SPM-Kit · Fathom

Usage Guide

Source manual `0.1.5.dev0` (Alpha)

2026-07-29

GitHub release `0.1.4` · PyPI `0.1.2`

Creator, author, and lead developer: José Labarca Baeza
License: MIT
Repository: https://github.com/kegouro/spmkit
Site: https://kegouro.github.io/spmkit

Canonical documentation: This PDF is a printable companion. The Markdown version at `docs/user-guide.md` is the authoritative reference and may contain additional detail. Both are verified against the development tree `0.1.5.dev0`. The latest tagged source release is `0.1.4`; PyPI currently serves `0.1.2`.

Acknowledgements: Tomás Corrales and the SPM Lab at Universidad Técnica Federico Santa María provided selected experimental datasets and laboratory context during the development and evaluation of SPM-Kit. María Saavedra Fredes and Benjamin Schleyer helped locate and share candidate datasets for the validation campaigns.

SPM-Kit is an open-source Python toolkit for analysing AFM/KPFM scanning-probe-microscopy data. Fathom is its desktop workspace. This guide covers installation, conceptual model, supported formats, the Fathom GUI, CLI reference, Python API, scientific validation, and extensibility.

This open-source project has been developed without dedicated institutional funding. Development status: Alpha (3 — Alpha). API may change before 1.0.

Contents

1 Introduction

1.1 What is SPM-Kit?

SPM-Kit is an open-source Python toolkit for analysing AFM/KPFM scanning-probe-microscopy data. It reads demonstrated instrument-file variants and produces inspectable results: roughness (ISO 25178), KPFM contact-potential statistics, nanomechanical property maps, thermal-tune cantilever calibration, grain detection, spectral/fractal analysis, and single-molecule force spectroscopy.

SPM-Kit was created and is authored and led by José Labarca Baeza as an independent software project. Acknowledged dataset and laboratory-context contributions do not create software authorship or institutional ownership.

1.2 What is Fathom?

Fathom is the desktop workspace application bundled with SPM-Kit. It is a PyQt6 + pyqtgraph GUI organised into *perspectives* (task-oriented views), not flat tabs. A command palette (`Ctrl+K`) provides keyboard access to every function.

1.3 Component relationship

```
spmkit (Python package)
  spmkit.core   -- numerical engine (pure Python, no UI imports)
  spmkit.cli    -- command-line interface (Typer + Rich)
  spmkit.gui    -- Fathom desktop workspace (PyQt6 + pyqtgraph)
```

- `spmkit` is the Python package and CLI entry point.
- **Fathom** is the graphical workspace.
- **SPM-Kit** is the project/ecosystem name.

1.4 What SPM-Kit does *not* do

- It is **not a microscope controller**.
- It is **not a certified metrology tool**.
- It does **not** provide medical, clinical, or regulatory reports.
- Simulation features are **educational**, not quantitative digital twins.

1.5 Status

SPM-Kit is **alpha-stage** (Development Status 3). The strongest retained evidence is **LEVEL 3** for Sa, Sq and Sz under frozen campaign conditions: 48 synthetic matrices produced 144/144 conforming comparisons and 12 public experimental GWY records produced 36/36 shared-matrix comparisons against Gwyddion 2.71. Six Nanoscope III `.spm` files support a partial **LEVEL 2** claim with a documented pre-freeze unblinding incident. KPFM, spectral, grain, and resonance utilities remain Level 1; contact and chain models have claim-scoped synthetic Level 2 evidence. No general Level 4 or Level 5 claim is made (see §14).

2 Installation

Requirements: **Python >= 3.11**.

The documented source is `0.1.5.dev0`; the latest GitHub release is `0.1.4`; PyPI currently serves `0.1.2`. Choose the source explicitly.

2.1 Core only

```
python -m pip install "spmkit @ git+https://github.com/kegouro/spmkit@v0.1.4"
```

Installs the numerical core and CLI. Optional format dependencies still require extras.

2.2 GUI / Fathom

```
python -m pip install "spmkit[gui] @ git+https://github.com/kegouro/spmkit@v0.1.4"
```

Adds: PyQt6, pyqtgraph, matplotlib, Crameri colormaps, matplotlib-scalebar.

2.3 All features

```
python -m pip install "spmkit[all] @ git+https://github.com/kegouro/spmkit@v0.1.4"
```

2.4 Extras matrix

Extra	Provides	Deps
`gui`	Fathom desktop workspace	PyQt6, pyqtgraph, matplotlib
`viz`	Publication figures, colormaps	matplotlib, cmcrameri, scalebar
`gwy`	Gwyddion .gwy read/write	gwyfile
`hdf5`	HDF5 import/export	h5py
`grains`	Grain/particle detection	scipy
`report`	HTML/PDF reports	Jinja2 + spmkit[viz]
`nanosurf`	Optional NSFopen dependency	NSFopen
`afm`	JPK QI, .ibw, HDF5, NT-MDT	afmformats
`jpk`	JPK TIFF force curves	tifffile
`parallel`	Parallel force-volume	joblib
`pandas`	DataFrame export	pandas
`dev`	Lint, test, type-check	pytest, ruff, mypy, black
`test-gui`	GUI test runner	pytest-qt
`docs`	Documentation build	mkdocs-material
`all`	Reader, GUI, visualization, report, grain, HDF5 and GWY extras	—

2.5 Development installation

```
git clone https://github.com/kegouro/spmkit
cd spmkit
pip install -e ".[all,dev]"
```

2.6 Headless / HPC

Install the tagged core source shown above. Add `[viz]` for headless figure generation.

2.7 Platform notes

Developed and tested on **macOS** and **Linux**. Windows is not regularly tested. Fathom requires a display; set `QT_QPA_PLATFORM=offscreen` for headless tests.

3 Verifying the installation

```
spmkit --version          # 0.1.4 tag; 0.1.2 PyPI; 0.1.5.dev0 main
python -c "import spmkit; print(spmkit.__version__)"
```

4 Conceptual model

4.1 Architecture

Three strictly separated layers (enforced by `tests/test_architecture.py`):

- `core/` — pure Python, zero UI imports.
- `cli/` — Typer + Rich. Only imports the public `core` API.
- `gui/` — PyQt6 + pyqtgraph. Only imports the public `core` API.

4.2 Data model

```
from spmkit import load, SPMData, SPMChannel

data: SPMData = load("scan.nid")
ch: SPMChannel = data["Z-Axis"]

ch.data         # 2D ndarray in physical units
ch.unit         # "m", "V", "{\textdegree}"
ch.x_range      # metres
ch.y_range      # metres
ch.direction    # "forward" | "backward"
```

4.3 Force-spectroscopy domain

For force curves: `ForceCurve`, `ForceSegment`, `ForceVolume` (lazy-loaded), `Calibration`.

4.4 Result objects

Analysis results are immutable dataclasses. Use `to_dict()` before passing them to the export helpers.

5 Supported file formats

Extension	Source	Type	Access	Status
.nid	NanoSurf classic	Image + spectroscopy	Read	Implemented, scoped
.nhf	NanoSurf HDF5	Images	Read	Experimental; hdf5 extra
.gwy	Gwyddion	Images	R/W	Implemented; gwy extra
.spm/.00N	Bruker / DI	Images	Read	**Partial LEVEL 2**
.jpk-force/.jpk	JPK	Single curve	Read	Implemented, scoped
tagged TIFF	JPK export	Force data	Read	Experimental; jpk extra
.jpk-qi-data/map/series	JPK	Force data	Read	afmformats adapter
.ibw/.h5/.tab	Adapter sources	Force data	Read	afmformats adapter

.npz	SPM-Kit Phantoms		Images	Read	Declared bundle only

5.1 Maturity vocabulary

LEVEL 0 Claimed, without retained executable evidence.
LEVEL 1 Software verified under declared scope.
LEVEL 2 Known values recovered under declared numerical scope and tolerance.
LEVEL 3 Frozen external cross-validation under declared conditions.
LEVEL 4 Scoped physical-reference validation.
LEVEL 5 Independent reproduction of the declared result.

5.2 What the .nid evidence means

The parser has synthetic byte-budget and orientation tests plus selected laboratory-context comparisons. The private instrument corpus is not distributed, so this supports scoped Level 1 software evidence rather than universal NanoSurf compatibility. The Sa/Sq/Sz shared-matrix campaigns are separate Level 3 algorithm evidence. Parser, algorithm, and physical-validation claims must remain distinct.

6 Quick start

```
# Inspect a file
spmkit info scan.nid

# Roughness (ISO 25178)
spmkit roughness scan.nid -c Z-Axis --level plane

# Publication figure
spmkit figure scan.nid -c Z-Axis -o topo.png --colormap batlow

# Launch Fathom
spmkit gui scan.nid

# Force curve fit
spmkit forcecurve curva.jpk-force --model dmt --tip-radius 2e-8
```

Python:

```
from spmkit import load
from spmkit.core.analysis import leveling, roughness

data = load("scan.nid")
flat = leveling.plane_fit(data["Z-Axis"])
stats = roughness.statistics(flat)
print(f"Sa = {stats.Sa:.3g} {stats.unit}")
```

7 Fathom desktop workspace

7.1 Launching

```
spmkit gui               # Fathom (default)
spmkit gui scan.nid      # open file on launch
spmkit gui --legacy      # legacy 7-tab app (fallback)
```

7.2 Opening data

Drag and drop a file onto the window, or `Ctrl+O`. Fathom inspects the file and routes image channels to image perspectives and force/spectroscopy data to force perspectives. If a file contains both, Fathom asks which to open.

7.3 Projects (.spmproj)

`Ctrl+S` saves the open file, pipeline parameters, and active perspective as a human-editable YAML file.

7.4 Appearance

- `Ctrl+Shift+L` — toggle light/dark theme.
- `Ctrl+Shift+A` — appearance dialog (theme presets, accent colour, font size).
- Settings persist between sessions.
- The theme feeds the Qt app, pyqtgraph, and matplotlib simultaneously.

7.5 Command palette

`Ctrl+K`, then type a command name and press Enter. Indexes all perspectives and registered commands.

8 Perspectives

Fathom presents 12 perspectives through 6 modules. Each perspective shows only the panels relevant to its workflow.

8.1 Image perspectives

Module: image	
`image`	Channel viewer, leveling (plane/poly/align-rows), colormap, line profile ROI, roughness + KPFM analysis, profile export.
`grains`	Particle/grain detection on levelled topography; colour overlay, size statistics. Requires `grains` extra.
`spectral`	Radial PSD (log-log), fractal dimension D, Hurst exponent H, correlation length.
`resonance`	Thermal tune: SHO fit → f_0, Q, spring constant by equipartition.
`evaporation`	Mass sensing: load series of tuning .nid files → f(t), mass, d^2 law fit.

8.2 Force perspectives

Module: force	
`force`	Navigate curves, fit Hertz/DMT/Sneddon, inspect modulus ± 1σ, adhesion, dissipation, R^2. Editable pipeline, Monte Carlo uncertainty. Export scientific CSV.

`smfs`	Single-molecule force spectroscopy: prominence-based rupture detection, WLC/FJC polymer fits, contour-length histogram.
`map`	Force-volume → property maps (modulus, adhesion, contact, R^2) + histogram. Vectorised (CPU/GPU) or per-curve pipeline.
`batch`	Process folders of force curves → summary CSV.

8.3 Other perspectives

Module: figure	
`figure`	WYSIWYG publication figure editor: title, axes, colormap, scale bar, draggable annotations. Export PNG/SVG/PDF.
Module: view3d	
`view3d`	3D topography surface with hillshade lighting, visual Z exaggeration.
Module: simulator	
`simulator`	Educational cantilever digital twin: thermal noise spectrum, resonance shift with added mass.

9 End-to-end workflows

9.1 Topography leveling and roughness

CLI:

```
spmkit roughness scan.nid -c Z-Axis --level plane
```

Python:

```
from spmkit import load
from spmkit.core.analysis import leveling, roughness

data = load("scan.nid")
flat = leveling.plane_fit(data["Z-Axis"])
stats = roughness.statistics(flat)
print(stats.Sa, stats.Sq, stats.Sz, stats.unit)
```

GUI: Drag file → Image perspective → select channel → Plane Fit → read results from Analysis panel.

Caveats

Roughness depends on scan size, pixel density, and leveling method. Always report these alongside roughness values.

9.2 Force-curve contact mechanics

```
spmkit forcecurve sample.jpk-force --model dmt --tip-radius 2e-8
```

Contact models: `sphere` (Hertz), `paraboloid` (Sneddon paraboloid), `cone` (Sneddon cone), `dmt` (Hertz + adhesion offset). The experimental JKR model (`spmkit jkr`) fits adhesive contact but has not been validated against an independent reference.

Caveats

Modulus depends on tip radius (uncertain), contact model (approximation), and calibration. Report all three alongside the result. JKR is experimental.

9.3 Force-volume property mapping

```
spmkit forcemap volume.nid --model dmt --fast -f maps.png
spmkit forceexport volume.nid -o ./results
```

Creates property maps (modulus, adhesion, contact point, R^2), histograms, and CSV exports. Use `-pipeline` for variable-length curves (QI data), `-parallel` for multi-core, `-backend gpu` for GPU (requires CuPy).

9.4 Thermal tune

Load a thermal spectrum in the Thermal Tune perspective or use:

```
from spmkit.core.analysis.mechanics import thermal_spring_constant
```

9.5 Evaporation / mass sensing

```
spmkit evaporation ./tuning/ -k 0.3 -o evap.csv
```

Tracks f(t) → mass → evaporation rate. Fits the d^2 law for diffusion-limited evaporation.

9.6 KPFM

```
spmkit analyze scan.nid --tip-wf 4.8 -o ./results
```

Reports mean CPD, standard deviation, and sample work function ($\Phi_{\text{sample}} = \Phi_{\text{tip}} - e \cdot \text{CPD}$).

9.7 Other workflows

```
spmkit grains scan.nid --min-size 10           # grain detection
spmkit psd scan.nid -c Z-Axis                   # fractal analysis
spmkit batch ./measurements/ -o summary.csv     # batch processing
spmkit fbatch ./curves/ -o batch.csv            # force batch
spmkit figure scan.nid -o topo.png --colormap batlow  # figure export
```

10 Command-line reference

19 commands are registered in the Typer application.

Command	Description
`info`	Show file metadata and channels
`roughness`	ISO 25178 parameters (Sa, Sq, Sz, Ssk, Sku)
`psd`	Spectral: fractal dimension, Hurst, correlation length
`analyze`	Full pipeline: roughness + KPFM → CSV + JSON
`nanomech`	Force-curve fit (Hertz/DMT/Sneddon) → Young's modulus
`grains`	Grain/particle detection and size statistics
`batch`	Process folder → summary CSV

`evaporation`	Mass sensing: f(t) → mass, evaporation rate, d^2 law
`figure`	Publication figure: PNG/SVG/PDF with scale bar
`convert`	Format conversion (.nid → .gwy / .h5)
`verify`	Up to 8 structural/numeric .nid checks
`gui`	Launch Fathom (or legacy with –legacy)
`workspace`	Force-curve workspace (redesign)
`forcecurve`	Fit single force curve (JPK/NanoSurf)
`forcemap`	Force-volume → property maps
`forcereport`	Force-volume report (HTML/LaTeX/PDF)
`forceexport`	Full export bundle to folder
`fbatch`	Batch force processing → CSV
`jkr`	[**Experimental**] JKR adhesive contact fit

10.1 Common options

- `--channel / -c` — channel name (default: `Z-Axis` for images, `Deflection` for force)
- `--output / -o` — output path
- `--level / -l` — leveling: `plane`, `poly`, `none`
- `--model` — contact model: `sphere`, `paraboloid`, `cone`, `dmt`
- `--tip-radius` — tip radius in metres (default: `1e-08`)

Run `spmkit <command> -help` for full options.

11 Python API

11.1 Public imports

```
from spmkit import load, SPMData, SPMChannel, __version__
from spmkit.core.analysis import leveling, roughness, kpfm, mechanics
from spmkit.core.analysis import spectral, grains, resonance
from spmkit.core.export import to_csv, to_json, to_hdf5
from spmkit.core.viz import FigureSpec, save_figure
```

11.2 Loading and inspecting

```
data = load("scan.nid")     # auto-detect format
print(data.names)           # channel names
ch = data["Z-Axis"]         # SPMChannel
```

11.3 Analysis

```
flat = leveling.plane_fit(data["Z-Axis"])
stats = roughness.statistics(flat)
cpd = kpfm.statistics(data["CPD"], tip_work_function=4.8)
frac = spectral.fractal_dimension(flat)
grains_result = grains.detect(flat, min_size=4)
```

11.4 Force spectroscopy

```
from spmkit.core.io.forceload import load_nid_force
from spmkit.core.analysis.mechanics import fit_hertz

volume = load_nid_force("sample.nid")
```

```
result = fit_hertz(volume.curves[0], tip_radius=10e-9, model="dmt")
print(result.young_modulus, result.r_squared)
```

11.5 Export and figures

```
to_csv(stats, "roughness.csv")
to_json(stats, "roughness.json")
to_hdf5(data, "scan.h5")

spec = FigureSpec(title="AFM Topography", colormap="batlow")
save_figure(flat, spec, "topo.png")
```

11.6 Recipes

```
from spmkit.core.pipeline import Recipe, Step, run

recipe = Recipe(steps=(
    Step(op="calibrate"),
    Step(op="find_contact_point"),
    Step(op="fit_elasticity", params={"model": "dmt", "tip_radius": 2e-8}),
))
_, ctx = run(recipe, curve)
print(ctx["young_modulus"], ctx["r_squared"])
```

12 Keyboard shortcuts

Shortcut	Action
Ctrl+K	Command palette
Ctrl+O	Open file
Ctrl+S	Save project
Ctrl+M	Calculate map
Ctrl+Shift+R	Generate report
Ctrl+Shift+L	Toggle light/dark theme
Ctrl+Shift+A	Customize appearance
Ctrl+Left / Ctrl+Right	Previous / next curve
Ctrl+1 ... Ctrl+5	Tab switch (legacy app only)

13 Exports and provenance

13.1 Export formats

Format	Via	Notes
CSV	to_csv(), CLI	Units in headers; empty cells for NaN
JSON	to_json(), CLI	Structured; programmatic consumption
HDF5	to_hdf5(), convert	Self-describing binary; requires hdf5 extra
.gwy	spmkit convert	Round-trip with Gwyddion
PNG/SVG/PDF	spmkit figure	Publication-quality figures
HTML/PDF	spmkit forcereport	Full force-volume report
YAML	core.pipeline	Reproducible pipeline recipe

13.2 Provenance

- `SPMData.source_path` records the original file path.
- `SPMData.metadata` preserves instrument parameters.
- `spmkit verify` performs up to eight structural and numeric checks over the declared .nid header and binary layout; malformed files can stop early.
- YAML recipes capture the exact pipeline parameters.
- Exports include metadata when the format supports it.

14 Scientific validation

Evidence belongs to a claim, data family, version, preprocessing path, tolerance, and campaign. A high level for one metric never transfers automatically to an adjacent reader, model, or sample.

14.1 Evidence-level vocabulary

LEVEL 0 — CLAIMED Documented without retained executable evidence.
LEVEL 1 — SOFTWARE VERIFIED Automated tests exercise the declared behavior.
LEVEL 2 — NUMERICALLY VERIFIED Known values are recovered within a stated scope and tolerance.
LEVEL 3 — CROSS VALIDATED An external route is compared under a frozen protocol.
LEVEL 4 — PHYSICALLY VALIDATED A physical reference supports the scoped capability.
LEVEL 5 — REPRODUCIBILITY VALIDATED An independent party reproduced the result.

14.2 Status summary

Capability	Evidence	Status	Limitations
Sa/Sq/Sz synthetic	48 cases; 144/144	**LEVEL 3**	Shared matrices; no preprocessing; three metrics
Sa/Sq/Sz public GWY	12 records; 36/36	**LEVEL 3**	Algorithm track; parser observations separate
Nanoscope III .spm	6 files; 18/18 metrics	**LEVEL 2**	Partial; pre-freeze unblinding; no blind holdout
NanoSurf .nid mapping	Byte/orientation tests	**LEVEL 1**	Private corpus not distributed
Hertz/cone/DMT/maps	Synthetic recovery	**LEVEL 2**	No certified calibration or broad physical campaign
JKR/WLC/FJC/SLS	Synthetic recovery	**LEVEL 2**	JKR/SLS experimental; no physical campaign
KPFM, PSD, grains, resonance	Software tests	**LEVEL 1**	No frozen public physical-reference campaign
Fathom	Offscreen GUI tests	**LEVEL 1**	Qt/platform behavior varies

14.3 Gwyddion as reference

Gwyddion is an external software route only where a campaign declares its version, matrix handoff, wrapper, preprocessing, and tolerance. It is not universal ground truth. Retained summaries live at https://github.com/kegouro/spmkit-validation. Passing them does not constitute physical validation or feature parity.

14.4 Simulator

The cantilever simulator is **educational only**. It models a damped harmonic oscillator with additive noise. Not suitable for calibration or metrology.

15 Safety and interpretation warnings

- **Results require domain judgment.** High R^2 does not guarantee correct model choice.
- **Calibration values matter.** Young's modulus depends on tip radius (uncertain), spring constant (calibration), and contact model (approximation).
- **Check units.** `SPMChannel.unit` tells the physical unit.
- **Experimental readers may misinterpret metadata.** Format parsing is best-effort for reverse-engineered formats.
- **Fitted parameters are not automatically physical truth.**
- **SPM-Kit is not a medical, clinical, regulatory, or safety-critical instrument.**
- **Preserve original data.** SPM-Kit reads without modifying files.

16 Extensibility

16.1 Entry-point groups

Group	Purpose
`spmkit.plugins.v1`	Readers, analyses, domains
`spmkit.gui.modules`	Fathom modules (perspectives + panels)

16.2 Adding a format

Implement the `Reader` Protocol (`inspect + load`) and register via `spmkit.plugins.v1`.

16.3 Adding a perspective

Define a `ModuleSpec` with `PanelSpec` and `PerspectiveSpec`, register via `spmkit.gui.modules`. See `docs/extending.md` for full details.

17 Troubleshooting

`Missing GUI extra`
: `pip install "spmkit[gui]"`

`Qt platform plugin error`
: Install Qt dependencies or set `QT_QPA_PLATFORM=offscreen`.

`Unsupported format`
: Check extension matches `.nid/.nhf/.gwy/.jpk-force`. For experimental readers, install `afm` or `jpk`.

`Missing optional dependency`
: Install the relevant extra (e.g. `pip install "spmkit[grains]"`).

`Large force-volume memory`
: Use lazy loading or the `-fast` CPU vectorised path.

`GPU fallback`
: CuPy is not bundled. Install separately: `pip install cupy-cuda12x`.

`Export failure`
: Check write permissions. Use `-output` with a writable path.

18 Development and quality

```
git clone https://github.com/kegouro/spmkit
cd spmkit
pip install -e ".[all,dev,test-gui]"
pre-commit install
```

Quality checks (CI gate):

```
make check        # lint + types + tests
make lint         # ruff check
make type         # mypy (strict on core)
make test         # pytest with coverage
```

GUI tests (headless):

```
QT_QPA_PLATFORM=offscreen pytest tests/gui -q
```

Documentation:

```
pip install -e ".[docs]"
python -m mkdocs build --strict
```

19 SPM-Kit ecosystem relationship

- **SPM-Kit Core** reads demonstrated variants and performs numerical analysis.
- **Fathom** invokes Core through an interactive workspace.
- **Data Hunter** records public candidates and provenance; discovery is not validation.
- **Phantoms** generates deterministic truth-bearing synthetic fixtures independently of the analyzer.
- **Validation** runs frozen black-box campaigns against installed public interfaces.

Artifacts move through declared files, manifests, hashes, and commands. Not every located candidate is used, accepted, redistributable, or scientifically suitable, and there is no automatic pipeline from every Data Hunter record into Validation.

20 Citation, license, and contributions

20.1 How to cite

```
@software{spmkit2026,
  author       = {Jos\'e Labarca Baeza},
  title        = {spmkit: Open-source AFM/KPFM analyser
                  for scanning probe microscopy},
  year         = {2026},
  version      = {0.1.4},
  doi          = {10.5281/zenodo.21303280},
  url          = {https://github.com/kegouro/spmkit},
}
```

20.2 License

MIT. See `LICENSE` in the repository root.

20.3 Contributing

Issues and pull requests at https://github.com/kegouro/spmkit/issues. See `CONTRIBUTING.md`.

20.4 Funding

This open-source project has been developed without dedicated institutional funding.

20.5 Project status

Alpha (source `0.1.5.dev0`; GitHub release `0.1.4`; PyPI `0.1.2`). APIs may change before 1.0. Evidence and compatibility remain scoped.

21 Acknowledgements

José Labarca Baeza is the creator, author, and lead developer of SPM-Kit.

21.1 English

Tomás Corrales and the SPM Lab at Universidad Técnica Federico Santa María provided selected experimental datasets and laboratory context during the development and evaluation of SPM-Kit.

María Saavedra Fredes and Benjamin Schleyer helped locate and share candidate datasets for the validation campaigns.

21.2 Español

Tomás Corrales y el SPM Lab de la Universidad Técnica Federico Santa María proporcionaron datasets experimentales seleccionados y contexto de laboratorio durante el desarrollo y la evaluación de SPM-Kit.

María Saavedra Fredes y Benjamin Schleyer ayudaron a localizar y compartir datasets candidatos para las campañas de validación.

These acknowledgements do not imply that every located dataset was used, accepted, redistributable, or scientifically suitable. They do not constitute software authorship, institutional ownership, or endorsement.

A Quick reference card

```
# Install the documented source
python -m pip install "spmkit[gui] @ git+https://github.com/kegouro/spmkit@main"

# Inspect
spmkit info scan.nid

# Analyse
spmkit roughness scan.nid -c Z-Axis --level plane
spmkit psd scan.nid -c Z-Axis
spmkit analyze scan.nid -o ./results --tip-wf 4.8

# Figures
spmkit figure scan.nid -o topo.png --colormap batlow
```

```
# Convert
spmkit convert scan.nid scan.gwy

# Verify
spmkit verify scan.nid

# Force
spmkit forcecurve curve.jpk-force --model dmt --tip-radius 2e-8
spmkit forcemap volume.nid --fast -f maps.png
spmkit forcereport volume.nid -o report
spmkit forceexport volume.nid -o ./export
spmkit fbatch ./curves/ -o batch.csv

# Evaporation
spmkit evaporation ./tuning/ -k 0.3 -o evap.csv

# Grains
spmkit grains scan.nid --min-size 10

# GUI
spmkit gui [file]
```

B CLI command index

Command	Category	Input
info	Inspection	.nid, .nhf
roughness	Image	.nid, .nhf, .gwy
psd	Spectral	.nid, .nhf, .gwy
analyze	Pipeline	.nid, .nhf, .gwy
nanomech	Force	.nid (spectroscopy)
grains	Image	.nid, .nhf, .gwy
batch	Batch	Folder of SPM
evaporation	Resonance	Folder of .nid tuning
figure	Figure	.nid, .nhf, .gwy
convert	Format	.nid
verify	Audit	.nid
gui	GUI	— (optional file)
workspace	GUI	Force-curve file
forcecurve	Force	.jpk-force, .nid
forcemap	Force	.nid (force-volume)
forcereport	Force	.nid (force-volume)
forceexport	Force	.nid (force-volume)
fbatch	Batch	Folder of force curves
jkr	Force (exp.)	Calibrated curve

C Perspective index

Key	Label	Module	Type
image	Imagen	image	Image
grains	Granos	image	Image

`spectral`	Espectral	image	Image
`resonance`	Sintonía térmica	image	Image
`evaporation`	Evaporación	image	Image
`force`	Curva de fuerza	force	Force
`smfs`	SMFS	force	Force
`map`	Mapa	force	Force
`batch`	Batch	force	Force
`figure`	Figura	figure	Figure
`view3d`	Vista 3D	view3d	Image
`simulator`	Simulador	simulator	Educational

D Supported formats matrix

Extension	Source	Type	Access	Status
.nid	NanoSurf	Image + force	Read	Implemented, scoped
.nhf	NanoSurf HDF5	Images	Read	Experimental
.gwy	Gwyddion	Images	Read+Write	Interoperability
.spm/.00N	Bruker/Nanoscope	Images	Read	**Partial LEVEL 2**
.jpk-force/.jpk	JPK	Single curve	Read	Implemented, scoped
JPK tagged TIFF	JPK	Force data	Read	Experimental
.ibw/.h5/.tab	Adapter sources	Force data	Read	Experimental
.npz	Phantoms	Images	Read	Declared bundle

E Keyboard shortcuts

`Shortcut`	Action
`Ctrl+K`	Command palette
`Ctrl+O`	Open file
`Ctrl+S`	Save project
`Ctrl+M`	Calculate map
`Ctrl+Shift+R`	Generate report
`Ctrl+Shift+L`	Toggle theme
`Ctrl+Shift+A`	Appearance settings
`Ctrl+Left / Ctrl+Right`	Previous / next curve
`Ctrl+1 ... Ctrl+5`	Tab switch (legacy app)