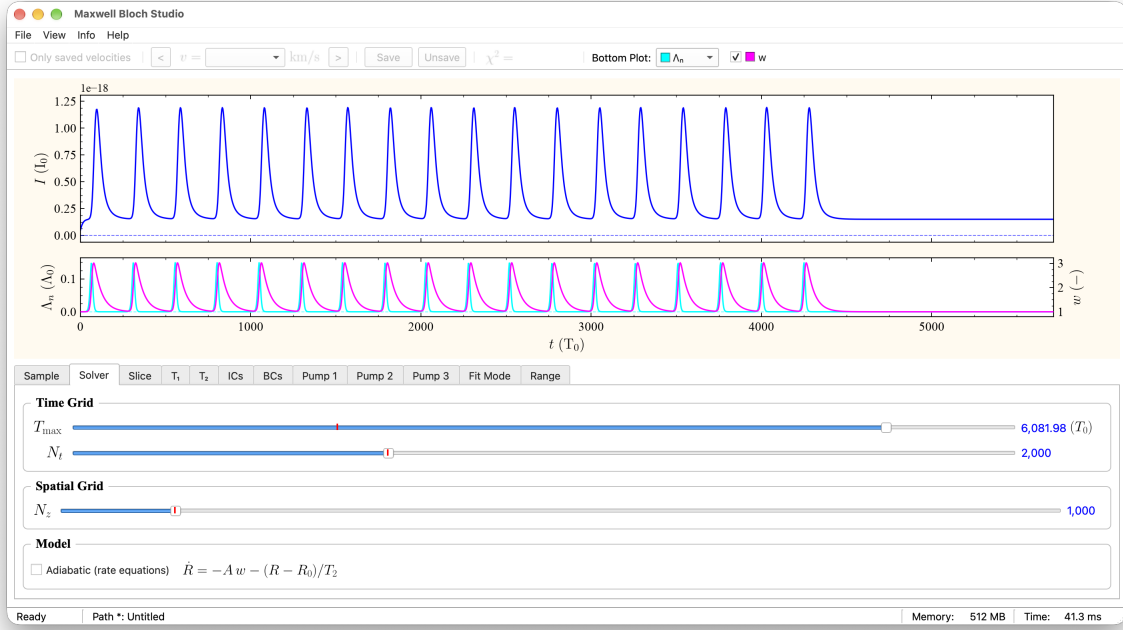


Figure 1: The main window in standalone mode: plot area (top) and parameter tabs (bottom), here the **Solver** tab with the grids and the Model option. The fit-mode toolbar is inactive.

4.1 Sample Tab

The **Sample** tab defines the global physical scaling:

Parameter	Symbol	Description
Transition frequency	ν	Line frequency
Spontaneous emission rate	Γ	Einstein- <i>A</i> -type rate
Time scale	T_0	Physical reference time
Sample length	L	Physical propagation length
Number density	n_0	Medium number density

From these values, the app derives internal scaling quantities such as the coupling constant η . The expression used by the app can be viewed from **Info** → **Scaling Formula**.

4.2 Solver Tab

The **Solver** tab controls the numerical resolution:

Parameter	Symbol	Description
Time points	N_t	Number of time-grid samples
Spatial points	N_z	Number of propagation steps
Total time span	$T_{\max}$	Simulation duration in units of T_0

Increasing N_t or N_z generally improves resolution but also increases solve time.

Model. Adiabatic (rate equations) replaces the coherence equation by its steady state, $R = R_0 - T_2 A w$, so only w and A are solved:

$$\frac{\partial w}{\partial \tau'} = A R_0 - T_2 A^2 w - \frac{w - w_0}{T_1} + \Lambda_n, \quad \frac{\partial A}{\partial z'} = -\eta R_0 + \eta T_2 A w.$$

The approximation holds when T_2 is short compared with the time over which A and w change. It needs a finite T_2 : setting $T_2 = \infty$ with the option on (or ticking the option while $T_2 = \infty$) is refused with a message, and the changed setting goes back to its previous value. The derivation and the numerical scheme are in the section *Adiabatic (Rate-Equation) Limit* of the Equations Reference.

4.3 Slice Tab

The **Slice** tab selects which spatial position is shown in the plots. The displayed solution is taken from one sampled z plane within the interval $[0, 1]$. Changing only the selected slice updates the displayed curve immediately. If the underlying slice array length is changed, the app performs a new solve.

4.4 T_1 and T_2 Tabs

The **T_1** and **T_2** tabs set the longitudinal relaxation time T_1 and the transverse dephasing time T_2 , in units of T_0 . Each uses a step-profile editor (Section 2.2):

- $N = 1$: the relaxation time is a single constant value,
- $N > 1$: the relaxation time changes between N levels through smoothed steps, each with its own center τ_0^i and width w^i (or w_l^i and w_r^i in asymmetric mode).

With $N = 1$ only the level slider is shown; the step controls appear when N is increased. Index $i = 0$ is the initial level and has no step, so its center and width controls are disabled.

4.5 ICs Tab

The **ICs** tab defines the initial state of the medium. Two input styles are available:

- **θ_0 mode:** the initial state is set through a Bloch-angle parameter.
- **Direct mode:** the initial inversion and coherence are entered directly as $w|_{\tau=0}$ and $R|_{\tau=0}$.

This tab is especially important when exploring the transition between weak response, maser-like response, and superradiant burst behaviour.

4.6 BCs Tab

The **BCs** tab defines the boundary field $A|_{z=0}(\tau)$ as a sum of sech^2 pulses (Section 2.2). Unlike a single-pulse editor, this widget supports a **multi-component** profile:

- N : number of components,
- i : currently edited component index,
- component amplitude A^i ,
- center τ_0^i ,
- either a symmetric width w^i or asymmetric widths w_l^i and w_r^i ,

- optional regular spacing via τ_p .

Additional checkboxes allow the user to force all components to share the same amplitude and/or widths. The **Show values** button opens a table of all component values.

4.7 Pump 1, Pump 2, and Pump 3 Tabs

The **Pump 1**, **Pump 2**, and **Pump 3** tabs define three independent contributions to the total pump,

$$\Lambda_n(\tau) = \Lambda_n^{(1)}(\tau) + \Lambda_n^{(2)}(\tau) + \Lambda_n^{(3)}(\tau). \quad (8)$$

Each tab uses the same pulse-profile editor as the **BCs** tab. This makes it easy to build single-pulse, repeated-pulse, or multi-episode pumping scenarios. A tab whose amplitudes are all zero contributes nothing.

4.8 Fit Mode Tab

The **Fit Mode** tab is enabled only when a data folder is loaded. It does not change the physical solve itself. Instead, it controls the conversion, alignment, and scaling used when the model is displayed against data.

Control	Description
Data time unit	Time unit used for the imported data axis (d, h, m, s, ms, μ s, ns)
Data flux unit	Flux unit for the imported data (currently Jy)
Data time offset	Constant shift applied to the loaded data times
Results time offset	Horizontal shift applied to the model curve
Intensity scale	Vertical scale applied to the displayed model intensity

Note

In fit mode, the displayed model curve is rescaled for plotting. The time axis is converted using the chosen data unit, and the intensity is peak-normalized internally before the user-selected display scale is applied.

4.9 Range Tab

The **Range** tab controls the visible axis limits for:

- time,
- top-panel flux/intensity,
- w and R ,
- Λ_n ,
- $A|_{z=0}$,
- T_1 ,
- T_2 .

Each quantity uses a two-handle range slider. The sliders of the bottom panel are shown only for the curves currently selected there.

5 The Plot Area

5.1 Top Panel

The top panel shows:

- **standalone mode:** model intensity $I(t)$,
- **fit mode:** observational flux points together with the transformed model curve.

In fit mode, the plotted data points come from the loaded light-curve file for the currently selected velocity.

5.2 Bottom Panel

The bottom panel displays one auxiliary quantity chosen with the **Left** selector:

Option	Displayed quantity
None	No curve (left axis numbers hidden)
Λ_n	Total pump profile
$A _{z=0}$	Boundary field $A _{z=0}$
T_1	Longitudinal relaxation time $T_1(\tau)$
T_2	Transverse dephasing time $T_2(\tau)$

The **Right** selector draws the population inversion w or the coherence R at the selected z slice on a separate right axis of the same panel (or None), independently of the left one. Only one of them is shown at a time: R is usually much smaller than w (it starts at $\sin \theta_0$), so on a shared axis it would look like a flat line.

5.3 Current Span Marker

When a BC, pump, T_1 , or T_2 tab is active, the app can highlight the currently selected component: a vertical line marks its center τ_0^i and a shaded region indicates its width. For pulse profiles this is the selected pulse; for step profiles it is the selected step. Its colours and edge can be changed from the plot's right-click menu (Section 5.4).

The marker is shown only when it matches the quantity in the bottom panel:

- BC marker with bottom plot $A|_{z=0}$,
- pump marker with bottom plot Λ_n ,
- T_1 marker with bottom plot T_1 ,
- T_2 marker with bottom plot T_2 .

No marker is shown when T_1 or T_2 is a constant ($N = 1$).

5.4 Plot Style

Right-click a panel to change how it looks:

- **Style of ...:** shown first when you click on a curve; edits that curve.
- **Curves:** every curve of the panel, each with its colour; **Style...** sets colour, line style, width, marker and marker size.
- **Grid:** horizontal and vertical grid lines, major and minor, for the panel and its right axis. **Grid style...** sets line style, width and opacity for all grid lines of the figure; grid colours follow the curves automatically.

- **Current span style...**: the current span marker (Section 5.3): fill colour and opacity, edge colour, opacity, width and line (None for no edge) of the shaded area, and colour, opacity, width and line of the centre line.
- **Reset this panel**: returns the panel's curves, grids and span to their defaults.

In every style dialog each change shows on the plot at once; **Cancel** puts the old style back. These choices are saved between sessions and are also used when exporting the plot. **View** → **Reset Plot Style** returns the whole figure to its defaults.



Figure 2: Right-clicking the bottom panel: **Curves** lists each curve with its colour, each with a **Style...** entry.

6 Fit Mode

Fit mode is used to compare the model with observed light curves one velocity component at a time. Figure 3 shows a data folder loaded: the toolbar is active, the top panel shows the observed flux with the model, and the time axis is in days.

6.1 Loading a Data Folder

To enter fit mode:

1. choose **File** → **Open Data Folder...** (Ctrl+O),
2. select the folder containing the data files,
3. if more than one period group is present, choose the desired period,
4. review the import summary dialog.

If matching saved parameters already exist in the folder, they are loaded automatically.

Expected File Naming

The application expects filenames of the form

SourceName_v=12.3.txt

SourceName_p1_v=12.3.txt

where the optional token p1, p2, ... is a period label and v= denotes the velocity.



Figure 3: Fit mode with a data folder loaded. The selected velocity component is saved (✓), and χ^2 shows the mismatch for it; the **Fit Mode** tab aligns and scales the model to the data.

Folder Rules

A valid data folder should satisfy the following conditions:

- all valid files belong to the same source,
- files are either all period-tagged or all non-period-tagged,
- each light-curve file is plain text,
- the app reads the **first two numeric columns** as time and flux,
- blank lines and lines beginning with # are ignored.

Important

File → Open Parameters... is intended for standalone parameter snapshots, not for fit-mode source files. Fit-mode parameter sets are loaded automatically from the selected data folder.

6.2 Velocity Navigation

Once a data folder is open, the velocity toolbar becomes active. You can:

- move to the previous or next component,
- choose a component directly from the drop-down list,
- filter the list with **Only saved velocities**,
- save or unsave the current component.

Each velocity component has its own stored parameter state.

6.3 Save and Unsave Behaviour

In fit mode, pressing **Save** or using **Ctrl+S** saves the current velocity's parameters into a single source-level JSON file in the loaded folder:

`SourceName_params.json`

`SourceName_p1_params.json`

The file stores the saved states for all velocities in the selected source/period group.

The **Unsave** button removes the saved state of the current velocity from that file.

If a velocity has unsaved changes and you try to switch away from it, the app asks whether you want to save, discard, or cancel the change.

6.4 Mismatch Readout

The toolbar shows a quantity labeled χ^2 . Within the app, this is a **reduced χ^2 -like mismatch** computed over the currently visible data range. It is useful as a quick comparison metric, but it is **not** a formal statistical reduced chi-square based on measurement uncertainties.

6.5 Closing Data

Choose **File** → **Close Data** (**Ctrl+W**) to leave fit mode and return to the default standalone state. If there are unsaved changes, the app prompts before closing the folder.

7 File Menu Reference

The shortcuts below use Windows/Linux conventions. On macOS, replace **Ctrl** with **Cmd** and **Alt** with **Option**.

Menu Item	Shortcut	Description
Open Data Folder...	Ctrl+O	Import a data folder and enter fit mode
Recent Data Folders	—	Reopen a recently used source folder
Open Current Source Folder	Ctrl+Alt+O	Open the active source folder in the file browser
Open Parameters...	Ctrl+Shift+P	Import a standalone parameter snapshot
Close Data	Ctrl+W	Close the loaded data folder
Save Parameters	Ctrl+S	Save current parameters
Save Parameters As...	Ctrl+Shift+S	Save a standalone snapshot to a new file
Save Parameters as Default	Ctrl+Alt+S	Store the current state as the startup default
Export Plot...	Ctrl+E	Export the current figure
Quit	Ctrl+Q	Exit the application

Note

In fit mode, **Save Parameters As...** is disabled. The active save target is the source-level parameter file inside the loaded data folder.

8 View Menu Reference

Menu Item	Shortcut	Description
Show Slider Range	Ctrl+Alt+R	Show or hide slider range labels
Reset Plot Style	—	Return curve styles, grids, grid style and span style to their default

Grid lines, curve colours, line styles and the span style are set from the plot's right-click menu (Section 5.4).

9 Info and Help Menus

Menu Item	Shortcut	Description
<i>Info</i>		
Data Summary	Ctrl+I	Show source, period, velocity, and data ranges
Scaling Formula	Ctrl+Alt+F	Show the scaling expression used by the app
Solver Step Viewer	Ctrl+Alt+V	Open the interactive solver-step viewer
<i>Help</i>		
User Guide	F1 (Cmd+? on macOS)	Open this guide
Equations Reference	Ctrl+Alt+E	Open the bundled equations reference PDF
Use from Python	—	Show the example script for your own scripts
Check for Updates	Ctrl+U	Ask PyPI whether a newer version exists
About Maxwell Bloch Studio	—	Version, links, and the libraries the app is built with

The app also checks for a newer version a few seconds after it starts. When one exists,

a notice shows how to update and, below it, **What's new**: the release notes of every newer version, newest first. **Open Release Notes** opens all notes on GitHub. Without an internet connection the check is skipped silently.

Update Now installs the new version while the app stays open, with the same command as `mbstudio update` (Section 3.2), and shows its progress. When it has finished, **Restart Now** closes the app (asking about unsaved parameters as usual) and starts the new version; **Later** keeps working with the old one until the next start. If the update cannot finish, for example without a network, the window says so; then close the app and run `mbstudio update` in a terminal. A copy run from a source folder (`pip install -e`) is updated with `git pull` instead, so the button is off there.

10 Exporting Plots

Use **File** → **Export Plot...** to save the current figure.

- **PDF**: the first page is a report figure: the top panel, then one panel each for w and R , then the pump, boundary field, T_1 and T_2 where they are not trivial. It appends one or more text-report pages starting with *How to cite* (the references of the app and of mbsolver, with versions and DOIs), then metadata and the current parameter values.
- **PNG / SVG**: exports the figure only.

The exported PDF also includes document metadata such as title, creator, and author name.

11 Saving and Loading Parameters

11.1 Saved JSON Structure

Saved parameter snapshots use the structure

```
{
  "metadata": { ... },
  "config":   { ... },
  "params":   { ... }
}
```

where:

- **metadata** stores version, save time, username, and mode information,
- **config** stores widget/configuration state,
- **params** stores the current parameter values.

In fit mode, the **params** section stores the saved values for all saved velocities in the source/period group.

The `solve.sample` entry stores only the physical sample values (ν , Γ , T_0 , L , n_0 , in SI units). The derived quantities such as η and θ_0 are computed from them whenever the model is solved; values of η and θ_0 found in files from older versions are ignored.

11.2 App Defaults

The command **File** → **Save Parameters as Default** stores the current standalone state in the platform's persistent application settings using Qt's `QSettings`. On later launches, the app restores these saved defaults instead of the factory defaults.

12 Using the Solver from Python

Parameter files saved by the app can be solved in your own scripts, for example to run many cases without the window. Install the app with `pip` in the Python environment you use (a `pipx` installation is kept separate from your own Python):

```
pip install maxwell-bloch-studio
```

A minimal script:

```
import mbstudio

f = mbstudio.ParamsFile("run_1.json")      # a file saved by the app
params = f.params()                       # a fresh dictionary, safe to change
params["solve.sample"]["l"] = 1.0e13      # change any value
res = mbstudio.solve(params)              # same numbers as the app

res["time"]                               # time grid
res["intensity"][-1]                      # intensity at the exit of the medium
```

- `ParamsFile` also gives `metadata`, `app_version` and, for the parameter file of a data folder, the list `velocities`; choose one component with `f.params(12.3)`.
- `params` is the dictionary stored under "params" in the file (Section 11.1). Files from older versions are converted automatically.
- η and θ_0 are computed from the sample values $(\nu, \Gamma, T_0, L, n_0)$, as in the app; `mbstudio.scaling(nu=..., gamma=..., t0=..., l=..., n0=...)` returns them together with the other derived quantities.
- The result holds `time`, `intensity`, `w` and `R` (one row per stored z plane, from the entrance to the exit), `lambda_n` (pump), `A0` (boundary field), `t1` and `t2`, in the solver's dimensionless units. The flux scaling and offsets of the Fit Mode tab are not applied.

Note

Help → **Use from Python...** shows a commented example script that lists every parameter and result key; copy it or save it as a `.py` file to start from.

The numerical solver itself is the separate package `mbsolver` (`pip install mbresolver`), for building parameters directly in code.

13 Typical Workflows

13.1 Standalone Simulation Workflow

1. Launch the application.

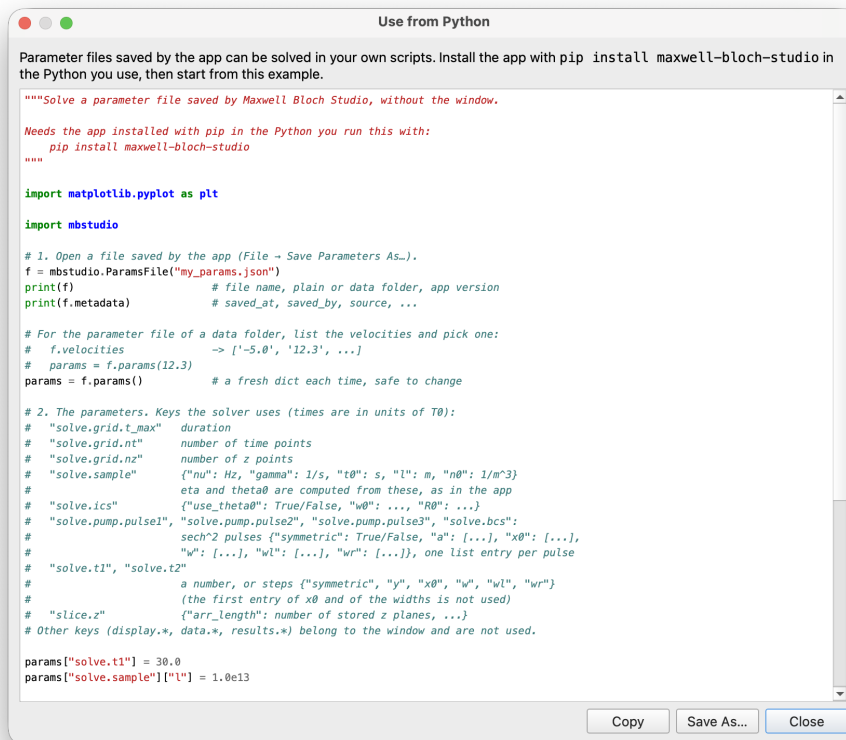


Figure 4: Help → Use from Python...: the example script, ready to copy or save.

2. Set the physical sample parameters in **Sample**.
3. Adjust the numerical resolution in **Solver** if needed.
4. Choose the initial state in **ICs**.
5. Set the relaxation times in **T₁** and **T₂**, either as constants or as step profiles.
6. Define the boundary field in **BCs**.
7. Add up to three pump contributions in **Pump 1**, **Pump 2**, and **Pump 3**.
8. Inspect different propagation slices in **Slice**.
9. Use **Range** to zoom the displayed region.
10. Export the plot or save the parameters when satisfied.

13.2 Observational Fitting Workflow

1. Open a data folder with File → Open Data Folder....
2. Select a velocity component from the toolbar.
3. Use **Fit Mode** to choose the correct data time unit and apply any needed offsets.
4. Adjust the physical and driving parameters in **Sample**, **T₁**, **T₂**, **ICs**, **BCs**, and the pump tabs.

5. Use the fit controls to align the model in time and scale it vertically.
6. Monitor the χ^2 -style mismatch as a quick diagnostic.
7. Save the current velocity.
8. Repeat for the remaining velocities.

14 Troubleshooting

The plot does not seem to change after I edit a value.

Solver parameters under `solve.*` trigger a new solve. Pure display parameters such as range settings, selected slice index, or fit offsets may update the display without re-solving.

Open Parameters refuses one of my saved JSON files.

That file is probably a fit-mode source parameter file. Load the corresponding data folder instead; the app will read that file automatically.

The imported folder is rejected.

Check the filename pattern and make sure the folder contains only one source name and is not mixing period-tagged and non-period-tagged data files.

The solver becomes slow.

Large values of N_t and N_z increase compute time. The first solve after launch may also be slower because the JIT-compiled kernels are being prepared.

The mismatch value is blank.

This happens when there is no loaded data, no visible overlap between the plotted data and model, or no valid interpolated comparison range.

The PDF export contains extra pages.

This is expected. PDF export includes a report section with metadata and parameters in addition to the figure page.