EM Inversion Workshop
GUI Manual

This manual describes how to set up a Python environment, install dependencies, and run the four workshop GUIs. For each GUI it explains how to launch it, how to set parameters, and—for non-expert users—what each parameter means physically and what impact it has on the simulation or workflow.

The four GUIs are:

- **01_fw_setup** — Prepare FD modelling inputs from a SEG-Y file.
- **02_visualization** — Run FD modelling and visualize amplitude/phase from shot records.
- **03_inversion** — Generate inversion inputs and run inversion locally.
- **04_results** — View inversion results and compare real data with synthetics.

All GUIs are Jupyter notebooks served as code-hidden applications via Voila (ipywidgets and Plotly). Run Voila from the **workshop root** so that `scripts` and `third_party` are on the Python path.

1 Environment setup

1.1 Creating a virtual environment (venv)

From the workshop root:

```
python3 -m venv .venv
source .venv/bin/activate   (Linux/macOS)
.venv\Scripts\activate   (Windows)
```

Then upgrade pip: `pip install -upgrade pip`.

1.2 Creating a conda environment

```
conda create -n em_workshop python=3.10
conda activate em_workshop
```

1.3 Installing dependencies

With the virtual or conda environment activated, install:

Package	Purpose
`voila`	Serves notebooks as standalone web apps (code hidden).
`ipywidgets`	Interactive widgets in the GUI.
`plotly`	Interactive plots.
`numpy`	Numerical arrays (used by inversion and results GUIs).
`segyio`	Read/write SEG-Y files (required by 01_fw_setup).

Command:

```
pip install voila ipywidgets plotly numpy segyio
```

For reproducibility you can add a `requirements.txt` in the project root with these packages and then run `pip install -r requirements.txt`.

2 How to run the GUIs

From the **workshop root** directory, use the provided shell scripts. Each script changes into the script directory and starts Voila with the corresponding notebook (sources stripped so users see only the interface).

- `./start_01_fw_setup.sh` → 01_fw_setup
- `./start_02_visualization.sh` → 02_visualization
- `./start_03_inversion.sh` → 03_inversion
- `./start_04_results.sh` → 04_results

Alternatively, from the workshop root:

```
voila 01_fw_setup.ipynb -strip_sources=True
```

(and similarly for `02_visualization.ipynb`, `03_inversion.ipynb`, `04_results.ipynb`.)

Note: 01_fw_setup and 04_results both use `.voila_server.pid`; avoid running both at the same time if you rely on that PID file.

3 GUI 1: 01_fw_setup

3.1 Purpose

Generate finite-difference (FD) modelling inputs from a SEG-Y resistivity model: conductivity/permittivity files (`sg.rss`, `ep.rss`), source wavelet (`wav2d.rss`), survey geometry (`Survey.rss`), and `mod.cfg` settings. Intermediate files are written to a temporary session folder; final outputs go to `FDmodel`.

3.2 Parameters (physical meaning and impact)

- **SEGY file** — Path to the input SEG-Y file containing the resistivity model. This defines the subsurface you are simulating: the grid and resistivity values control where EM waves propagate and how they attenuate. Use a model that matches the geology and scale of your survey.
- **ep constant** — Electrical permittivity (relative dielectric constant) assigned uniformly in the model. It affects wave speed and wavelength. It is usually kept constant when the focus is on conductivity/resistivity. Changing it shifts phase and can affect how well synthetics match real data if the data are sensitive to permittivity.

- **tx0 (m), tz0 (m), dtx (m), ntx** — Transmitter (source) geometry: first source position in x and z, spacing between sources, and number of sources. This is where the EM signal is "injected." All positions must lie inside the model bounds. More sources give more data but increase run time.
- **rx0 (m), rz0 (m), drx (m), nrx** — Receiver geometry: first receiver x and z, receiver spacing, and number of receivers. This is where the field is "measured." Spacing controls spatial sampling and resolution; typically receivers are offset from the source.
- **flist (Hz)** — Comma-separated list of source frequencies (e.g. `2000,4000,6000`). Lower frequencies penetrate deeper but have lower resolution; higher frequencies resolve shallow structure better. Choose frequencies that match the content you want to invert or compare with data.
- **dt (s)** — Time step for the wavelet and for the FD solver. It must be small enough for numerical stability (CFL) and to sample the highest frequency. Too large a value causes instability or numerical dispersion; too small increases run time and memory.
- **n_periods** — Number of periods of the lowest frequency in the wavelet. This controls how long the source radiates and affects the amplitude and phase of the synthetic data.
- **alpha** — Ramp parameter (0–1) for the ramp square-wave wavelet. It shapes the rise and fall of the source and affects which frequencies dominate in the synthetic.
- **f_min (Hz), f_max (Hz)** — Frequency range used to design the FD grid. The grid is chosen so that the minimum skin depth (at `f_max` and minimum resistivity) and the CFL condition are satisfied. A wider range or more extreme resistivity usually implies a finer grid and heavier runs.
- **rho_min (Ohm-m), rho_max (Ohm-m)** — Expected resistivity range in the model. These are used to compute skin depth and thus grid spacing: lower resistivity gives shorter skin depth and requires a finer grid; higher resistivity allows a coarser grid. Set them to bracket your actual model to avoid undersampling or unnecessary refinement.
- **dimension** — 2D or 3D. The FD solver uses a 2D or 3D grid accordingly. 2D is much faster and is appropriate for line surveys.
- **points/skin** — Number of grid points per skin depth at the worst case (highest conductivity, highest frequency). More points improve accuracy but increase grid size and run time; too few cause numerical dispersion and wrong amplitudes/phases.
- **k_cfl** — CFL-like factor for time stepping. Larger values allow larger time steps (faster) but can cause instability; smaller values are safer but slower.
- **apertx (m)** — Horizontal extent (aperture) of the FD domain. It must cover all sources and receivers plus padding for absorbing boundaries. Too small truncates the wavefield; too large wastes computation.
- **Buttons** — "Load model" loads the SEG-Y and updates resistivity bounds; "Plot model" refreshes the resistivity plot; "Build intermediate sg/ep" writes conductivity and permittivity files; "Build intermediate wav2d" and "Plot wavelet" create and visualize the wavelet; "Generate survey.cfg + Survey.rss" writes survey geometry; "Compute FD design" computes recommended grid spacing and time step; "Generate FD inputs (Finalize setup)" interpolates to the target grid and writes everything to `FDmodel`. Use "Quit GUI server" to stop Voila.

4 GUI 2: 02_visualization

4.1 Purpose

Run FD modelling (via `FDmodel/runmod.sh` and the MPI binary), load `Hxshot.rss` and `Hzshot.rss` outputs, compute two-point amplitude and phase, and visualize or save the results.

4.2 External requirement

The MPI FD binary (e.g. `~/software/rockem-suite/bin/mpiEmmodADITE2d`) must be installed and the path in `runmod.sh` must be correct. The GUI and run script reference this binary.

4.3 Parameters (physical meaning and impact)

- **nproc** — Number of MPI processes used to run the FD modelling. More processes typically shorten wall-clock time if you have enough cores; it does not change the physics of the result.
- **flist (Hz)** — Frequencies at which amplitude and phase are extracted from the time-domain FD output. They should match the frequencies in your wavelet and setup so that synthetics are comparable to data or to theory.
- **start_t (s), end_t (s)** — Time window over which the FD solution is Fourier-transformed to obtain amplitude and phase. The window must contain the full arrival of the signal for each frequency; too short a window truncates the waveform and distorts amplitude and phase.
- **n_pairs** — Number of frequency pairs used in the two-point amplitude/phase processing. It affects how many frequencies are analyzed and the structure of the saved results.
- **component** — Hx or Hz: which magnetic field component is plotted (in-line or vertical). Different components have different sensitivity to structure and are used for different interpretation or inversion choices.
- **plot** (metric) — Which quantity is plotted: amplitude or phase vs receiver, vs transmitter, or vs frequency. Amplitude reflects attenuation and geometric spreading; phase reflects travel path and velocity. Choose the view that matches your interpretation or data comparison.
- **frequency, tx, local rx, trace idx** — Selectors for which frequency, source, receiver index, or trace to display. Use them to inspect specific source–receiver pairs or frequencies relevant to your interpretation.
- **Buttons** — "Run modelling (background)" starts the FD run; "Stop run" stops it; "Refresh run status" updates job progress; "Load FD outputs" loads `Hxshot.rss`/`Hzshot.rss`; "Compute amplitude/phase" runs the extraction; "Save processed results" writes the NPZ; "Quit GUI server" stops Voila.

5 GUI 3: 03_inversion

5.1 Purpose

Generate inversion inputs from `FDmodel` and the inversion template, run the inversion locally (staged into the next `InversionRunN` directory), and monitor logs and model progress.

5.2 External requirement

The MPI inversion binary (e.g. `~/software/rockem-suite/bin/mpiEminvADITE2d`) must be installed and the path in `runinv.sh` must be correct. The GUI and run script reference this binary.

5.3 Parameters (physical meaning and impact)

- **Initial model** — Whether the starting model is defined by uniform conductivity (sigma) or uniform resistivity (rho). The physics is the same (sigma = 1/rho); the choice is for convenience. A good starting value is often a background estimate from your data or prior knowledge.
- **sigma (S/m), rho (Ohm.m)** — Value of the uniform initial model. If it is too far from the true structure, the inversion may need more iterations or get stuck; if it is reasonably close, convergence is faster and more stable.
- **Max iter** — Maximum number of inversion iterations. More iterations allow a better fit but increase run time; too few may leave the model under-fit.
- **apertx (m), dtx, dtz** — Grid and extent parameters passed to the inversion. They should be consistent with the FD model used to generate synthetics so that the forward and inverse use the same discretization.
- **Clean selected InversionRunN before run** — If checked, the chosen run directory is cleared before staging new inputs. Use this to avoid mixing old files when re-running; uncheck to preserve or append to an existing run.
- **Load run** — Select which `InversionRunN` to monitor or use as context. Lets you switch between runs to compare progress or results.
- **nproc** — Number of MPI processes for the inversion run. More processes reduce wall-clock time when you have enough cores.
- **Buttons** — "Generate inversion inputs" writes inputs to `InversionInput/`; "Run inversion locally" stages and starts the inversion; "Refresh from progress.log" updates the plot and logs; "Stop run" stops the process; "Quit GUI server" stops Voila.

6 GUI 4: 04_results

6.1 Purpose

View inversion results: select a run and model (e.g. `sg0`, `sg_ls` at an iteration), view 2D resistivity with true-model comparison, resistivity vs depth or vs x slices, compare real data with synthetics (either loaded from the run or generated by running FD for the selected model), and export selected resistivity models to SEG-Y.

6.2 Parameters (physical meaning and impact)

- **Run, Model** — Which inversion run and which model file (e.g. initial `sg0`, or linesearch `sg_ls` at an iteration). Use these to inspect how the model evolved and to compare with the true model.
- **Export selected model to SEGY** — Writes the currently selected model to `InversionRunN/SEGY_output/` using the SEG-Y from setup as a template (same header layout, trace/sample organization, origin, and sampling). If the model grid differs from

the template grid, the GUI interpolates to the template grid before writing so exported SEG-Y matches the setup SEG-Y geometry exactly.

- **Export naming and overwrite behavior** — Exported files are deterministic and overwrite on repeated export of the same model: `sg_up.rss-11` → `sg_up_iter011.segy`, `sg0.rss` → `sg0_iter000.segy`. For `sg_ls.rss`, the GUI uses the latest iteration seen in `progress.log` when available (`sg_ls_iterNNN.segy`), otherwise `sg_ls_iterunknown.segy`.
- **Export status (near button)** — A dedicated status banner is shown directly below the run/model/export controls. It reports in-progress, success, or failure, including output filename and folder, so users do not need to scroll to verify export results.
- **x (m), z (m)** — Positions for vertical and horizontal resistivity slices. Use them to see resistivity vs depth at a given x, or vs horizontal position at a given depth, for interpretation.
- **Freqs (Hz)** — Frequencies used when loading real data or generating/running synthetics. They must match the frequencies in your survey and inversion for a meaningful comparison.
- **Load real data / Load synthetics from run / Generate synthetics from model** — Real data are your observations; "Load synthetics from run" loads synthetics saved during inversion; "Generate synthetics from model" runs the FD solver for the selected model so you can test that model without re-running the full inversion. Use these to compare observed vs predicted and assess fit.
- **nproc** — Number of MPI processes when "Generate synthetics from model" is used; it only affects the run time of that FD job.
- **component, plot, frequency, tx, local rx, trace idx** — Same interpretation as in 02_visualization: choose the component and plot type that match your comparison or interpretation needs.
- **Quit GUI server** — Stops the Voila server.

7 Troubleshooting

- **Missing `segyio`** — If 01_fw_setup fails to import workshop modules, ensure `segyio` is installed and that Voila is started from the workshop root so `scripts` and `third_party` are on the path.
- **SEGY export cannot find template** — 04_results export expects a setup SEG-Y template from `FDmodel/setup_metadata.json` (`segy_template_path`), with fallback to `input.segy` in the project root. Run 01_fw_setup and finalize setup first, or place a valid `input.segy` in the root.
- **Missing MPI FD binary** — 02_visualization and 04_results (when generating synthetics) need the MPI FD binary (e.g. `~/software/rockem-suite/bin/mpiEmmodADITE2d`). Check the path in `FDmodel/runmod.sh` and that the binary exists.
- **Missing MPI inversion binary** — 03_inversion needs the MPI inversion binary (e.g. `~/software/rockem-suite/bin/mpiEminvADITE2d`). Check the path in `scripts/templates/runinv.sh` and that the binary exists.
- **PID file** — 01_fw_setup and 04_results both use `.voila_server.pid`; do not run both at once if you rely on that file.

8 File locations

- `FDmodel/` — FD inputs and outputs (e.g. `mod.cfg`, `sg.rss`, `Survey.rss`, `Data/`).
- `InversionInput/` — Inversion input files (e.g. `inv.cfg`, `sg0.rss`).
- `InversionRun0/`, `InversionRun1/`, ... — Staged run directories for each inversion.
- `InversionRunN/SEGY_output/` — SEG-Y exports generated from 04_results for selected models.
- `scripts/templates/` — Templates for `inv.cfg`, `mod.cfg`, `survey.cfg`, `runinv.sh`.
- `third_party/` — Vendored Rockseis Python code used by the workshop.