Calango

User Guide

Visual atomistic modeling,
simulation and analysis

Leandro Seixas

Version 26.8.46

About this guide

Calango is a desktop application for building, visualizing, simulating and analysing atomic structures. It pairs a C++20 / OpenGL core — fast enough to orbit tens of thousands of atoms in real time — with an embedded CPython interpreter running the Atomic Simulation Environment (ASE), so every structure you see on screen can be handed to a real calculator without leaving the program.

This guide is written for the person using Calango, not the person building it. It walks through the workspace, then covers each menu in the order you are likely to need it: getting data in, building and editing structures, making them look right, running simulations locally or on a cluster, and analysing the results. Chapter 17 is a reference you can jump straight to — keyboard shortcuts, the complete menu map, file formats and troubleshooting.

Conventions

Analysis ⟩ Raman Modes A menu path. Each ⟩ step is one level deeper.

Compute A button, field, checkbox or tab label exactly as it appears in the interface.

Ctrl+R A keyboard shortcut. On macOS, Ctrl means Cmd wherever the shortcut is a standard editing action (open, save, quit, undo); single-letter viewport shortcuts have no modifier on any platform.

`run.py` A file, directory or path.

Three kinds of callout appear throughout:

> **Note**
> Background you do not strictly need, but that explains why something behaves the way it does.

> **Tip**
> A shortcut or working habit that saves time.

> **Requires**
> An external dependency — a Python package, a solver binary, a network connection — that the feature cannot work without.

Version

This guide documents Calango **26.8.46**. The version you are running is shown in **Help ⟩ About Calango**, together with the Python and ASE versions actually in use — worth checking first whenever something behaves unexpectedly.

Contents

CHAPTER 1

Getting Started

1.1 Installing Calango

Calango ships as a native installer for each desktop platform:

macOS A drag-and-drop `.dmg`. Mount it and drag `calango.app` onto the `Applications` shortcut.

Linux A Debian package. Install it with `apt` so dependencies resolve automatically:

```
sudo apt install ./calango_<version>_amd64.deb
```

The package registers a desktop launcher, an application icon, and a MIME type for `.calproj` files, so double-clicking a project in your file manager opens the workspace.

Building from source is covered in the companion *Calango Packaging Guide* (`docs/packaging.pdf`), which also documents how the installers are produced.

1.2 The Python environment

Calango embeds a CPython interpreter and drives ASE through it. Almost everything that touches chemistry — reading a CIF, building a slab, running a calculation — goes through that interpreter, so pointing Calango at a Python that has ASE installed is the single most important piece of setup.

1.2.1 How the interpreter is chosen

An embedded interpreter does not inherit an activated virtual environment the way a shell does, so Calango resolves one explicitly, in this order:

1. `$CALANGO_PYTHON` — an explicit path to a Python executable. Highest priority; use this to override everything else.
2. `$VIRTUAL_ENV` — an activated virtual environment, if Calango was launched from that shell.
3. A Python bundled inside the installed application (`Contents/Resources/python` on macOS, `<prefix>/lib/calango/python` on Linux), when the installer was built with one.
4. The interpreter the build was configured against.

Requires

Calango needs `ase` and `numpy` at minimum. Individual features add their own requirements: `spglib` (symmetry, Raman modes), `paramiko` (remote access), `pillow` / `imageio` / `imageio-ffmpeg` (animation export), `mace-torch` (ML potentials), `gpaw` (DFT bands and PDOS). A complete environment:

```
python3 -m venv .venv
.venv/bin/pip install ase numpy spglib paramiko pillow imageio \
                      imageio-ffmpeg
```

1.2.2 Checking the environment

Before opening any structure, confirm the interpreter is the one you expect:

```
calango --probe-python
```

This runs headlessly and prints the resolved interpreter, the Python version and the ASE version, exiting with status 0 only when ASE imports successfully. It is the first thing to run when structure loading fails.

If ASE cannot be imported at launch, Calango still starts — so you can adjust settings — but structure I/O and job features are disabled, and a warning explains how to point at a working interpreter.

Tip

Simulation jobs do not have to use the embedded interpreter. The calculator, phonon and band-structure dialogs each carry an **Execution Environment** selector that routes the job to any Conda environment or interpreter path (Section 9.3). This is how you keep, say, a GPAW environment separate from the one Calango itself uses.

1.2.3 The environment file

At launch Calango reads an environment file — `~/.env` by default — and exports `MP_API_KEY`, the Materials Project API key, into the process environment. A key already present in your shell takes precedence at startup.

Point Calango at a different file, or re-read it on demand, in **Edit ⟩ Preferences**. The dialog reports whether the file exists and whether it actually defines a key. The same controls are mirrored in the **Materials Project** tab of the database browser.

1.3 First launch

Calango opens with an empty workspace. Three quick ways to get something on screen:

1. **File ⟩ Open ⟩ Structure** and pick any structure file (Section 5.1 lists the formats).
2. **Build ⟩ From Database** and load one of the bundled presets — diamond, bulk and monolayer MoS_2, graphene, benzene, naphthalene, coronene.
3. Pass a file on the command line: `calango structure.cif`. Each file argument opens in its own tab; a `.calproj` argument restores a whole saved session.

CHAPTER 2

Tutorial: silicon from start to finish

This tutorial walks the whole arc of a materials study on one small, well-understood system: crystalline silicon. You will build the crystal, relax it, converge a ground state, compute the phonon dispersion, and finish with the electronic band structure and projected density of states.

Silicon is the right first system precisely because the answers are known. Every stage below ends with a number you can check, so when something goes wrong you find out at that stage instead of three stages later.

> **Requires**
> GPAW is used throughout. Install it into a Conda environment and point Calango at it once, in **Edit** ⟩ **Preferences** ⟩ **Python & Environments**. Every wizard then binds to that environment automatically. Without GPAW you can still follow the tutorial with the **EMT** calculator — the workflow is identical — except for the absence of electronic structure calculations.

2.1 Step 1 — Build the crystal

Open **Build** ⟩ **From Database...** and stay on the **Bulk** tab. It is already set up for this system:

Build from `Prototype (ase.build.bulk)`.

Formula `Si`.

Crystal structure `diamond`.

Lattice constant a leave at `0` to accept ASE's tabulated value, 5.43 Å.

Cell shape leave unchecked for the two-atom primitive cell.

Click **Build Crystal** and the crystal opens in a new workspace tab. The **Structure** dock on the left should now read `Si2`, 2 atoms, with $a = b = c = 3.84$ Å and $\alpha = \beta = \gamma = 60°$: that is the rhombohedral primitive cell of the diamond lattice, not an error. The conventional cubic cell with its 8 atoms and 90° angles is what you get by ticking **Conventional cubic cell**.

> **Tip**
> Work in the primitive cell. Every stage after this one scales with the number of atoms — the phonon step worst of all, since it builds a supercell of whatever you hand it. Eight atoms instead of two would make that step roughly an order of magnitude slower for identical physics.

2.2 Step 2 — Relax the geometry

Silicon's tabulated lattice constant is not PBE's lattice constant, so the first job is to let the calculator find its own minimum.

Open **Simulation** ⟩ **Geometry Optimization...**. The wizard runs in four stages; the **Next** button carries you through them.

1. **Calculator & Execution Environment.** Choose **GPAW**. The Conda environment is filled in from Preferences.
2. **Calculator Settings.** For this system:
   - **Mode** `PW`, **Plane-wave cutoff** `400 eV`
   - **XC functional** `PBE`
   - **k-points** `8 × 8 × 8`
3. **Optimization Settings. Force convergence (fmax)** `0.01 eV/Å`. Silicon's primitive cell has no free internal coordinates by symmetry, so this converges almost immediately.
4. **ASE Script Review.** Read the script — it is the whole job, and it is editable. Press **Run**.

The **Processes** dock tracks the run and the **Results** dock plots energy against step as it goes. When it finishes, the relaxed structure loads back into the tab and **Results** ⟩ **Geometry Optimization Viewer...** shows the energy and force convergence history.

> **Note**
> Relaxing *positions* will not change silicon's lattice constant — BFGS moves atoms, not the cell. Getting PBE's lattice constant (about 5.47 Å, some 0.7 % above experiment, which is PBE behaving normally) means an energy–volume scan: build several cells with **Build** ⟩ **Supercell...** or the structure editor, run a single point on each, and fit the minimum. This tutorial keeps the tabulated constant so the later stages stay comparable to published numbers.

2.3 Step 3 — Converge the ground state

Simulation ⟩ **Single-point Calculation...** (Ctrl+R) now does double duty. It reports the total energy, and — because the calculator is GPAW — it writes `single_point.gpw`, the converged density and wavefunctions.

Use the same calculator settings as Step 2, but raise the k-grid to `12 × 12 × 12`. Run it.

> **Note**
> This file is the *baseline* that the later analysis workflows inherit. Electronic Structure, Optics, Wannier Functions, GW and the charge density difference all load a completed ground state and evaluate at fixed density rather than re-converging their own. That is deliberate: a spectrum computed from a silently different SCF solution than the one you inspected is not attributable to anything. It also means those wizards will refuse to open

until this step exists — if one tells you it needs a baseline, this is the step you skipped.

Results ⟩ Single-Point Viewer... shows the total energy, the Fermi level and the maximum residual force.

2.4 Step 4 — Phonon dispersion

Open **Simulation ⟩ Phonon...**. This wizard has four stages.

1. **Calculator & Execution Environment.** GPAW again.
2. **Calculator Settings.** As before.
3. **Phonon Settings.**
   - **Displacement** δ `0.01` Å — small enough to stay harmonic, large enough to lift the forces clear of numerical noise.
   - **Supercell** `3` × `3` × `3`. The supercell sets how far force constants are sampled, and therefore how much of the dispersion is real rather than interpolated. $2 \times 2 \times 2$ runs faster and already gives the right shape.
   - **Symmetry-reduced displacements on**. The naive scheme costs $6N$ force evaluations; silicon's space group relates almost all of them, and spglib (through phonopy) reduces the set by roughly an order of magnitude for identical force constants.
   - **Remove residual forces on**. A relaxation stops at a finite `fmax`, so the reference geometry still carries small forces; left in, they add a spurious linear term that shows up as non-zero acoustic frequencies at Γ.
   - **Enforce acoustic sum rule on**.
4. **q-Path Definition.** The embedded Brillouin-zone builder suggests silicon's conventional path, $\Gamma \to X \to W \to K \to \Gamma \to L$. Accept it.

This is the longest step of the tutorial — it is many force evaluations on a supercell. When it completes, the **Phonon Viewer** opens on its own.

2.4.1 Checking the answer

Two atoms in the primitive cell give $3 \times 2 = 6$ branches: three acoustic and three optical. Check these before reading anything else into the plot:

Acoustic branches vanish at Γ. All three should start at zero. Residual frequencies of a few cm^{-1} are normal; tens of cm^{-1} mean the acoustic sum rule or the residual-force subtraction is not doing its job, or the relaxation was too loose.

The optical mode at Γ sits near 15.5 THz (520 cm^{-1}). This is silicon's Raman line, one of the best-measured numbers in solid state physics. Within a few percent means the calculation is sound.

No imaginary frequencies. Plotted as negative, they would mean the structure is not at a minimum — for silicon that indicates a problem with the setup, not a real instability.

Analysis ⟩ **Phonon Thermodynamics...** integrates the density of states into internal energy, free energy, entropy and heat capacity over 0–1000 K. Two checks come free: C_V must approach the Dulong–Petit limit $3Nk_B$ at high temperature, and the entropy must go to zero at 0 K.

2.5 Step 5 — Electronic structure

Open **Simulation** ⟩ **Electronic Structure...**. Stage 1 asks for a **Baseline SCF density** — the `single_point.gpw` from Step 3. Select it.

The run is non-self-consistent by construction: it loads that density and evaluates the bands along the k-path at fixed density. Set the k-path the same way as for the phonons ($\Gamma \to X \to W \to K \to \Gamma \to L$) and enable **PDOS** to get the element- and orbital-projected density of states beside the bands.

When it finishes, the band/PDOS viewer opens: bands as $E - E_F$ against k-distance on the left, the PDOS sharing the energy axis on the right.

2.5.1 Checking the answer

The gap is indirect. The valence-band maximum sits at Γ; the conduction-band minimum does not — it lies along $\Gamma \to X$, about 85 % of the way to X. A band structure showing silicon's minimum at Γ is wrong.

The gap is too small, and that is expected. PBE gives roughly 0.6 eV against an experimental 1.17 eV. This is the standard band-gap underestimation of semilocal DFT, not a convergence failure. Raising the cutoff or the k-grid will not fix it, because it is not an error in the calculation.

The PDOS is sp^3. The valence band is a mixture of Si s and p character, as tetrahedral bonding requires.

> **Tip**
> To correct the gap rather than accept it, run **Simulation** ⟩ **GW Calculations...** on the same baseline. One-shot G_0W_0 replaces the DFT exchange–correlation potential with the self-energy and opens semiconductor gaps by typically 0.5–2 eV. The **GW Viewer** reports the DFT gap, the quasiparticle gap and the renormalization between them.

2.6 Where to go next

The same baseline supports the rest of the analysis suite without recomputing the ground state:

- **Simulation** ⟩ **Optics...** — the dielectric function $\varepsilon(\omega)$ and the absorption, reflectivity, refractive-index and loss spectra derived from it.
- The **ELF** checkbox under **Density Exports** in the single-point setup — the covalent bond charge between neighbours, written as `elf.cube` by the same SCF and rendered as an isosurface in the **Volumetric Data** dock. There is no separate ELF module: the field comes out of the ground state you already have.
- **Analysis** ⟩ **Charge Density Difference (CDD)...** — $\Delta\rho = \rho_{A+B} - \rho_A - \rho_B$ between two

fragments of the same run, showing where the charge went when they were put together.

- **Wannier Functions ⟩ Wannierization…** — the four sp^3 bond orbitals of the diamond lattice.
- **Analysis ⟩ Raman Modes…** — the factor-group analysis that labels silicon's Γ optical mode T_{2g} and Raman-active.

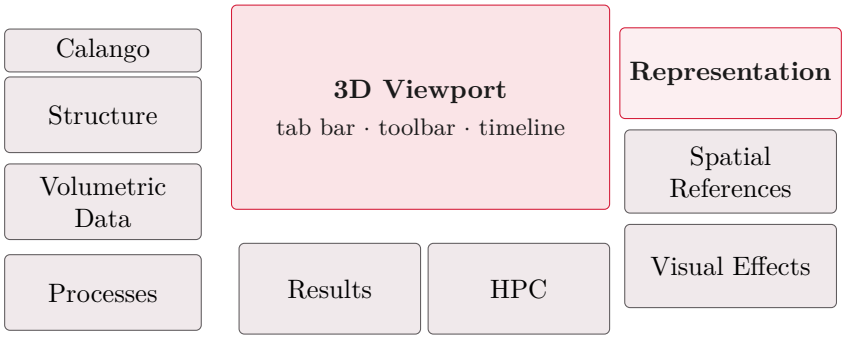
Save the workspace with **File ⟩ Save** before you stop. Job directories of a saved project are kept alongside the `.calproj`, so every result above stays linked in the **Processes** dock and can be reopened without recomputation.

CHAPTER 3

The Workspace

3.1 The default layout

Calango opens into two full-height side columns flanking the 3D viewport, with a short row of job panels between them. Every panel is a Qt dock widget, so the layout is a starting point rather than a constraint — any panel can be resized, re-docked, tabbed with another, floated onto a second monitor, or hidden entirely.



Calango The branding strip heading the left column. Purely decorative; hide it from **View** if you want the vertical space.

Structure Formula, atom and bond counts, lattice parameters, cell volume and periodicity, plus **Edit Structure...** (Chapter 7).

Volumetric Data Scalar fields — charge density, ELF, Wannier orbitals — as isosurfaces and colour slices (Section 15.10).

Processes The process manager (Section 3.5): every background task with live status.

the viewport The document tab bar, the frame toolbar, the 3D canvas, and the playback timeline (visible only for trajectories). Covered in Chapter 4.

Results Console and live metric plots for the running calculation (Section 9.4).

HPC SSH connection, cluster job submission and queue monitoring (Chapter 11). It opens at the narrowest width that still shows its whole form.

Representation Appearance controls — style, mode, color mapping, radii and bond widths (Chapter 8).

Spatial References Four tabs: the cell wireframe, the orientation triad, the per-atom vector overlay, and (last, since it draws a whole extra object into the scene rather than an overlay on

the structure the way the other three do) the ground plane. All four draw something onto the scene that is not the atoms.

Under **Vectors**, **Vector scale** and **Vector width** set the arrows' length and thickness. Arrowheads are always drawn — a plain shaft does not say which way a vector points, which is the one thing the overlay exists to state; thinning the arrows is what relieves the clutter on a dense magnetic structure. **Vector color** carries an **Opacity** spin box on the same row, in percent; like the color, it is remembered PER OVERLAY, so switching from Force to Velocity restores that property's own fade rather than carrying one over.

Under **Unit cell**, **Show atoms of the neighboring unit cell** draws exactly the periodic images that terminate a bond leaving the cell, so no bond ends in mid-air. Only those: duplicating every atom on a face filled the view with periodic repetition rather than information. Both **Cell color** (the wireframe) and **Cell fill color** carry their own percent **Opacity** spin box on the same row as the color picker — the same composite control the Vectors and Floor tabs use, so all four translucent surfaces in this dock express opacity identically.

Under **Floor**, **Normal** comes first — orientation, then position along it, which is also the order **Height** depends on, since Height moves the plane along whatever direction Normal names. **Height** raises or lowers the ground plane relative to its automatic position with a linked slider and spin box — the slider's own range scales with the structure's size, and the box beside it accepts anything further out than the slider currently reaches. It moved here from **Visual Effects**, where it sat as its own tab, because the plane is a reference surface drawn in the scene, the role the cell wireframe and the triad also play, not a post-process image filter. **Opacity**, now also a percent spin box, sits beside **Color** on the same row; the orientation is set by **Normal** alone — an earlier **Plane** preset dropdown beside it was removed as a pure read-out of the same value, never a second setting of its own.

Visual Effects Five tabs: **Light**, **Shadow**, **Fog**, **Blur** and **SSAO**.

> **Tip**
> Your layout is saved when you quit and restored on the next launch, including floating panels and window geometry. Every panel has a toggle at the bottom of the **View** menu, so a panel you closed by accident is one menu click away — and **View** ⟩ **Reset Layout** puts everything back where this diagram shows it.

> **Note**
> After an upgrade that changes the default arrangement, Calango discards the saved layout once so the new default appears; any rearrangement you make after that persists as usual.

3.2 Documents, tabs and the timeline

Each structure or trajectory you open occupies its own *tab* above the viewport, and each tab carries its own undo history. All tabs share one accelerated viewport — switching tabs changes which document the views observe, so display settings stay consistent and no OpenGL context is recreated.

When a document contains more than one frame, the **playback timeline** appears below the viewport:

Transport buttons first, previous, play/pause, next, last.

Scrubber a tick-marked slider over the frame range.

Rate playback speed in frames per second, 0.1–120, default 15.

Trajectories arrive from several places: opening a multi-frame file, finishing an MD or relaxation run, generating displaced structures in the phonon builder, or producing a noise ensemble. During a running simulation the timeline grows in real time as frames stream in (Section 9.5).

3.3 The Structure panel

The **Structure** panel reports what Calango knows about the active document:

Formula Hill-ordered chemical formula.

Atoms, Bonds Counts, with bonds as currently perceived.

a, b, c Cell vector lengths in Å, to two decimals.

α, β, γ Cell angles in degrees.

Cell volume, Periodic Volume and per-axis periodicity.

Tolerance The spglib symmetry tolerance (*symprec*) used for the three fields below, in Å, up to four decimals. Default 0.001. Raise it for structures with numerical noise — a relaxed cell often needs 0.01 or more before its true symmetry is recognised.

Space group, Point group, Crystal system Detected symmetry, e.g. `Fd-3m (227)`, `m-3m`, `cubic`.

> **Requires**
> Symmetry detection needs `spglib` and a defined unit cell.

3.4 Projects and session storage

A *project* is one `.calproj` file that restores an entire session: every tab with its structures and trajectory frames, the viewport colour mapping, and the job console with its recorded metric series.

File ⟩ Project Workspace ⟩ Open Project Ctrl+Shift+O. Replaces the current session, warning first if tabs are open.

File ⟩ Project Workspace ⟩ Save Project Ctrl+S.

File ⟩ New Workspace Ctrl+N. Closes everything and starts clean.

Calango tracks unsaved changes. Every undoable edit, tab close and job launch marks the session dirty; saving or loading a project clears the flag. Quitting with unsaved changes asks whether to **Save**, **Discard** or **Cancel** — cancelling (or cancelling the save dialog that follows) leaves the application open.

3.4.1 The managed temporary directory

Simulation jobs and generated ensembles need somewhere to put their working files. Once a project has been saved, Calango places them in a `.calango_tmp/` folder *next to the `.calproj` file*, one timestamped subdirectory per task:

```
my_study.calproj
.calango_tmp/
    job_20260721_143255/     run.py, structure.extxyz, md.traj, logs
    noise_20260721_144012/   perturbed.extxyz
    job_20260721_150130/     bands.json, pdos.json
```

Until a project has been saved, the same directories are created under the per-user application data location instead. Either way the process manager keeps a live link to each one.

3.5 The process manager

The compact **Processes** panel in zone 5 lists every background task of the session — local calculations, remote submissions, band-structure runs, noise ensembles — with its name, colour-coded status (*queued*, *running*, *completed*, *failed*) and start time.

Two buttons act on the selected task:

Open Folder Reveals the task's working directory in your file manager, for the raw logs and any files Calango does not import.

Load Result Brings the result back into the workspace. Double-clicking the row does the same. Calango inspects the directory and opens whatever it finds: band structure and PDOS data open in the electronic-structure viewer, trajectories and final geometries open as tabs.

The point of keeping these links is that a finished calculation stays one click from further analysis. A completed MD run can be re-loaded weeks later and fed to the RDF, distribution or dataset tools without recomputing anything.

CHAPTER 4

The 3D Viewport

The viewport is an OpenGL 3.3 core-profile canvas using instanced rendering — one draw call per mesh type — so structures with tens of thousands of atoms stay interactive.

4.1 Interaction modes

What a left-drag does depends on the active *interaction mode*. The six modes are the first group of buttons on the frame toolbar, each with a single-letter shortcut:

Key	Mode	Left mouse button
R	Rotation	Drag orbits the camera around the structure. The default.
T	Translation	Drag pans the scene.
S	Selection	Drag draws a rubber-band box and selects every atom inside it.
I	Insertion	Click empty space to add an atom; drag from one atom to another to bond them.
D	Distance	Click two atoms to measure their separation.
A	Angle	Click three atoms (vertex second) to measure the angle.

Three gestures work in *every* mode, so navigation never requires switching back:

Middle-drag, or Shift+left-drag Pan.

Wheel Zoom.

Double-click Reframe the structure.

The mouse cursor changes with the mode — an arrow for rotation, an open hand for panning, a crosshair for selection, insertion and measurement.

4.2 Selecting atoms

A left *click* (no drag) picks the atom under the cursor by ray-cast, in any mode. Ctrl+click (Cmd+click on macOS) toggles an atom in or out of the selection, so you can build up a set one atom at a time. Clicking empty space clears the selection.

In **Selection** mode a rubber-band drag selects every atom whose centre falls inside the box; holding Ctrl adds the boxed atoms to the existing selection instead of replacing it. With the viewport focused, Delete or Backspace removes the selected atoms and the bond network is rebuilt immediately.

The status bar reports the selection size, and several tools act on it: the bond-order buttons need exactly two atoms, the bond editor can pull a pair straight from the selection, and **Edit** ⟩ **Selection** can change the element of, or translate, whatever is selected.

4.3 Measuring distances and angles

Measurement modes accumulate clicks on atoms and draw the result directly on the canvas — markers on the chosen atoms, dashed connecting lines, and the value in a floating label. The same value is written to the status bar, for example:

```
Distance Si(0) - Si(4): 2.351 A
Angle O(2) - Si(0) - O(5): 109.47 deg
```

Distance Two atoms; result in Å to three decimals.

Angle Three atoms; the *second* atom clicked is the vertex. Result in degrees to two decimals.

Clicking empty space clears the current measurement, and clicking again after a completed one starts fresh. Dragging still orbits the camera, so you can rotate the structure between clicks to reach an atom hidden behind another. Measurements are cleared automatically when you switch modes or load a different structure.

4.4 Fixed-angle rotations

The toolbar's rotation cluster turns the scene by exact increments about the world axes — useful for reproducible figure orientations.

Angle step An editable spin box, 1–180°, default 15°.

X+ X− Y+ Y− Z+ Z− Counter-clockwise and clockwise rotation about each axis by exactly that step.

Each click animates smoothly over about 200 ms, and rapid clicks compose exactly: four presses of **Z+** at 90° return the scene to where it started. The axes triad follows the rotation, and picking, selection and measurements all stay consistent with what you see.

4.5 Framing, alignment and projection

F Frame the structure: centre it and back the camera off far enough to see all of it. Also on the toolbar and in **View** ⟩ **Alignment**.

Align with XY / XZ / YZ Look straight down the remaining Cartesian axis. Toolbar buttons, or **View** ⟩ **Alignment**. Alignment clears any accumulated fixed-angle rotation, so the result is a true axis-aligned view.

O Toggle orthographic versus perspective projection. The orthographic frustum is matched to the perspective field of view at the camera's target distance, so the structure keeps its apparent size across the switch — and zoom keeps working in both.

> **Tip**
> Orthographic projection is what you want for figures where lattice directions must read as parallel: perspective foreshortening makes a supercell look subtly sheared.

4.6 Inserting atoms and bonds

In **Insertion** mode (I) the toolbar's element button — which shows the active symbol, `C` by default — selects what gets placed. Click it to choose any element from the graphical periodic table.

Click on empty space Inserts one atom of the active element there. The atom lands on the plane through the camera's target point facing the viewer, so what you click is where it appears.

Drag from one atom to another Creates a bond between them, added as a manual bond override.

Click on an existing atom Selects it, as in any other mode.

Both operations are single undo steps.

> **Note**
> Because inserted atoms land on the camera-target plane, the depth of a new atom depends on the current view. Place atoms from an aligned, orthographic view when the exact coordinate matters, or insert roughly and then set exact coordinates with **Edit** ⟩ **Add Atom**, which takes numeric x, y, z.

4.7 Visual effects

View ⟩ **Visual Effects** toggles the **Visual Effects** dock — the scene stays interactive while you tune it. It is a five-tab panel running from what lights the scene, through the scene itself, to the passes that post-process the finished image: **Light** (the studio lighting rig, Chapter 8), **Shadow**, **Fog**, **Blur** and **SSAO**. Every effect defaults to off. The ground plane used to sit here as a **Floor** tab; it now lives in the **Spatial References** dock instead (Chapter 8), since it draws a whole extra object into the scene rather than an image filter.

4.7.1 Shadows

Real-time shadow mapping with percentage-closer filtering: atoms and bonds cast shadows onto neighbouring geometry, including the floor when it is on. Off by default; adds roughly one extra depth-only pass per frame.

Intensity 0.00–1.00. How much direct light an occluded surface loses — ambient light is never attenuated, so even at 1.00 shadowed geometry stays readable rather than going black.

Softness / blur radius 0–6 texels. The PCF blur radius: 0 gives hard, aliased edges; each step averages a wider neighbourhood, so quality and cost both rise together. 2–3 suits most structures.

The shadow projection follows the **primary light** — the first entry under the **Light** tab. Re-aiming that light re-aims the shadows; fill lights stay unshadowed, which is what keeps occluded regions legible.

4.7.2 Distance fog

Fades distant geometry into the background colour, which makes the depth ordering of a thick slab or a large nanoparticle immediately readable.

Mode Linear (start → end) fades uniformly between two distances; **Exponential (density)** follows $e^{-\rho d}$.

Start distance, End distance The linear ramp, in Å. Defaults 15 and 80.

Density The exponential coefficient, default 0.03.

The fog colour tracks the viewport background automatically, so geometry always fades into whatever backdrop you have chosen.

4.7.3 Depth of field

A post-processing pass that blurs geometry away from the focal plane, in proportion to its circle of confusion.

Blur strength Maximum blur radius in pixels, 1–20.

Focus range The depth band that stays sharp, in Å.

Focus offset Shifts the focal plane relative to the camera target, in Å.

> **Note**
> Depth of field renders the scene into an offscreen buffer before compositing, which means multisample anti-aliasing is unavailable while it is enabled. For final figures, decide which of the two matters more — crisp edges or the depth cue.

CHAPTER 5

Getting Data In and Out

5.1 Supported formats

All structure I/O goes through `ase.io`, so Calango reads and writes every format ASE knows. The file dialog offers the common ones directly:

Family	Extensions
XYZ	`.xyz`, `.extxyz`
Crystallography	`.cif`
VASP	`POSCAR`, `CONTCAR`, `.vasp`
ASE	`.traj`
Quantum ESPRESSO	`.in`, `.pwi`, `.pwo`, `.out`
CASTEP	`.cell`
LAMMPS	`.data`, `.dump`, `.lammpstrj`
Gaussian	`.gjf`, `.com`
SHELX	`.res`

> **Note**
> Extended-XYZ per-atom arrays are imported as *scalar fields* and *vector fields*. A `charges` or `forces` column becomes selectable in the **Custom property** colour mode, and `forces` or `momenta` enable the force and velocity arrow overlays. This is the simplest route for visualising per-atom data computed elsewhere: write it as an extxyz column and open the file.

5.2 Opening structures and trajectories

File ⟩ **Open** ⟩ **Structure** Ctrl+O.

File ⟩ **Open** ⟩ **Trajectory** Ctrl+T.

Both entries read *every* frame in the file. A single-frame file opens as a plain structure tab; a multi-frame file opens as a trajectory with the timeline active. In practice the two entries differ only in the file filter, so either works for either kind of file.

Opening a `.calproj` file — from the command line, your file manager, or the structure dialog — restores the saved session rather than loading a geometry.

5.3 Saving structures and trajectories

File ⟩ Save ⟩ Structure As Ctrl+Shift+S. Writes the currently displayed frame. The name is suggested from the document's own, with the default extension; the format follows the filter you pick, and a name typed without an extension is given the filter's own.

File ⟩ Save ⟩ Trajectory As Ctrl+Shift+T. Writes all frames of the active document as extended XYZ, multi-frame XYZ, ASE `.traj`, or multi-model PDB.

> **Note**
> **Extended XYZ is the default everywhere.** It is pre-selected in every open and save dialog in the application — structures, trajectories, NEB endpoints, dataset imports — and it is what **Save Structure As** writes unless you choose otherwise. The reason is that it is the only format in the list that round-trips everything a Calango document carries: cell and periodicity, per-atom magnetic moments and charges, forces and velocities, and arbitrary extra columns. Plain `.xyz` silently drops the cell; CIF drops the calculator results; POSCAR drops nearly everything but the positions.
> Every other format is still one entry down in the same list, so nothing has become harder to reach — only the *default* changed, to the format that loses nothing.

5.4 The database and preset browser

Build ⟩ From Database opens a two-tab browser.

5.4.1 Presets

Seven benchmark structures ship with Calango, each annotated with a suggested calculator:

Preset	File	Suggested calculator
Diamond (bulk C)	`diamond.vasp`	MACE-MP-0
MoS_2 2H (bulk)	`mos2_2h_bulk.vasp`	MACE-MP-0
Graphene monolayer	`graphene.vasp`	MACE-MP-0
MoS_2 1H (monolayer)	`mos2_1h_monolayer.vasp`	MACE-MP-0
Benzene	`benzene.xyz`	MACE-OFF (or EMT)
Naphthalene	`naphthalene.xyz`	MACE-OFF
Coronene	`coronene.xyz`	MACE-OFF

Select a row and press **Load Selected**, or double-click it.

5.4.2 Materials Project

The second tab fetches structures by Materials Project ID.

API key Masked field, pre-filled from your saved setting or from `MP_API_KEY` in the environment. It is stored as you type.

Material ID For example `mp-149` (silicon). Pressing Enter triggers the fetch.

.env file Shows the current environment file path, with **Browse...** to point elsewhere and **Reload** to re-import the key (Section 1.2.3).

Fetch Structure Downloads the entry and opens it in a new tab, reporting the formula and atom count.

Requires

Needs a network connection and a personal API key from https://materialsproject.org/api.

CHAPTER 6

Building Structures

Everything in the **Build** menu produces a new structure, either replacing the active one or opening in a new tab. All of it runs through ASE, so a working Python environment is a prerequisite throughout this chapter.

6.1 Molecular Design — the 2D sketcher

The **first button on the viewport toolbar** — a benzene hexagon, ahead of the rotation-mode button — opens **Molecular Design**, a chemical drawing program inside Calango. You sketch a molecule the way it is drawn on paper and send it to the 3D viewport as a real structure. It is the one entry in this chapter that needs *no* Python environment: the sketch model, the SMILES parser and the 2D → 3D conversion are all native C++.

The window is **modeless**, so the 3D viewport stays visible beside it, and it keeps its drawing for the rest of the session.

6.1.1 The three zones

A tool palette on the left (a two-column grid of icon-only buttons), the drawing canvas in the middle, and a compact output/appearance sidebar on the right.

6.1.2 Drawing

Select a bond tool and drag between two atoms to bond them, or into empty space to create a new atom on the far end. A plain click on an atom grows a bond in whichever direction around it is emptiest, so clicking the same carbon repeatedly walks out a chain. The free end snaps to the **30° family** at the standard bond length.

Drawing onto an existing bond cycles its order — single, double, triple, single. The **Wedge** and **Hash** tools mark stereochemistry with the narrow end on the atom the drag starts from; clicking a bond that already carries that mark reverses it.

Carbon vertices are not labelled — a line junction is a carbon. Heteroatoms show their symbol and their hydrogen count (OH, NH_2). Those hydrogens are implicit: counted, not drawn, taken from the standard valences (C 4, N 3, O 2, S 2/4/6, P 3/5, halogens 1), with the smallest valence that fits the drawn bonds winning. A formal charge shifts the valence the conventional way.

An atom whose bonds exceed every valence its element has — a pentavalent carbon — is **circled and left alone** rather than refused. Chemists sketch intermediates.

The ring palette covers cyclopropane through cyclooctane, benzene, cyclopentadiene and naphthalene. Clicking empty space stamps a ring; clicking an existing bond *fuses* the ring onto

it, reusing that bond's two atoms — two benzenes fused this way give naphthalene, with the ring-junction carbons at the right valence.

The **Chain** tool drags out a zig-zag alkyl chain with a live length count; **Tidy** regularizes bond lengths, angles and ring geometry over the selection, or over the whole canvas when nothing is selected.

6.1.3 SMILES

The **SMILES** field reads and writes. It shows the SMILES of the drawing as you work, and typing a string draws that molecule. The parser is native and covers the **organic subset plus bracket atoms**: bare `B C N O P S F Cl Br I` and their aromatic forms, bracket atoms with a hydrogen count and formal charge, the bond symbols, branches, ring closures including `%nn`, and aromatic rings kekulized on import. Stereochemistry, isotope labels and atom maps parse and are then **dropped**; wildcards and SMARTS syntax are refused. A refused string leaves the canvas untouched and says what was wrong and where.

6.1.4 Send to 3D Viewport

Builds a 3D structure and opens it in a new tab named after the formula, through the same import machinery every builder uses. **The selection wins**: with atoms selected only those are sent, otherwise every fragment on the canvas goes.

Implicit hydrogens become real atoms and the geometry is relaxed against a **small molecular-mechanics force field written for this module** — bond stretching from the covalent radii and bond order, angle bending from the coordination, an sp^2 planarity restraint, a torsion term that keeps double bonds coplanar, and soft non-bonded repulsion. Perceived aromatic rings get one bond length for the whole ring rather than the alternating pair a Kekulé structure would otherwise relax to.

> **Accuracy**
>
> This is a **clean-up, not a converged geometry**. It reproduces bond lengths to about 0.02 Å and angles to a couple of degrees for ordinary organic connectivity, and gets planarity and ring geometry right. It has no electrostatics, no hydrogen bonding, no dispersion, and it is not a conformational search. Run **Simulation** ⟩ **Geometry Optimization** on the new tab for a real geometry.

Benzene, as the reference case: C_6H_6, planar to within 10^{-3} Å RMS, all six C–C bonds equal at **1.391 Å** (experiment 1.397 Å), C–H at **1.070 Å** (experiment 1.084 Å), every C–C–C angle 120.000°.

6.1.5 Export and appearance

Export image saves a PNG at 3× for print or a resolution-independent SVG, optionally with a transparent background. The appearance controls set bond width, label size, element colours on/off (off gives a monochrome drawing, which is what most journals want), and whether the canvas follows the application theme — by default it does not, staying **white** like every other 2D figure in Calango.

6.1.6 Highlights

Two soft fills, both painted *under* the structure and both **annotations rather than chemistry**: they go into an exported image, they survive undo, copy and paste, and they are not part of what **Send to 3D Viewport** exports.

Aromatic rings (off by default) fills every ring the model perceives as aromatic. The rule is derived, never stored — the sketch holds Kekúlé bond orders only — and deliberately conservative: a five- or six-membered ring qualifies when every member contributes to a closed π system (one ring double bond, or a heteroatom lone pair with none) and the total is $4n+2$. Benzene, pyridine, pyrrole, furan, thiophene and both rings of naphthalene fill; cyclohexene and cyclopentadiene do not.

To colour an arbitrary **region**, select atoms and click one of the six palette swatches. Several differently-coloured regions can coexist. The colour lives on the atom, so a bond is coloured exactly when both of its atoms are, the eraser and **Clear** take it away with the atom it belonged to, and paste carries it into the copy. **Remove highlight** takes it off the selection without touching the atoms.

6.1.7 Clearing the canvas

The bin button, in the same action grid as undo and tidy, wipes the whole drawing in **one undo step**. No confirmation prompt: the editing model here is snapshot undo throughout, and Ctrl+Z brings the drawing back.

6.1.8 Keyboard

Scoped to the window, so the main window's viewport modes are untouched, and all remappable under **Molecular Design** in **Edit** ⟩ **Preferences** ⟩ **Hotkeys**: V selection, 1/2/3 single/-double/triple bond, 4/5 wedge/hash, C chain, L atom label, X caption, P charge, E eraser, B ring, Y tidy, Ctrl+Return send to the viewport.

6.2 Supercells

Edit ⟩ **Unit Cell** ⟩ **Create Supercell** repeats the active structure along its lattice vectors. Three spin boxes, **Repeat along a/b/c**, each 1–20, default $2 \times 1 \times 1$. The operation is undoable and requires a defined cell along every repeated direction.

6.3 The surface slab wizard

Build ⟩ **Surface Slab** cleaves a slab from the active bulk structure in three guided stages. Unlike a plain Miller-index dialog, each stage shows you the consequence of your choice before you commit.

6.3.1 Stage 1 — surface orientation

Set the Miller indices $(h\,k\,l)$ with three spin boxes (-9 to 9), or work the other way round: drag the red **u** and green **v** handles on the interactive lattice canvas to any two lattice points, and Calango computes the nearest integer $(h\,k\,l)$ for the plane they span. The header shows the current vectors and resulting indices live, and the info line reports $|u|$, $|v|$ and the angle between them.

Rotate the axonometric view by dragging; only non-collinear handle pairs are accepted. **Next** unlocks once a test cut succeeds — $(0\,0\,0)$ is rejected as not a plane.

6.3.2 Stage 2 — cut, thickness and terminations

Calango builds a reference stack of eight bulk repeats and clusters the atoms into atomic layers (0.10 Å tolerance). The cross-section canvas draws each layer as a dashed line labelled with its z coordinate, with the current selection band highlighted between an orange *top termination* and a teal *bottom termination*.

Click a layer Moves whichever boundary is nearer the click, so you can set either termination directly.

Layers in slab Sets the count numerically.

Thickness The same choice in Å; it snaps to whole atomic layers above the bottom termination.

The info line reads back the selection, for example *layers 1–8 of 16 (8 layers, 12.45 Å)*.

6.3.3 Stage 3 — vacuum and preview

Top vacuum and **Bottom vacuum** (0–80 Å, default 10 each) pad the cell along the surface normal. **Symmetric vacuum (centered slab)**, checked by default, mirrors the top value to the bottom. A live 3D preview shows the finished slab, with atom count, slab thickness and total cell height. **Finish** inserts it as a new tab labelled, for example, *(1 1 1) slab, 8 layers*.

6.4 Liquid and gas interfaces

Build〉**Liquid / Gas Interface** opens a fluid region on the structure in the current tab and packs it with a liquid, a gas, a mixture, or an ionic solution — the solid/liquid and solid/gas cells that electrochemistry, catalysis and adsorption studies start from. It consumes a structure rather than generating one, so build or open the substrate first; a surface slab from §6.3 is the usual input. The result opens as a new tab and the substrate's own tab is left untouched.

6.4.1 Stage 1 — geometry and region

Interface direction Which lattice vector the region is opened along; c for a slab built by this application. The region is bounded by planes normal to the *other* two lattice vectors, so it stays commensurate with the cell even when that cell is tilted.

Region thickness The gap between the top face of the structure and the bottom face of its own periodic image. The cell is grown — or shrunk — to make this *exact*, whatever vacuum the input already carried. Asking for 20 Å of water and silently getting 20 Å plus the slab's existing 15 Å is the usual way to end up with an accidentally dilute interface.

Lateral supercell Repeats the substrate in the two directions that lie *in* the interface plane, before the region is opened. A wider cell is usually what a solvation study needs: the fluid's correlation length is several Å, and a 1×1 surface cell forces it to be periodic on a shorter length than that.

Surface clearance Closest approach between any fluid atom and any substrate atom (default 2.2 Å).

Move the substrate to the bottom of the cell On by default, so the fluid region is one contiguous block. Turn it off when the slab was positioned deliberately — a symmetric slab centred in its cell, for instance.

The summary underneath reports the cell before and after, the fillable volume, and how many water molecules that volume holds at ambient density.

6.4.2 Stage 2 — solvation and mixture

Liquid / gas mixture is a table of species and mole fractions. The fractions are normalised, so {3 water, 1 ammonia} and {0.75, 0.25} request the same mixture. **Amount** chooses between a target mass density and an explicit molecule count.

Ionic species inserts salts or bare ions by *formula unit*. A salt expands into its ions, so three units of $(NH_4)_2SO_4$ is six ammonium and three sulfate and the cell stays neutral by construction. Bare ions are offered too, for the cases where an unbalanced charge is deliberate; the net charge is reported below the table, in orange when it is not zero.

Ions are placed *before* the solvent and their mass counts against the density target. Both matter: a packing that saturated could otherwise drop an ion and silently change the stoichiometry, and a density target that ignored the ions would give a brine at the density of water with salt added on top of it.

> **Tip**
> A gas at its 1 bar density comes to a fraction of a molecule in a cell of interface size, so picking a gas switches the page to an explicit molecule count. That is the physically meaningful control: in a periodic cell the molecule count *is* the partial pressure.

6.4.3 What the packer does, and what it does not

Molecules are proposed at random positions with uniformly random orientations and kept when they clear everything already placed. Two fluid molecules are excluded on their centres, using a tabulated effective radius per species, with an atom-level floor on top so that two molecules facing each other cannot interlock their hydrogens; the substrate is checked atom by atom, because that surface is fixed and a molecule started inside it cannot relax out. The packing is seeded and therefore reproducible.

> **Requires**
> The result is a legitimate *starting point* — right composition, right density, no overlaps — and it is not an equilibrated liquid. A random packing has no hydrogen-bond network and a potential energy far above the equilibrium one. Quoting a diffusion coefficient, an interface energy or a work of adhesion computed directly on this cell would be reporting a property of the random number generator. Equilibrate with molecular dynamics (§9.1.3) first.

Random sequential packing saturates below the equilibrium density of a hydrogen-bonded liquid; around 90% of the target is typical for water and is perfectly usable. A larger shortfall is reported explicitly after the cell opens, together with what to do about it.

> **Note**
> The species library holds only molecules whose geometry is fixed by symmetry plus a bond length and, where needed, one angle: water, ammonia, HF, H_2S, the common diatomic and small polyatomic gases, the noble gases, the alkali and alkaline-earth cations, the halides, and the molecular ions NH_4^+, H_3O^+, OH^-, NO_3^-, CO_3^{2-}, SO_4^{2-} and PO_4^{3-}. Species that need a real optimised geometry — the alcohols, anything with a torsional degree of freedom — are deliberately absent rather than approximated: a molecule with plausible-looking but wrong internal coordinates produces a cell that nobody notices is wrong until the first

relaxation step tears it apart.

6.5 Dislocations

Build ⟩ Dislocation inserts a Volterra line defect into the structure in the current tab by displacing its atoms with a closed-form elastic field. The result opens as a new tab; the parent is left alone.

A dislocation is one operation: cut the crystal along a half-plane bounded by a line, slide the two faces past each other by a lattice vector **b**, and weld them back together. Everything the module does is that operation, evaluated analytically at every atom.

6.5.1 Stage 1 — type and geometry

Line direction names the Cartesian axis the dislocation line runs along; the other two axes follow in cyclic order as $(\mathbf{e}_1, \mathbf{e}_2)$. For a line along z those are (x, y), and an edge dislocation then has **b** along x with its glide plane normal along y — so naming the line names everything.

Burgers vector is $|\mathbf{b}|$ in Å. Use a lattice repeat of the host: a perfect dislocation's Burgers vector *is* a lattice vector, and any other length leaves a stacking fault trailing behind the line that this module does not model. **Line position** places the line in the plane normal to it, as a fraction of the structure's extent. The default is deliberately a little off centre — linear elasticity is singular *on* the line, so a line placed exactly on a lattice site evaluates that atom's displacement at infinity.

Edge Single edge dislocation. $\mathbf{b} \perp$ line, the crystal carries an extra half-plane ending on it, and the atom count is unchanged — this is the elastic field, not a plane of atoms inserted by hand.

Screw Single screw dislocation. $\mathbf{b} \parallel$ line; the lattice planes around it form one continuous helical ramp, and only the component along the line moves.

Glide Two edge dislocations of opposite sign in the *same* glide plane — what a dislocation leaves behind after gliding a distance d. Between the cores the crystal above the glide plane has slipped by exactly one **b** relative to the crystal below. Conservative: no atom is created or destroyed.

Climb Two edge dislocations of opposite sign stacked normal to the glide plane, with a platelet of *missing* material between them — a collapsed vacancy disc one **b** thick. Climb is mass transport, so this is the one construction that changes the atom count, and that change is how you tell it happened.

Anisotropic A single dislocation solved from the full elastic tensor via Stroh's sextic formalism. The only option that can describe a *mixed* dislocation — one whose Burgers vector has both an edge and a screw component — and the only one that is correct for a strongly anisotropic crystal.

6.5.2 Stage 2 — elasticity

The isotropic types read a single **Poisson ratio**; a screw dislocation does not depend on it at all. The two dipoles read a **Core separation** — the distance the dislocation travelled, for glide, and the height of the vacancy platelet, for climb.

The anisotropic type reads an elastic tensor instead, built from the constants of a chosen **Elastic symmetry** (cubic, hexagonal or isotropic), with room-temperature single-crystal presets for Cu, Al, α-Fe, Ni, W and Si. Only the *ratios* of the constants reach the displacement field — the

absolute scale cancels — but keeping physical units means numbers can be read straight off a table. The **Burgers direction** gives **b**'s components in the dislocation frame: $(1, 0, 0)$ is a pure edge, $(0, 0, 1)$ a pure screw, anything between a mixed dislocation.

The page runs the real builder on every change and reports what it produced — atom count, largest displacement, closest resulting pair, and any warnings. It is not a model of what **Finish** will do; it is what **Finish** does.

6.5.3 Periodicity

A single dislocation has a net Burgers vector, so its displacement field is multivalued: it cannot be made periodic in the two directions normal to the line, whatever the cell. Edge, Screw and Anisotropic therefore leave the cell periodic *only along the line* and say so — which is the correct setup for a cylinder with free lateral surfaces, the usual way core structures are computed.

A dipole has zero net Burgers vector, so Glide and Climb stay periodic in all three directions. Reaching that takes one extra step: superposing two opposite fields cancels the net Burgers vector but leaves the average plastic distortion the pair carries, so a compensating uniform distortion — a simple shear for glide, a uniaxial contraction for climb — is applied to the atoms *and* to the cell vectors. Without it the construction looks perfectly normal in a viewer and carries a $|\mathbf{b}|$-sized discontinuity at the periodic boundary.

> **Requires**
> Nothing here is relaxed, and linear elasticity is singular at the line. The handful of atoms nearest the core arrive at positions that are qualitatively right and numerically meaningless. Relax the core before quoting anything about it — a core energy, a Peierls barrier, a dissociation width.

> **Note**
> Only the vacancy sense of climb is built (material removed). The interstitial sense would mean inserting a partial plane of atoms, which needs the periodicity of the host lattice and not merely its Burgers vector; it is refused rather than approximated. The module also reports when the requested $|\mathbf{b}|$ is smaller than the plane spacing it was meant to remove, in which case the result is an elastic dipole with no mass transport at all.

6.6 Solid interfaces

Build ⟩ Solid Interface builds the constructions that put two pieces of crystal next to each other. All five are the same operation seen from different distances: decide, for every point of space, which crystal orientation occupies it; fill each region with that lattice; reconcile the seams.

Stacking fault Everything above the chosen plane is shifted rigidly by a fraction of an in-plane lattice vector. The cheapest planar defect there is, and the one whose energy decides how readily a dislocation splits into partials.

Twin boundary The half above the plane is discarded and replaced by the mirror image of the half below it. A coherent twin: every atom on the boundary is shared between the two orientations, so the interface carries no free volume. Anything that existed only in the replaced half is gone — that is what twinning does.

Bicrystal Two grains with independent orientations filling the two halves of a box built from

the parent cell. The general grain boundary.

Polycrystal A Voronoi tessellation: N seeds scattered in the box, each with a uniformly random orientation, every point of space joining the grain whose seed is nearest.

Multi-phase polycrystal The same tessellation, with each grain drawing its lattice from a different phase. Every grain is single-phase; the interfaces between them are phase boundaries.

6.6.1 Stage 1 — interface and plane

Boundary normal names a *lattice* vector rather than a Cartesian axis, which is what keeps the boundary plane — spanned by the other two lattice vectors — commensurate with the cell by construction. **Boundary position** places it as a cell fraction.

The stacking fault reads a **Fault vector**, given as fractions of the two in-plane lattice vectors; $\frac{1}{3}$ along one of them is the classic partial-dislocation fault vector of a close-packed plane. Both planar defects can open a **Boundary gap** perpendicular to the interface, which gives a relaxation somewhere to start from instead of two planes on top of each other.

The space-filling kinds read a **Merge tolerance** instead: atoms landing closer than this to one already placed are deleted. Two crystals meeting at an arbitrary angle always overlap somewhere, and without this the boundary is a pile-up rather than a boundary. Too large and it starts eating the grains themselves — watch the reported atom count.

6.6.2 Stage 2 — grains

Box repeats sets the size of the constructed cell in multiples of the parent. A bicrystal reads two rotations about the boundary normal, so the pair forms a twist boundary and the misorientation is their difference; 36.87° is the Σ5 coincidence-site relationship of a cubic lattice. A polycrystal reads a **Grains** count and a **Random seed**; the same seed rebuilds the same cell exactly, which is what makes a published polycrystal reproducible. A multi-phase polycrystal adds a table of the open workspace tabs with a weight each — a weight of zero excludes that phase entirely.

Every atom of a space-filling construction carries its grain index and phase index as per-atom scalar fields, so the tessellation can be *seen* in the viewport (Representation ⟩ colour by scalar field) rather than assumed.

Requires

A periodic cell cannot contain an odd number of parallel interfaces. Insert one boundary at the middle of a cell and you have inserted two: the one you asked for, and the one where the top face meets the bottom face of the next image. That is not a defect of the implementation, it is what periodic boundary conditions mean, and every result reports both. A calculation that attributes the whole excess energy of the cell to "the" boundary is out by a factor of two.

Note

Rotating a lattice and demanding it still fit the box is a strong condition: only the coincidence-site (CSL) misorientations satisfy it exactly. At any other angle the two crystals meet the periodic boundary out of register. The mismatch is measured and reported in Å rather than hidden, so a bicrystal built at an arbitrary angle announces what it is.
Twinning additionally requires that the stacking vector be perpendicular to the boundary plane. Reflecting a tilted cell would produce a half-crystal that is periodic with a *different* cell, so the construction is refused with a message asking for an orthogonalised cell rather

than returning a box that describes neither half.

Requires
Nothing here is relaxed. Grain boundaries have structure — free volume, reconstruction, segregation — that no geometric construction produces. These are starting points.

6.7 Nanomaterials

Build ⟩ Nanomaterial Builder generates four families of low-dimensional carbon and TMD structures. Pick the type at the top; the form below changes to match.

Graphene sheet Lattice constant a (default 2.46 Å), repeats $n_x \times n_y$ (default 4×4), vacuum (default 10 Å).

Graphene nanoribbon Width and length in unit cells (default 6 each), **Edge type Zigzag** or **Armchair**, and **Hydrogen-terminate edges** (on by default).

Carbon nanotube Chiral indices n and m (default 6, 6), length in unit cells, C–C bond length (default 1.42 Å), vacuum (default 8 Å). $(0, 0)$ is rejected.

TMD monolayer (MX_2) An editable formula combo — `MoS2`, `MoSe2`, `WS2`, `WSe2`, `MoTe2`, `NbSe2`, or anything else you type — **Phase** `2H` or `1T`, lattice constant (default 3.16 Å), X–X thickness (default 3.19 Å), repeats and vacuum.

6.8 Metallic nanoparticles

Build ⟩ Metallic Nanoparticle (Wulff) builds clusters two ways. Choose the element from the periodic table, then a mode.

6.8.1 Wulff construction

The equilibrium shape of a crystal is the one minimising total surface energy at fixed volume; the Wulff construction produces it from the relative surface energies of the exposed facets.

Facet table One row per facet: h, k, l and γ (relative surface energy). Defaults are $(111) = 1.0$, $(100) = 1.1$, $(110) = 1.2$. **Add Facet** and **Remove Selected** edit the list.

Target atoms The requested size, default 200. The construction lands on the nearest achievable closed shape.

Size rounding `closest`, `above` or `below`.

Lattice `fcc`, `bcc` or `sc`.

Only the *ratios* of the γ values matter. Lowering $\gamma(111)$ relative to $\gamma(100)$ grows the (111) facets and drives an fcc metal from a cube towards a truncated octahedron.

6.8.2 Spherical cluster

Carves every atom within a cutoff radius of the centre out of a bulk lattice — `fcc`, `bcc`, `sc` or `hcp`. Set **Radius** (default 10 Å). Useful when you want a specific diameter rather than a thermodynamic shape.

Both modes centre the cluster with 6 Å of vacuum and clear periodicity. **Lattice constant** may be left at **ASE default** to use the tabulated value for the element.

6.9 Special quasirandom structures

Build ⟩ Special Quasirandom Structure (SQS) builds a substitutional alloy whose short-range order approximates a truly random solid solution — the standard way to model a disordered alloy in a small periodic cell.

Supercell Repetitions $n_1 \times n_2 \times n_3$, default $2 \times 2 \times 2$.

Replace element Whose sites form the alloy sublattice.

Composition Target occupancies as symbol:fraction pairs, e.g. `Cu:0.75, Au:0.25`. Fractions are normalised and rounded to whole atoms by largest remainder.

First shell cutoff, Second shell cutoff The pair shells whose Warren-Cowley parameters are driven toward zero. Defaults 3.2 and 4.8 Å; set the second to **off** to use one shell.

MC steps, Random seed Anneal length and reproducibility.

If `icet` is importable, Calango uses its `generate_sqs`; otherwise it falls back to a built-in simulated annealer that minimises $\sum \alpha^2$ over the chosen shells using only NumPy and ASE neighbour lists. The resulting tab is labelled with the backend that produced it, and the internal backend also reports its residual objective — a value near zero means the decoration is essentially ideal.

> **Tip**
> Verify the result with **Analysis ⟩ Warren-Cowley analysis** (Section 15.6): a good SQS shows $\alpha \approx 0$ for the shells you optimised.

6.10 Normal modes and phonons

Build ⟩ Normal Modes / Phonon Builder sets up a finite-displacement vibrational calculation. Calango detects whether the structure is periodic and adapts:

Periodic Supercell finite displacements via `ase.phonons`, giving dispersion and DOS.

Isolated molecule Γ-point normal modes via `ase.vibrations`, giving frequencies and per-mode animation trajectories that Calango opens directly.

6.10.1 Parameters

Mode **Run vibrational calculation**, or **Generate displaced structures only (new tab)** to export the displacement set for an external DFT code.

Supercell (a × b × c) Default $2 \times 2 \times 2$, periodic structures only.

Displacement δ Default 0.01 Å.

Displaced structures A read-only count: $6N + 1$ for N supercell atoms.

Calculator EMT, Lennard-Jones or MACE — deliberately limited to calculators needing no external binary or pseudopotentials.

Band-path points, DOS k-grid Sampling for the dispersion and the phonon density of states (defaults 100 and 20).

Like the calculator dialog, the generated ASE script is shown in an editable pane and can be saved with **Save Script...**.

Outputs are Γ-point frequencies in the job console, plus `phonon_bands.csv` and `phonon_dos.csv` in the job directory.

> **Note**
> Displacements are applied to every atom of the expanded supercell ($6N_{\text{supercell}} + 1$ configurations), following the standard finite-displacement recipe without symmetry reduction. For large supercells this is the dominant cost; symmetry-reduced displacement sets are a planned improvement.

6.11 Random Noise Setup

Simulation ⟩ Random Noise Setup perturbs the structure in the current tab into a trajectory — for shaking a structure out of a symmetric saddle point, or for generating training ensembles for machine-learning potentials. It is a pure generator: native throughout, with no script and no job, since the perturbation is evaluated in process and the result opens immediately as a scrubbable trajectory tab.

> **Note**
> Evaluating the ensemble (energies, forces, ...) is not this dialog's job any more. Save the generated trajectory (**File ⟩ Save Trajectory As...** on the tab it opens) and load it into a **Structure Container** node on the Orchestration canvas, which fans a Single-point Calculation node out once per structure — the same batch machinery every other multi-structure Orchestration pipeline uses. An earlier version of this dialog ran a single-point pass on every member itself; nothing about that configuration was ever saved to a project file, so there is no old setup to migrate.

Distribution **Gaussian (normal)** or **Uniform**.

Amplitude Default 0.05 Å. For Gaussian this is σ per component; for uniform it is the half-width. With the ramp on (below), this is the value the LAST frame reaches.

Random seed Default 42 — the same seed regenerates the same trajectory exactly, ramp on or off.

Perturb atomic positions On by default.

Perturb unit cell vectors (random strain) Atoms follow the cell affinely, preserving fractional coordinates.

Structures Default 20 (1–1000): how many displaced copies to generate, on top of the always-included reference frame.

Accumulation **Independent** draws each frame from the original structure; **Cumulative** performs a random walk from the previous frame.

Amplitude schedule **Constant amplitude**, or **Linear ramp (0 → max)**.

Frame 0 is always the unperturbed structure. With the linear ramp on, frame i of N total frames (reference plus displaced members) uses $i/(N-1)$ of the set amplitude — frame 0 is already the ramp's zero-noise endpoint by construction, with no special case needed, and the last frame reaches the full amplitude. Each frame still draws its own independent random displacement; only the magnitude is interpolated, so the same seed still reproduces the same trajectory exactly. A ramped trajectory is what a training set for an interatomic potential usually wants: near-equilibrium configurations shading into the more anharmonic displacements a model needs to see to generalize, from one generated set.

The generated trajectory opens immediately as a scrubbable tab, so the displacement magnitude can be judged by eye — and regenerated with a different seed or amplitude — before it is saved anywhere.

6.12 2D Ripples

Modules ⟩ **2D Materials** ⟩ **2D Ripples** imposes a sinusoidal out-of-plane corrugation on a monolayer supercell, and contracts the cell in-plane so the sheet's own length is unchanged. It is the one entry on that menu that *produces* a structure — the four below it read one — and, like Random Noise Setup, it is a pure in-process generator with no script and no job.

A monolayer built by any of the 2D builders comes out perfectly flat, and a real one is not: suspended graphene and every other 2D crystal carries static ripples, and their amplitude is what couples to the electronic structure, the bending rigidity and the effective membrane area.

6.12.1 The profile

The displacement is along the out-of-plane normal, sinusoidal in the in-plane *fractional* coordinates:

$$h(f_1, f_2) = A \sin(2\pi n_1 f_1) \sin(2\pi n_2 f_2), \qquad h(f_1) = A \sin(2\pi n_1 f_1).$$

Fractional, not Cartesian, and that is what makes it work on the cells these materials actually have. Written as $\sin(2\pi x/L_x)$ the profile is periodic only in an orthogonal cell; written in fractional coordinates with an *integer* number of periods it is periodic by construction for any cell shape — which matters immediately, because graphene, h-BN and the TMDs all have in-plane vectors at 120°. A supercell of graphene's primitive cell is the ordinary case here, not the awkward one.

Out-of-plane axis Seeded from the geometry and left editable. Getting it wrong does not produce a slightly wrong sheet — it corrugates the structure sideways.

Direction Both in-plane directions (an egg-box corrugation), or one only (straight crests running across the sheet).

Amplitude A Peak height, so the sheet spans $2A$ between its lowest and highest points.

Periods per cell A whole number, per in-plane axis. The control has no other setting: fractional 0 and 1 are the same periodic image, so a fractional period count is a seam in the sheet that no amount of relaxation repairs.

Contract the cell to preserve arc length On by default.

6.12.2 The in-plane contraction

Rippling a sheet does not stretch it: the extra path the ripple takes comes out of the *footprint*. Leaving the cell at its flat length while displacing the atoms applies a tensile strain equal to the whole arc-length excess, silently, and growing as A^2. So each rippled cell vector is contracted to the L satisfying

$$\int_0^L \sqrt{1 + (\mathrm{d}h/\mathrm{d}x)^2}\,\mathrm{d}x = L_0,$$

solved numerically. The leading behaviour $L_0 - L \approx \pi^2 n^2 A^2 / L$ is an anchor for the numerics, not a substitute: over a 25 Å cell it is out by 0.11 % of the cell length at $A = 2$ Å and 0.69 % at $A = 3$ Å — that much accidental tensile strain, for free.

Only the direction(s) the profile varies along are contracted, the lattice angle survives exactly, and the vacuum axis is untouched. Measured on a 6×6 graphene supercell at $A = 0.6$ Å: the mean C–C bond goes from 1.420282 Å flat to 1.423687 Å (+0.24 %) with the contraction, against 1.435091 Å (+1.04 %) without it.

6.12.3 The amplitude series

Amplitude series generates a whole ramp instead of a single structure: N frames from **A minimum** to **A maximum**, opened as one scrubbable trajectory. This is the point of the module — a single rippled sheet answers nothing; $E(A)$ does. Save it and load it into an Orchestration **Structure Container** node to fan a calculation out over the series.

Every frame is a full build at its own amplitude, contraction included: the contraction is quadratic in A, so an interpolated series would be right only at its two ends. Each frame carries its own amplitude as the per-atom scalar field `ripple_amplitude`, which survives an extended-XYZ round trip.

An amplitude the cell cannot hold is refused rather than clamped — a sinusoid of amplitude A with n periods is at least $4nA$ long whatever its footprint — and the wizard says so before **Generate** is pressed.

CHAPTER 7

Editing Structures

Every operation in this chapter is a single undo step. **Edit** ⟩ **Undo** (Ctrl+Z) and **Edit** ⟩ **Redo** (Ctrl+Shift+Z) walk a per-document history up to 50 states deep.

7.1 Atoms

Edit ⟩ **Add Atom** Ctrl+Shift+A. Choose the element from the graphical periodic table and type exact x, y, z coordinates. For placing atoms by eye instead, use Insertion mode in the viewport (I).

Edit ⟩ **Selection** ⟩ **Change Element of Selection** Replaces the species of every selected atom, again via the periodic table.

Edit ⟩ **Selection** ⟩ **Translate Selection** Shifts the selection by a Δx, Δy, Δz vector in Å.

Edit ⟩ **Delete Selected Atoms** Delete. Removes the selection; with the viewport focused in Selection mode, Backspace does the same.

Deleting atoms re-indexes everything that referenced them — bond overrides, bond orders and per-atom fields all follow correctly.

7.2 The Edit Structure dialog

Edit Structure... in the **Structure** panel opens a two-section editor over a working copy: the unit cell above, and **Atomic Configurations** below.

7.2.1 Coordinates

The table shows *one* coordinate triple, and the **Fractional coordinates** checkbox decides which. Unchecked — the default — gives Cartesian x, y, z in Å; checked converts the values in place and relabels the columns u, v, w. Both are editable: a fractional edit recombines all three components of that row and converts back, so changing u alone does not discard v and w.

The checkbox is disabled without a unit cell. Fractional coordinates there are not merely unavailable, they are undefined.

> **Note**
> One triple rather than two side by side is what makes room for the extended per-atom properties below. It also removes a real hazard of the old layout: with $x \ldots w$ all editable

> at once it was possible to type into a Cartesian cell and a fractional cell of the same row before either was committed, and only one of the two edits survived.

7.2.2 Extended per-atom properties

Any per-atom array the structure carries gets its own column, to the right of the magnetic moments: partial charges (`initial_charges`), velocities, forces — anything an extended-XYZ file put in `atoms.arrays`. A scalar array gives one column; a vector array gives three, suffixed x, y, z. Columns appear and disappear with the data, so a plain `.xyz` opens with none and an MD frame opens with several.

Most of these columns are **read-only**, and deliberately: they are results. A charge came out of a Bader partitioning; typing over it in a geometry editor would leave a frame whose arrays no longer correspond to anything that was computed, and the file would carry no sign of it.

Velocities and forces are the two exceptions, editable like the coordinates, because both are legitimately an *input* as well as a result: hand-set velocities are the initial conditions of a molecular-dynamics run, and hand-set forces are a way to assemble force-field training frames or to exercise a force-arrow viewer. Edits write into the structure's own arrays, so a save to extended XYZ carries them (velocities through ASE's momenta), and a derived |forces| magnitude column follows the edit rather than going stale. The magnetic moments likewise remain editable, through the spin-mode columns.

7.2.3 Sorting

Sort by reorders the atoms by element or by any coordinate, ascending or **descending**.

This *renumbers* the atoms — it permutes the structure itself, not just the view. That is normally the point: grouping by element is what VASP's POSCAR/POTCAR pairing wants, and sorting by z gives a slab a layer-ordered index list you can select ranges from. Everything index-aligned travels with its atom: per-atom scalar and vector fields, magnetic moments, residue annotation, bond overrides and manual bond orders.

The sort is stable, so atoms that tie on the key — every atom of one element, every site in one layer — keep their existing relative order rather than being reshuffled on each press.

7.2.4 Spin polarization

The **Spin polarization** dropdown adds editable magnetic-moment columns to the table:

Unpolarized No columns. This means *no moments*, not moments of zero — an all-zero column would still switch a calculator into spin-polarized mode and cost the run its speed for nothing.

Collinear Spin-Polarized One signed **m (μ_B)** column. The sign is what makes a seed antiferromagnetic rather than ferromagnetic, which is why it is set per atom rather than as one number.

Non-collinear Spin Three columns, **mx**, **my**, **mz**.

What you type is stored as the structure's *initial* magnetic moments. It travels with the geometry into every calculation — the wizards no longer ask for a list of moments, they read these — and it can be drawn in the viewport through **Spatial References** ⟩ **Vectors** ⟩ **Vector overlay** → **Initial magnetic moments**.

> **Note**
> That overlay is deliberately separate from **Magnetic moment**, which shows the moments a calculation *produced*. A guess and a result are different claims, and drawing one as the other is how a seeded antiferromagnet gets reported as a converged one. The two even use different arrow colours.

The dialog opens on whatever mode the structure is already in, so a magnetic structure shows its moments instead of hiding them behind a dropdown you would have to know to change. The summary line reports the net moment alongside $\Sigma|m|$: a column of 2.2 and a column of ± 2.2 look nearly identical and mean opposite things.

7.3 The periodic table selector

Wherever an element must be chosen, Calango opens a graphical periodic table: $Z = 1$ to 118 in the standard 18-group layout, f-block below, coloured by chemical family. One click selects and closes.

7.4 Bond perception

By default Calango perceives bonds by distance: atoms i and j are bonded when

$$d_{ij} < t \cdot \left(r_{\mathrm{cov}}(i) + r_{\mathrm{cov}}(j)\right),$$

with covalent radii from Cordero *et al.* and a tolerance t you control. With a periodic cell the minimum-image convention applies, so bonds across cell boundaries are found and drawn correctly.

Edit ⟩ **Bond Editor** (Ctrl+B) exposes both the automatic rule and manual overrides:

Automatic bond detection Turn it off to draw *only* manually added bonds — the right choice for coarse-grained or schematic figures.

Covalent cutoff multiplier The tolerance t, 0.50–2.50, default 1.15. Applied live, so you can dial it until the picture is right.

Manual Bonds Two 1-based atom-index spin boxes, or **From Selection** to fill them from a two-atom viewport selection. The info line shows the current pair's distance next to the automatic cutoff for that pair, which makes it obvious whether a bond is borderline.

Add Bond Forces a bond regardless of distance.

Suppress Bond Removes an automatically perceived bond.

Active overrides Lists every override as *added* or *suppressed*, with the pair and its distance. **Clear Override** and **Clear All** undo them.

Overrides live on the structure and are saved in the project file.

7.4.1 Hydrogen bonds

The **Hydrogen Bonds** tab perceives D–H$\cdots$A contacts *geometrically* — a maximum donor–acceptor separation (2.0–6.0 Å, default 3.5 Å) and a minimum D–H$\cdots$A angle (90–180°, default

120°) — and draws them as a broken stroke. These are not stored overrides: they re-derive from the geometry, so they follow a trajectory instead of freezing at the frame they were detected on. A live count distinguishes "criteria too strict" from "none present".

Color The stroke colour.

Line style **Solid**, **Dashed** (the default, and what this overlay has always drawn) or **Dotted**. A broken stroke is the convention for an interaction that is *not* a covalent bond, and it says so beside the solid cylinders it sits among without needing a caption. Solid is for a figure in which the hydrogen bonds are the subject; dotted keeps the dash spacing and shortens the marks, for a dense structure where dashes compete with the geometry they cross.

Line width On the same scale as the unit cell's own line width, so the same number is the same thickness on both.

The stroke is real geometry, not an OpenGL line: core-profile OpenGL clamps line width (to 1 px outright under macOS's profile), so a width control over a line primitive would do nothing at all. The marks are built as thin crossed quads, exactly as the unit cell's edges are built as thin tubes and for the same reason. The one consequence is that the stroke is in *world* units, so it scales with the camera like a bond rather than staying a fixed number of pixels.

The live viewport draws hydrogen bonds, and so does the off-screen framebuffer capture behind image export, film frames and the animated GIF/MP4. The *ray-traced* export (POV-Ray, Tachyon) does not: its scene is built from the structure and the render style, and hydrogen bonds are a separate overlay that has never been part of it.

7.4.2 Rules apply to the whole trajectory

Every rule entered in the Bond Editor is applied to *every frame* of the active trajectory, not only to the frame on screen. Bonding is a statement about the chemistry of a system, and a system does not change its chemistry between two samples of the same run — an override that existed on frame 0 and vanished on frame 1 was never a physical statement, only an artifact of which frame happened to be displayed when the button was pressed. It also made every trajectory-wide export (the extended-XYZ writer, the ray-traced film, the animated GIF) disagree with the viewport.

The two kinds of rule propagate *differently*, and the difference is the point:

By Atomic Indices Names atoms. An atom keeps its index for the whole run, so the same pair — and the same bond order — is marked on every frame.

By Chemical Elements Names a *geometric condition*: every Si–O pair between 1.4 and 1.9 Å. The rule is therefore re-evaluated against each frame's own coordinates. Copying the first frame's match list forward would freeze a bond onto a pair that has since dissociated — precisely the event a reactive trajectory is being watched for, rendered invisible.

The scope is stated on the panel where the rules are entered, and the element-rule match count says which frame it was counted on. Frames whose atom count differs from the displayed one are skipped and reported: an index there refers to a different atom.

> **Note**
> Hydrogen bonds are the exception, and deliberately so. They are perceived geometrically rather than stored as overrides, so they re-evaluate whenever the geometry moves — they already follow a trajectory instead of freezing at the frame they were detected on.

7.5 Bond orders

Calango never guesses multiple bonds. Every perceived bond starts as a single bond, and orders are assigned deliberately — distance-based perception of double and triple bonds is unreliable, particularly for inorganic and metallic systems where short contacts are common.

To assign one, select exactly two atoms in the viewport (Ctrl+click the second) and use the **Bond order** buttons in the **Representation** panel: **Single**, **Double**, **Triple**. The buttons are disabled until a pair is selected.

An order of two or three renders as that many parallel cylinders, and also forces the bond to exist even if the pair lies outside the automatic cutoff. Orders persist in project files.

> **Note**
> A consequence worth knowing: molecules like benzene or CO_2 render with all-single bonds when first opened. Assign the orders you want to display; they are a presentation property, not an input to any calculation.

CHAPTER 8

Appearance and Representation

8.1 The Representation panel

8.1.1 Representation mode

Mode selects how atoms and bonds are drawn:

Ball-and-Stick Spheres at a fraction of the covalent radius, bonds as cylinders. The default.

Space-filling (CPK) Spheres at approximately the van der Waals radius; bonds are hidden inside the atoms.

Wireframe Bonds only, drawn as lines with points at the atom positions — the cheapest mode for very large systems.

Atom radius and **Bond width** scale everything globally, 0.20–3.00, each with a slider and a synchronised spin box for exact values.

8.1.2 Colouring atoms

Color by offers four mappings, applied consistently to atoms and to their halves of each bond:

Element (CPK) The Jmol palette, with per-element overrides.

Coordination number (CN) Discrete coordination numbers on a gradient.

Generalized CN (GCN) Continuous generalised coordination numbers — excellent for distinguishing terraces, steps, edges and vertices on nanoparticles and slabs.

Custom property Any per-atom scalar field the structure carries: charges, force magnitudes, extended-XYZ columns, or a computed field such as `local_entropy`. **Property** selects which.

For the three scalar modes, **Gradient** chooses the colour map — **Viridis**, **Plasma**, **Turbo**, **Inferno**, **Magma**, **Cividis**, **Hot**, **Afmhot**, **Coolwarm**, **Rainbow** — and **Invert palette** reverses the scalar-to-colour mapping, matching matplotlib's `_r` variants. **Range** reads back the min and max of the active field, so the legend is never ambiguous.

> **Tip**
> Viridis and Cividis are perceptually uniform and colour-vision safe; Coolwarm is the right choice for signed quantities such as charges or potentials, where the zero point should read as neutral. Rainbow is included because reviewers sometimes ask for it, not because it is a good default.

Colour mapping is recomputed for every frame during trajectory playback, so CN, GCN and property maps stay live throughout an animation.

8.1.3 Bonds and vectors

Gradient bond coloring Blends each bond smoothly from one atom's colour to the other's, instead of the classic half-and-half split.

Force vectors, Velocity vectors Draw per-atom vectors as lit 3D arrows. Each is enabled only when the structure actually carries that field; the tooltip explains what is missing when it does not.

Vector scale Arrow length in Å per field unit, 0.05–200, default 10.

8.1.4 Per-element styling

Element Settings... opens a table of the elements present in the structure. Each row offers a colour swatch and a radius scale (10–300 %, where exactly 100 % means "no override"), plus a **Reset** button. **Reset All** clears every override at once.

Background sets the viewport background colour. Distance fog, if enabled, automatically fades into whatever you choose.

8.2 Spatial References

The **Spatial References** dock (zone 12) holds four tabs — the cell wireframe, the orientation triad, and the per-atom vector overlay (covered with the rest of the default layout in Chapter 3), plus, last, the ground plane, since it draws a whole extra object into the scene rather than an overlay on the structure the way the other three do.

Show unit cell Also on **View** ⟩ **Overlays**.

Cell color Wireframe colour.

Cell line width 1.0–8.0, default 2.0. A width of 1 draws plain GL lines; anything larger renders the edges as lit tubes.

Show axes triad The corner orientation indicator.

Axes style **Cartesian (X, Y, Z)** or **Lattice vectors (a1, a2, a3)**.

Axes size On-screen size of the triad, 48–240 px.

> **Note**
> Core-profile OpenGL clamps hardware line width to one pixel, which is why thick cell edges are drawn as tube geometry rather than wide lines. The tubes are lit like the rest of the scene, so they also read correctly in ray-traced exports.

8.2.1 Floor

The **Floor** tab's **Ground plane (floor)** checkbox adds a large plane just under the structure, so an isolated molecule reads as an object resting in a space rather than one floating in a void. It

is a shadow *receiver* only — atoms and bonds cast onto it, from the **Shadow** tab of the Visual Effects dock (Section 4.7), and it casts nothing itself.

Plane **xy**, **xz** or **yz**; the default **xy** sits perpendicular to **c** (world $+z$), the axis Calango's default view has pointing up.

Offset Extra clearance beyond the automatic fit, which lands the plane 0.25 Å under the lowest drawn point and re-fits itself whenever the structure changes or a trajectory is scrubbed — the offset is what survives that re-fit.

Color, Opacity The plane's own shaded colour and translucency.

The floor is **display only**: never part of the structure, never picked by a click, never written into an exported POSCAR, CIF or XYZ. It *does* appear in every render of the scene — off-screen and publication renders, animations, POV-Ray and Tachyon scene files, and the Alembic cache, where it rides the same **Include unit cell** switch as the rest of the scene furniture (and, uniquely there, always bakes in fully opaque, since Alembic meshes carry no opacity channel).

8.3 Lighting

The **Light** tab of the **Visual Effects** panel (zone 9) edits up to four directional lights with Blinn-Phong shading. The default studio setup is three lights: a warm key carrying the speculars, a cooler fill from the opposite side, and a back (rim) light that separates atom silhouettes from the background.

Select a light in the list, then edit:

Direction (view space) Three components, −5 to 5. Because directions are in view space, the lighting stays fixed relative to the viewer as you orbit — the structure turns, the studio does not.

Ambient, Diffuse, Specular Per-light colours.

Add appends a fill light from the opposite side; **Remove** deletes the selected one. At least one light always remains.

CHAPTER 9

Running Simulations

9.1 The calculation dialog

Simulation ⟩ **New Calculation (ASE)** (Ctrl+R) is the main entry point. The left half is a form; the right half is the ASE script that form produces, live and editable.

9.1.1 Calculators

Calango is *calculator-agnostic*. The wizard, the script it generates and the job runner are written against ASE's calculator interface, not against any particular code, so one staged flow — calculator, settings, task, script review — drives empirical potentials, semi-empirical tight binding, plane-wave and all-electron DFT, quantum chemistry and machine-learning potentials alike. Adding an engine adds a settings group and nothing else: everything downstream of the **Calculation engine** dropdown is unchanged.

The table below is a representative selection of the engines in common use, not the complete library — that keeps growing, and the **Calculation engine** dropdown in the version you are running is its authoritative list.

Calculator	Notes
EMT	Effective medium theory. Fast, ships with ASE, useful for testing a workflow before committing compute.
Lennard-Jones	Ships with ASE.
Quantum ESPRESSO	DFT; needs `pw.x` and pseudopotentials.
VASP	DFT; needs a license and POTCARs.
MACE	Machine-learning potential; needs `mace-torch`.
GPAW	DFT as a Python package.
SIESTA	DFT; needs the `siesta` binary and pseudopotentials.
ORCA	Quantum chemistry; needs the ORCA binaries.

The DFT entries generate working script skeletons with clearly marked `EDIT ME` lines where you must supply pseudopotential paths or launch commands.

9.1.2 Tasks

Single-point energy One energy and force evaluation.

Geometry optimization (BFGS) Controlled by **Force convergence (fmax)** (default 0.05 eV/Å) and **Max optimization steps** (default 200). Writes `opt.traj`.

Molecular dynamics See below. Writes `md.traj`.

9.1.3 Molecular dynamics ensembles

Ensemble	Thermostat / barostat	Key parameters
NVE	Velocity Verlet	timestep
NVT	Langevin (default)	temperature, friction
NVT	Andersen	temperature, collision probability
NVT	Berendsen	temperature, τ_T
NVT	Nosé–Hoover chain	temperature, τ_T
NPT	Berendsen	temperature, pressure, τ_T, τ_P
NPT	Nosé–Hoover / Parrinello–Rahman	temperature, pressure, τ_T, τ_P

Shared parameters: **Temperature** (default 300 K), **Timestep** (default 1 fs), **MD steps** (default 1000), **Thermostat coupling** (default 100 fs), **Barostat coupling** (default 1000 fs) and **Pressure** (default 0 GPa). Fields enable themselves only for the ensembles that use them.

> **Note**
> Every MD run initialises velocities from a Maxwell-Boltzmann distribution and then removes the net centre-of-mass momentum, so the system does not drift as a whole. For isolated (non-periodic) systems the net angular momentum is removed as well. Without this, a slow overall translation contaminates every subsequent analysis — diffusion coefficients most of all.

9.1.4 Simulated annealing

The **Mode** selector at the top of the dynamics page switches the run between **Constant temperature** — ordinary MD at a fixed setpoint — and **Annealing**, which sweeps the thermostat target from an **Initial temperature** to a **Final temperature** over the course of the run. Everything else is unchanged: the same integrator, the same constraints, the same sampling, the same streamed trajectory. Only the target moves, and it is retargeted on *every* step rather than once per sampling interval.

Three schedules are offered, written in terms of the run fraction $x = n/N$ where n is the current step and N the total:

Schedule	$T(x)$
Linear	$T_0 + (T_1 - T_0)\,x$
Exponential	$T_1 + (T_0 - T_1)\dfrac{e^{-kx} - e^{-k}}{1 - e^{-k}}$
Logarithmic	$T_0 + (T_1 - T_0)\dfrac{\ln(1 + kx)}{\ln(1 + k)}$

All three are *endpoint-exact*: $T(0) = T_0$ and $T(1) = T_1$ whatever the coefficient k. That is deliberate — a schedule that only approaches its target asymptotically ends the run at a temperature nobody asked for, and the discrepancy is invisible in a plot that has been cooling for 50 ps. The coefficient k (**Ramp curvature**) is a curvature rather than a rate constant with units: it says how far the ramp bends away from a straight line, and every schedule degenerates to Linear as $k \to 0$. Linear has no free coefficient, so the field is disabled there.

Exponential moves most of the way early and crawls at the end, which is the shape wanted for quenching a melt. Logarithmic is the slowest of the three near the target; it is the practical, endpoint-exact relative of the Geman–Geman $\sim 1/\ln n$ schedule, the one with the global-optimum convergence proof and a run length nobody can afford.

The page restates the chosen schedule as the temperatures it actually produces, at 0/25/50/75/100 % of the run, together with the elapsed time the ramp spans. A named curve is not a number

anyone can check; "1000 → 631 → 419 → 315 → 300 K over 20 ps" is.

> **Note**
> Annealing needs a thermostat to retarget, so NVE is withdrawn from the ensemble list while the mode is selected (and the ensemble is moved to Langevin if NVE was the one chosen). The initial Maxwell-Boltzmann velocities are drawn at the *initial* temperature, not at the constant-temperature field, so a 1000 K quench starts where the schedule says it starts instead of spending its first picosecond catching up.

> **Tip**
> The MD Viewer plots the moving setpoint underneath the measured temperature, and the CSV export carries it as its own column (`target_temperature_K`). That second trace is what answers the only question an annealing run is asking — is the system following the schedule? — and it is invisible from the temperature alone. A constant-temperature run has no such series; it reports its single setpoint as a dashed reference line instead.

9.1.5 DFT and ORCA parameters

GPAW and VASP share the **Plane-wave cutoff** row and the **k-point grid**; Quantum ESPRESSO and SIESTA each have a settings group of their own (§9.1.6, §9.1.7), because the parameters that decide their basis are not the same parameters.

The cutoff is a property of the *plane-wave* basis, so under GPAW it is shown only in **PW** mode. In **FD** the basis is the real-space grid (controlled by **Grid spacing h**) and in **LCAO** it is the atomic orbital set — a cutoff there converges nothing, which is a trap worth not laying.

9.1.6 Quantum ESPRESSO settings

QE is a plane-wave code with a *dual* grid, and that is what its group is built around.

ecutwfc The cutoff for the wavefunctions, in Rydberg — the unit `pw.x` reads, so nothing is converted behind your back. Converge it against energy *differences*, not the absolute energy, which keeps falling long after everything you care about has stopped moving.

ecutrho The cutoff for the charge density, on QE's second grid. `auto` means QE's own default of 4× ecutwfc.

XC functional `input_dft`.

Occupations `smearing`, `fixed`, or one of the two tetrahedron methods. The smearing function and width appear only for `smearing`: the others take no width, and writing `degauss` beside them is how a QE input gets silently ignored.

Smearing Marzari–Vanderbilt ("cold") is QE's recommended default. Fermi–Dirac is the odd one out — it is a physical electronic temperature, not a convergence aid.

conv_thr The SCF threshold in Ry. A phonon or Born-charge run wants 10^{-10} or tighter, because it differentiates this quantity.

> **Requires**
> The 4× default for **ecutrho** is correct for *norm-conserving* pseudopotentials and badly under-converged for ultrasoft or PAW, whose augmentation charges are much harder than

the wavefunctions and want 8–12×. Leaving it on `auto` with a USPP library is the single most common way a QE run comes out quietly wrong. The note under the row restates the effective dual so the number is always in front of you.

9.1.7 SIESTA settings

SIESTA has no plane-wave cutoff, and the option is gone. Its basis is a finite set of numerical atomic orbitals; there is no plane-wave expansion to truncate. What the removed field used to do was set `MeshCutoff` — so raising it to "converge the basis" refined a real-space grid while the basis stayed exactly as small.

XC functional Resolved by ASE into SIESTA's `XC.functional` / `XC.authors` pair. `PZ`, `CA` and `PW92` are the LDA parametrizations; `DRSLL` and `VV` are van der Waals functionals.

Basis type `PAO.BasisType` — *how* the multiple-ζ orbitals are generated. `split` is the standard scheme and what almost every published SIESTA calculation uses.

Basis size `SZ`, `SZP`, `DZ`, `DZP`, `TZP` — how many orbitals per valence shell, and whether polarization orbitals are added. `DZP` is the standard production basis. **This is the parameter that plays the role a plane-wave cutoff plays elsewhere: it is what you converge.**

Energy shift `PAO.EnergyShift`: the energy by which confinement raises each orbital, which is what fixes its cutoff radius. *Smaller* means longer-ranged orbitals, a better basis and a more expensive run — the second knob to turn after the basis size. SIESTA's default is 0.02 Ry ≈ 0.27 eV.

Mesh cutoff `MeshCutoff`: the fineness of the real-space grid the Hartree and exchange-correlation terms are integrated on. Not a basis-set parameter. An under-converged mesh shows up as an "egg-box" force error as atoms move across grid points.

Under **Spin Configurations**, only the polarization *mode* is chosen here. The per-atom initial moments are a property of the structure: set them in **Edit Structure...** (§7.2), where you can see which atom is which, and they travel with the staged geometry into every calculation.

> **Note**
> For SIESTA, that mode reaches the generated script through ASE's own top-level `spin=` keyword (`non-polarized` / `collinear` / `non-collinear`) rather than a lone `SpinPolarized` `fdf` entry — the modern `Siesta` calculator always writes its own `Spin` line from that keyword, so a script relying on `SpinPolarized` alone left every run non-polarized regardless of what was requested. Verified end to end against a real `siesta` build.

ORCA has its own group:

ORCA method Editable: `B3LYP`, `PBE0`, `r2SCAN`, `wB97X-D3`, `HF`, `MP2`, or any ORCA keyword you type.

ORCA basis Editable: `def2-SVP`, `def2-TZVP`, `def2-TZVPP`, `cc-pVDZ`, `cc-pVTZ`, ...

Charge −10 to 10.

Multiplicity $2S + 1$, 1 to 11.

Solvation None (gas phase), **CPCM** or **SMD**, with **Solvent** naming the medium (water, acetonitrile, toluene, ...).

ASE writes the ORCA `.inp` file into the job directory and parses the output. The generated script contains an `EDIT ME` line for the path to your `orca` binary.

9.1.8 VASP parameters

Selecting VASP opens a **VASP settings** group exposing the primary INCAR tags. It is built into the shared calculator page, so every wizard that offers VASP shows the same controls.

POTCAR directory The PAW datasets, i.e. `VASP_PP_PATH`. Remembered across sessions and shared by every VASP run, since it describes the installation rather than a job. Both layouts work: the canonical one with a `potpaw_PBE/` level inside, and the flat one where the element folders sit directly in the directory — for the latter the generated script builds a symlink shim, because ASE cannot be told to look anywhere else.

XC functional ASE's `xc`, which expands to the matching `GGA`/`METAGGA` tag and its recommended defaults.

PREC / ALGO Grid accuracy and the electronic minimization. `Accurate` is the default: `Normal`'s coarser grid puts egg-box error into the forces, which is what a relaxation is most sensitive to.

NELM / EDIFF The SCF iteration cap and its energy threshold.

LREAL `Auto` is a large speed-up above roughly 20 atoms; `False` is exact and right for small cells.

Relaxation driver Who takes the ionic steps. Exactly one side may: VASP relaxes internally when `IBRION`/`NSW` say so, and ASE's optimizers relax anything that returns forces, so with both enabled *every force evaluation ASE asked for ran a complete VASP relaxation.*

ASE optimizer (the default) pins VASP to `IBRION` $= -1$, `NSW` $= 0$ and lets ASE take the steps. This is the mode the rest of the application is built around — geometry constraints, the variable-cell filters, the live streamed trajectory and the per-step metrics all come from ASE driving.

VASP internal relaxation creates no ASE optimizer at all; the tags below *are* the relaxation. Much faster per step, because VASP keeps the wavefunction and charge density between ionic steps instead of restarting — but the steps happen inside a single call, so they cannot be streamed or constrained from here. The ionic path is recovered from `OUTCAR` afterwards, so the trajectory tab and the Geometry Optimization Viewer work either way.

IBRION / ISIF / EDIFFG Shown only for a VASP-driven relaxation, since they describe nothing otherwise. A negative `EDIFFG` is a force criterion in eV/Å; a variable-cell relaxation is raised to `ISIF` ≥ 3 automatically.

Write `LCHARG` (on — CHGCAR is what every density analysis reads), `LWAVE`, `LAECHG` (the AECCARs a Bader analysis needs) and `LORBIT` $= 11$.

VASP writes its density as a `CHGCAR`, not as `.cube` files, and Calango registers it in the **Volumetric Data** dock when the run finishes — along with `AECCAR0`/`AECCAR2`, `LOCPOT` and `ELFCAR` when those were requested. That same `CHGCAR` is what an **Electronics** ⟩ **Electronic Structure** run reuses: with a VASP baseline selected it copies the file in and diagonalizes along the band path with `ICHARG` $= 11$, running no SCF of its own.

NCORE / KPAR Left at **auto** unless you know the machine: a wrong value is a performance cliff, not an error.

Extra INCAR tags One `TAG = value` per line, applied verbatim on top of everything above. No dialog can cover 300 INCAR flags, and one that tries becomes a ceiling.

> **Note**
> `ISMEAR` and `SIGMA` come from the shared **Smearing method** row rather than from this group, so one control drives every engine. Fermi-Dirac maps to `ISMEAR` = −1, Gaussian to 0 and Methfessel-Paxton to 1. **None** maps to a very narrow Gaussian rather than to the tetrahedron method `ISMEAR` = −5, which VASP refuses on the Γ-only meshes that small test cells use.
> `MAGMOM` is not set here either: it comes from the structure's initial magnetic moments, and ASE writes it out of the atoms object.

9.1.9 MACE models

MACE model **MACE-MP-0 (universal, materials)**, **MACE-OFF (universal, organic molecules)**, or **Custom trained model (file)**.

MACE model size `small`, `medium` (default), `large`.

Custom model file A `.model` or `.pt` checkpoint.

MACE device `cpu`, `cuda` or `mps`.

Foundation models download automatically on first use and are cached in `~/.cache/mace`.

9.2 The script editor

The right-hand pane always shows the complete, standalone ASE script the form describes — syntax-highlighted and fully editable.

Start typing in it and Calango shows *edited — form sync paused* in amber: from then on the form no longer overwrites your text, so hand edits are never silently lost. **Regenerate** discards them and re-syncs. **Save Script...** writes the current text to a `.py` file.

> **Tip**
> This is the intended escape hatch for anything the GUI does not expose. Configure what you can in the form, then edit the script for the rest — constraints, custom observers, a different optimiser. The script is ordinary ASE and runs unmodified on a cluster.

9.3 Execution environments

The **Execution Environment** group decides which Python actually runs the job — independent of the interpreter Calango embeds.

Leave it empty The job uses the embedded interpreter. The status line names it.

Env Folder... Point at a Conda environment directory; Calango finds `bin/python` inside it.

Python... Point directly at an interpreter.

The status line confirms the resolution live, turning red when no interpreter can be found at the given path. The setting is remembered between sessions, and the same selector appears in the phonon builder and the band-structure dialog.

> **Tip**
> This is how you keep heavyweight solver stacks isolated: a GPAW environment for band structures, a `mace-torch` environment with CUDA for ML potentials, and a minimal ASE environment for Calango itself.

9.4 The Job panel

Jobs run as separate processes in their own directory, so a crashing solver cannot take the GUI with it. The **Job** dock (zone 10) has five tabs.

9.4.1 Log

A status label (*Idle*, *Running…*, *Finished (exit N)*, *Crashed*), a progress bar, a **Kill** button, and the live output. Standard error is coloured red, the start line blue, and a clean exit green.

9.4.2 Metric plots

Four tabs plot job observables live, parsed from markers the generated scripts print:

Tab	Quantity	Unit	Reference line
Energy	Total energy	eV	—
Temperature	Ionic temperature	K	thermostat setpoint
Force	Max $\|F\|$	eV/Å	—
Pressure	Scalar pressure	GPa	barostat setpoint

Each plot shows the running series with the latest value read out numerically, and a dashed reference line at the setpoint for thermostatted or barostatted runs — so drift away from the target is visible at a glance. Every tab has its own **Export Data…** button writing CSV or whitespace-separated `.dat`. The series are saved in the project file, so a reopened project shows the plots from the run that produced it.

9.5 Live trajectory streaming

MD runs and geometry relaxations do not make you wait. The moment the job starts, Calango opens a trajectory tab marked *(live)* and streams each newly computed geometry into it as the simulation produces it.

- The timeline grows as frames arrive, and follows the newest frame automatically.
- Scrub back to an earlier frame and the view stays there — Calango stops following so you can inspect while the run continues.
- Everything else works normally on the partial trajectory: measurement, colour mapping, representation changes.
- When the job finishes the *(live)* marker is dropped and the tab becomes an ordinary trajectory.

Relaxations stream every step; MD streams at an interval chosen so a run produces at most about 400 streamed frames, keeping long simulations responsive. The complete trajectory is still written to `opt.traj` or `md.traj` in the job directory.

9.6 What happens when a job finishes

Calango inspects the job directory and does the useful thing:

- A live-streamed run keeps the tab it already built.
- An electronic-structure run opens the band/PDOS viewer (Chapter 12).
- A trajectory (`md.traj`, `opt.traj`) opens in a new tab.
- Otherwise, if a final geometry exists (`optimized.extxyz`), Calango offers to load it.

The process manager entry switches to *completed* or *failed*, and its directory link remains available for the rest of the session and beyond.

9.7 The MLIP dataset manager

Simulation ⟩ Dataset Manager (MLIP) assembles training datasets for machine-learning interatomic potentials from the trajectories you have generated — MD runs, noise ensembles, relaxation paths — and partitions them reproducibly.

9.7.1 Loading structures

Add Files... accepts multiple files of mixed provenance: `.extxyz`, `.traj`, `.cif`, VASP files, ASE databases. Each is listed with its frame count and the chemical formulas it contains, so a heterogeneous dataset stays legible. The summary line reports the total.

9.7.2 Splitting

Training, Validation Percentages; **Test** is the remainder, shown live.

Random seed Default 42.

The split is deterministic: the same frame count, percentages and seed always produce exactly the same partition, and the three subsets are disjoint and cover every frame.

9.7.3 Query-by-Committee ensembles

Active learning needs an ensemble of models trained on different data, so their disagreement can be used as an uncertainty estimate.

Committee members 1 exports a single dataset; $N > 1$ adds `committee_01...committee_N` subdirectories, each with its own training set.

Diversity How the members are made to differ:

- **Independent splits (seed + k)** re-splits the non-test pool with a different seed per member, keeping one common test set.

- **Bootstrap resampling of the training set** draws each member's training set with replacement — the classical bagging construction, which reuses roughly 63 % of the original frames per member.

9.7.4 Export

Extended XYZ (MACE-ready) Writes `train.extxyz`, `valid.extxyz` and `test.extxyz` (plus committee subdirectories), the layout MACE and most training scripts expect.

ASE database (.db) One SQLite database per subset.

> **Note**
> Export goes through `ase.io` directly rather than Calango's internal structure model, specifically so that energies, forces and stresses attached to the frames survive into the training files. A dataset exported from a finished MD run carries the labels a potential needs to train on.

9.8 The MLIP trainer

Modules ⟩ **MLIP** ⟩ **Trainer** is a wizard, one decision per page: **Framework**, then the framework's own parameter pages (**Dataset**, **Model**, **Training** for MACE), then **Config and launch**. It is a fixed height and every parameter page scrolls inside it, so no future addition can push it off a laptop screen.

9.8.1 Which framework

Every machine-learning potential Calango knows how to *run* is listed, and exactly one has an implemented trainer:

Framework	Trainer	Config / entry point
MACE	**implemented**	YAML, `mace.cli.run_train`
DeePMD-kit	not yet supported	JSON, `dp train`
NequIP	not yet supported	YAML, `nequip-train`
Allegro	not yet supported	YAML, `nequip-train`
CHGNet	not yet supported	Python API, no CLI
MatterSim	not yet supported	CLI flags, fine-tuning only
FAIRChem / OCP	not yet supported	YAML fragments, LMDB datasets

The unsupported six are listed rather than hidden — hiding them answers "can Calango train a NequIP model?" with silence, and silence reads as *look harder*. Selecting one shows what its trainer reads and runs, and leaves **Next** disabled: the gate is the button, not the row, so the reason can be read.

9.8.2 The MACE parameter pages

Dataset carries the training and validation files, the **Energy key** / **Forces key** (`energy`/`forces`, what ASE writes, versus MACE's own `REF_*`) and the isolated-atom energies. Getting the keys wrong does not fail the run — MACE warns, sets the corresponding loss weight to zero and trains on nothing — and E0s is not optional at all: without it MACE aborts before the first epoch.

Model carries the size preset, the cutoff radius, the channel count and the equivariance order, with

the two architecture constants (interaction layers, correlation order) under **Advanced**. **Training** carries the learning rate, batch size, epochs, device, seed and the per-property loss weights, with the schedule, the two-phase (SWA/EMA) loss and the Query-by-Committee ensemble under its own **Advanced** group.

9.8.3 The config has the last word

The final page shows the generated `mace_train.yaml` in a monospaced, editable view. **Whatever is in that editor is what gets written and what runs**, verbatim, hand edits included. A change on an earlier page regenerates the text — unless you have edited it, in which case the page says the settings moved under it and leaves your text alone. **Regenerate from settings** rebuilds it from the pages, and asks first.

The same page carries the execution-environment selector and **Check Environment**, which probes the chosen interpreter for the framework's package (reporting its version, recorded as a comment in the generated config) and for which PyTorch devices it can actually use. Both **Run (Local)** and **Run (Remote)** re-run that check before launching anything.

CHAPTER 10

Free energies and alloy thermodynamics

Three modules answer questions that a single energy cannot. Thermodynamic integration produces an *absolute* free energy from molecular dynamics; the ECI fit turns a set of computed alloy configurations into a predictive model; and the Cluster Variation Method turns that model into a configurational entropy and, where the lattice allows it, an order–disorder temperature.

10.1 Thermodynamic integration

Simulation ⟩ Thermodynamic Integration… computes F, and $G = F + PV$, in absolute terms.

That is not something a simulation can measure. Energy, temperature, pressure and stress are all mechanical averages over a trajectory; the free energy is the logarithm of a partition function, and no average of any observable equals it. Thermodynamic integration recovers it by building a *reversible* path from a reference system whose free energy is known in closed form:

$$U(\lambda) = (1-\lambda)\,U_{\mathrm{ref}} + \lambda\,U_{\mathrm{target}}, \qquad \frac{\partial F}{\partial \lambda} = \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_{\lambda} = \langle U_{\mathrm{target}} - U_{\mathrm{ref}} \rangle_{\lambda}, \tag{10.1}$$

$$\Delta F = \int_0^1 \langle U_{\mathrm{target}} - U_{\mathrm{ref}} \rangle_{\lambda}\, \mathrm{d}\lambda, \qquad F = F_{\mathrm{ref}} + \Delta F, \qquad G = F + PV. \tag{10.2}$$

Each λ is one MD run — a *window* — reporting one number. Every energy is per simulation cell, matching the Phonon Thermodynamics module's own convention; per-atom values are reported beside them with the convention spelled out in the field names.

10.1.1 The reference systems

The method rests entirely on F_{ref} being *exact*. An approximate reference does not add noise; it adds a bias no amount of MD removes.

Ideal gas (Sackur-Tetrode) $F_{\mathrm{id}} = k_{\mathrm{B}}T \sum_s N_s[\ln(\rho_s \Lambda_s^3) - 1]$, summed over *species*: atoms are grouped by mass, and each group carries its own thermal wavelength and its own $N!$. The factorial is the point — dropping it gives a number that scales almost right and is wrong by $Nk_{\mathrm{B}}T(\ln N - 1)$. Exact, with no caveat, and the natural reference for a liquid. It is also the reference with the worst endpoint behaviour (§10.1.2).

Einstein crystal (Frenkel-Ladd) $F_{\mathrm{Ein}} = 3Nk_{\mathrm{B}}T \ln(\beta\hbar\omega)$ with $\omega = \sqrt{\alpha/m}$. Exact, and free of the endpoint singularity because tethered atoms never overlap. There is deliberately *no* $1/N!$ here — the opposite of the ideal-gas case, and exactly where a copy-paste between the two goes wrong: Einstein atoms are tethered to distinguishable sites, cannot exchange, and a crystal samples one permutation. The **Hold the centre of mass fixed** box drives *both* the

`FixCom` constraint in the run and the matching $O(\ln N)$ finite-size correction in the closed form, so the two cannot disagree.

Lennard-Jones fluid See the warning below.

> **Requires**
>
> **The Lennard-Jones reference cannot carry a liquid.** There is no exact closed form for the free energy of an LJ fluid. Calango evaluates the exact Hirschfelder–Curtiss–Bird second-virial series, $\beta A_{\mathrm{ex}}/N = B_2\rho + O(\rho^2)$ — an exact expansion, not a fit — which is quantitative only for a reduced density $\rho^* = (N/V)\sigma^3 \leq 0.05$. Above that the reference is marked invalid and no free energy is reported at all.
> A liquid is an order of magnitude denser than that, so **for an actual liquid use the ideal gas or the Einstein crystal**. The LJ entry exists to put an LJ fluid *on* a path, as the target of its own ideal-gas integration whose excess free energy you then supply back — not to start one at liquid density. The alternative would have been to transcribe a 33-parameter equation-of-state fit; an unverifiable table of coefficients that silently produces plausible numbers is worse than a function that refuses.

10.1.2 The endpoint singularity

This is the failure mode of the method. With linear coupling (10.1) and an ideal-gas reference, atoms at small λ are almost free and may sit on top of one another; a target with a hard repulsive core then charges an unbounded energy for those configurations, and $\langle \partial U/\partial\lambda \rangle$ diverges as $\lambda \to 0$. A uniform grid integrates that divergence without noticing: every window returns a finite number, the quadrature returns a finite number, and the answer is wrong.

Three layers stand against it.

1. **The default schedule never samples $\lambda = 0$.** Gauss–Legendre nodes are strictly interior to $[0, 1]$ and already cluster towards both ends, which is where the integrand bends.

2. **A runtime detector.** The variance and the magnitude of the two end windows are compared against the *median over the path* — a ratio, not a fixed energy, because the integrand's scale is whatever the cohesive energy happens to be and only its *shape* says whether a divergence is being integrated. Above 8× either way, the result carries the diagnosis and both ratios.

3. **A warning in the wizard.** Selecting the ideal gas together with a schedule whose first node is $\lambda = 0$ (Uniform, or Power law) turns a red note on the settings page, before anything has been paid for.

> **Note**
>
> **There is no soft-core coupling, and there cannot be one here.** The textbook fix replaces the linear mixing with a coupling that regularises the *pair form* of the potential. Calango's target is an arbitrary ASE calculator — MACE, GPAW, xTB, a LAMMPS force field — and none of them exposes a pair form to modify. There is no general way to soften a machine-learning potential or a self-consistent DFT energy. So the module avoids the endpoint, detects it when it bites anyway, and says so; it does not pretend to have solved it.

10.1.3 Settings

λ schedule Gauss–Legendre (interior nodes) by default; also Uniform, Power law (clustered at $\lambda = 0$) and Clustered at both ends.

Quadrature Gauss–Legendre, Simpson or trapezoid. Gauss weights on non-Gauss nodes are not an approximation of anything, so that combination is *refused* rather than silently reweighted; Simpson needs a uniform grid and falls back to the trapezoid, saying so. The two combos are interlocked in the wizard for the same reason.

λ windows Default 12. Each is an independent MD run, and this is also the knob that fixes the quadrature error.

Dispatch as separate jobs Default 1. See the warning below.

Also sweep λ backwards The hysteresis check. A reversible path must give the same ΔF either way; it will not if the windows were under-equilibrated, if the system changed phase along the path, or if the coupling drove it through a barrier — and none of those looks like noise. Doubles the cost and needs one sequential chain, so it is withdrawn when the run is split.

Ensemble NVT–Langevin by default. **NVE is deliberately absent**: $\langle \partial U/\partial\lambda \rangle$ must be a canonical average at *one* temperature, and a microcanonical window sits at a different temperature for every λ. NPT is offered but warned about — the reference free energy is a function of V, and under a barostat there is no single volume to evaluate it at.

Equilibration steps / window Default 2000, and not optional in practice. Averaging over the transient biases every window in the *same* direction, so the bias survives the λ integral instead of cancelling — and it looks nothing like noise on a plot.

Production steps / window, Sampling interval 10 000 and 10.

> **Requires**
>
> **λ windows run sequentially.** Splitting the path into jobs produces contiguous slices that go through the ordinary job queue, which has a single runner, so they execute one after another exactly as the unsplit path would. What splitting buys is *resumability* — a dead slice costs its own windows, not the path — and *cluster dispatch*. It is not a wall-time speed-up, and should not be budgeted as one.

10.1.4 Two error bars, and which one to spend on

The generated script writes the **raw** $\partial U/\partial\lambda$ series, not merely its mean and variance: the autocorrelation time cannot be recovered from moments, and MD samples are not independent. Calango computes the integrated autocorrelation time by **Sokal's automatic windowing** — truncated, because the tail of the correlogram is pure noise whose variance grows with the number of terms added, so the naive full sum does not merely lose precision, it does not converge — and reports

$$\sigma_{\mathrm{mean}} = \sqrt{\frac{\mathrm{var} \cdot \tau_{\mathrm{int}}}{N}}. \tag{10.3}$$

A naive $\sigma/\sqrt{N}$ over correlated samples under-reports by $\sqrt{2\tau}$, routinely a factor of three: exactly the size of the discrepancies people then explain away. A block-averaging estimate is computed independently as a cross-check, and the two disagreeing by more than 2.5× is reported as a production run too short for either.

Because every rule here is linear, the quadrature is expressed as *weights* rather than as a number, which lets the statistical error propagate exactly:

$$\sigma_I = \sqrt{\textstyle\sum_i w_i^2 \sigma_i^2}. \tag{10.4}$$

Separately, a **λ-grid error** is estimated by comparing the chosen rule against a closed trapezoid on the same nodes and on every second node. When it exceeds 3× the statistical error the report says so in as many words: *the lambda grid, not the sampling, dominates the error — add windows rather than MD steps.*

10.1.5 Partial failure

The number of windows the run was *supposed* to have is passed to the integrator separately from the list that came back; passing the length of the list defeats the check, which is the whole reason the parameter exists.

A path with a dead window is not a noisier path, it is a *different integral*: quadrature weights are a property of the node set, so dropping a node and reweighting the survivors integrates a curve that was never sampled. A run missing any window therefore reports

```
*** INCOMPLETE -- NO FREE ENERGY IS REPORTED ***
```

and leaves ΔF at zero. The summary distinguishes *missing* (no file — the window never ran) from *failed* (a file with a failure status — it ran and died), which is what tells you whether to re-run a slice or to change the physics.

10.1.6 The integrand viewer

The free energy is a *number*; what goes wrong is a *shape*. Every characteristic TI failure is visible in the integrand and invisible in ΔF, so the plot is not decoration. It shows $\langle \partial U/\partial\lambda \rangle$ against λ with

- 1σ error bars from the *autocorrelation-corrected* estimate, not the raw variance — which understates a correlated series by exactly the factor this module exists to compute;
- the shaded integral, because ΔF *is* that area;
- failed windows as vertical rules, so a gap is visible rather than inferred from a warning list;
- the endpoint band, when the detector fired;
- forward and backward sweeps overlaid when the hysteresis check ran.

It opens *by itself* when a run finishes. There is no menu entry for it: an "open the results" action whose only job is to ask which file you meant is a question the application can already answer. The other three ways in all skip that question — the **Load Results...** button on the wizard's *Integration Path & Sampling* stage, the same button on this window (which walks from one run to the next), and the Processes panel, which offers **Thermodynamic Integration Viewer** on any job directory holding a `ti.json`. **Export Results...** writes the per-window table as CSV — with the run's conditions and the assembled F, PV and G as commented header lines — or the report text as shown. The quadrature, the statistics, the closed-form references and the endpoint diagnostics all run *at read time* in C++, so a run downloaded from a cluster is analysed by the same code as a local one.

10.1.7 A worked example

Liquid copper, 32 atoms at 1600 K, MACE as the target, ideal-gas reference, five Gauss–Legendre windows. This was a development run rather than a shipped test, so the numbers are reported and not asserted.

Quantity	Value
$\langle \partial U/\partial\lambda \rangle$ across the path	$-53.6 \rightarrow -120.9$ eV
Per-sample variance across the path	$301 \rightarrow 1.17$ — a factor of **257**
Endpoint detector	**fired**, at $\times 107$ against the path median
ΔF	-106.35 eV
Statistical error	0.46 eV
λ-grid error	**6.57 eV**

The conclusion is the opposite of the instinct: **more MD time would buy nothing.** The sampling error is already 0.46 eV and the grid error is 6.57 eV — 14× larger. Five nodes cannot resolve an integrand whose variance changes by a factor of 257 along the path. The run needs more windows, clustered harder towards $\lambda = 0$; longer production runs would shrink the smaller of the two errors and leave the answer exactly as wrong. This is precisely why the two errors are reported separately rather than added.

Requires

Only EMT and MACE have been exercised end to end. GPAW and xTB targets are supported by construction — the target is an ordinary ASE calculator and the coupling is engine-agnostic — but they are untested, and a DFT target pays a full SCF per MD step, per window. The C++ half is covered by the `thermodynamic_integration` test (161 assertions): the closed-form references against their analytic values, the Gauss–Legendre rule against the polynomial exactness that defines it, the autocorrelation estimator against block averaging, and the refusal paths.

10.2 Effective cluster interactions

Modules ⟩ Alloys ⟩ Effective Cluster Interactions (ECI Fit)… turns a completed Cluster Expansion run into a predictive model,

$$E(\sigma)/\text{atom} = J_0 + \sum_{\alpha} m_\alpha J_\alpha \Phi_\alpha(\sigma), \tag{10.5}$$

by regressing the computed energies on the cluster correlations Φ_α that the builder emitted alongside them.

10.2.1 Why a plain least-squares solve is a trap

A cluster-expansion design matrix is ill-conditioned *by construction*: orbits at nearly the same radius have nearly the same correlation on every configuration one can afford to compute, so their columns are near-parallel; the empty cluster is exactly parallel to the intercept; and the number of candidate orbits grows faster with the cutoff than the number of configurations anyone is willing to run, so the system is frequently underdetermined outright. Forming $X^{\mathsf{T}}X$ squares the condition number of an already borderline matrix and then inverts it. The failure is not a crash — it is a fit with a perfect training residual and ECIs of alternating sign and absurd magnitude that predicts nothing. Every solver here goes through an SVD or through coordinate descent, never through an explicit inverse.

10.2.2 Methods

LASSO (selects clusters) L_1. *Selects* orbits, driving them to exactly zero — bit for bit, not merely small. The default, because deciding which clusters carry the physics is the central difficulty of a cluster expansion and this is the method that decides.

Ridge (keeps all clusters) L_2. Keeps every orbit and only shrinks it; never sparse. It cannot tell you that a cluster does not matter, only that its coefficient is small, which is a different statement.

ARD / Bayesian Learns one precision per orbit by evidence maximisation and prunes outright, with no regularisation parameter to choose. Sharper support recovery than lasso when it works; slower, and it can prune too hard on very small data sets.

Columns are centred and scaled before fitting and unscaled afterwards; without that, neither penalty being scale-invariant, the *units* of an orbit would decide how hard it is penalised.

10.2.3 The CV score is the diagnostic

Requires

A cluster expansion is **not** judged by its training RMSE but by its **cross-validation score**, the RMS error on configurations the fit never saw. The two numbers moving in opposite directions as the regularisation is relaxed *is* the definition of overfitting, so both are shown side by side. A fit whose CV score exceeds 3× the training error is flagged in red: *this expansion fits its own data and does not predict.*

CV folds defaults to leave-one-out, which is the cluster-expansion convention — "the CV score" in the CE literature *means* the leave-one-out score. The fitted orbits are ranked by $m \cdot J$ rather than by J: a large ECI on a multiplicity-1 orbit and a small one on a multiplicity-24 orbit can be the same physics.

A run made *before* correlations were emitted has energies and no design matrix, and is **refused with instructions** rather than fitted against whatever else is present — the alternative being an ECI file of zeros that the CVM downstream would turn into a plausible and entirely wrong entropy curve.

The builder now defaults the **Triplet cutoff** to 3.0 Å (on), because a pair-only basis is symmetric under A↔B at complementary compositions and cannot distinguish A_3B from AB_3 — an ensemble built without triplets silently discards the term that chooses the ordered structure. **Quadruplet cutoff** stays off: each orbit adds K^4 histogram columns, and with the usual handful of configurations the fit runs out of samples before it runs out of clusters.

10.2.4 Send to CVM

Send to CVM… opens the CVM module with the fitted nearest-neighbour pair ECI already converted and applied.

Requires

In the ±1 correlation basis the pair energy is Js_is_j with $s = +1$ for A and -1 for B, so

$$e_{AB} = -J_2, \qquad e_{AA} = e_{BB} = +J_2 \qquad \Longrightarrow \qquad J_2 > 0 \text{ ORDERS.} \tag{10.6}$$

Half the literature writes the opposite convention. Getting it backwards turns an ordering alloy into a clustering one and *nothing fails* — you get a smooth, plausible entropy curve for the wrong material. That is why the conversion is done by shared code and why the receiving window shows the provenance of the number it was given.

10.3 The Cluster Variation Method

Modules ⟩ Alloys ⟩ CVM / Alloy Thermodynamics... computes the configurational entropy of a substitutional alloy, and — with the long-range-order option — its order–disorder temperature.

The question is how much of an alloy's entropy is really $k_B \ln(\text{arrangements})$. The textbook $S_{ideal} = -k_B \sum_i x_i \ln x_i$ assumes the species are placed *independently* on every site. They are not: if unlike neighbours are favoured the alloy orders, if like neighbours are favoured it clusters, and either way there are fewer distinguishable arrangements than the ideal count. S_{ideal} is an *upper bound* — and for a high-entropy alloy advertised as stabilised by configurational entropy, the gap between the bound ($\ln 5 = 1.609\, k_B$ for an equiatomic quinary) and the truth is the entire question.

Kikuchi writes the entropy as a sum over clusters with alternating-sign corrections, so that correlations already accounted for by a cluster are not counted again by its subclusters:

$$S/k_B = -\sum_{\alpha} a_\alpha \sum_{\text{configs}} \rho_\alpha \ln \rho_\alpha, \tag{10.7}$$

with a_α the Kikuchi–Barker coefficients, fixed by requiring every subcluster's contribution to cancel down to exactly one net counting.

10.3.1 A hierarchy, not a menu

Approximation	Also known as	What it sees
Point	Bragg–Williams	Nothing. Sites independent; this *is* S_{ideal}, and no interaction enters.
Pair	Bethe–Peierls–Guggenheim	Nearest-neighbour pair correlations. **Exact on a 1D chain.**
Tetrahedron	Kikuchi	Points, pairs, triangles and tetrahedra — the smallest cluster containing the frustrated triangles FCC is built from.

Each is the previous one plus more correlation, so **computing all three is the deliverable**: the spread between them *is* the size of the correlation correction, and a single number from any one of them does not say whether it mattered. For an FCC binary at $x = 0.5$ with $e_{AB} = -0.02\,\text{eV}$, at 500 K:

$$S_{pair} = 0.5358 \;<\; S_{tet} = 0.6220 \;<\; S_{ideal} = 0.6931 \quad k_B. \tag{10.8}$$

The ordering is the physically meaningful part. The pair approximation cannot see the frustration of the close-packed tetrahedron — on FCC one cannot make every nearest-neighbour triangle unlike-decorated — so it believes the alloy orders more than it does, and therefore *over*estimates ordering.

> **Tip**
>
> **Warren–Cowley** α is reported at every temperature, and it is what makes the temperature dependence legible: S alone does not say *which way* the alloy departs from random. $\alpha = 1 - P(j|i)/x_j$ is zero for a random alloy, negative for ordering and positive for clustering. It is also the more sensitive probe — the residual short-range order is *first* order in βJ while the entropy deviation is *second* order — and in the disordered phase it is directly comparable against a diffuse-scattering measurement, which the entropy is not.

10.3.2 Triplets

The tetrahedron approximation consumes a nearest-neighbour *triplet* interaction as well as a pair one, and this changes what the model can express at all. A binary with pair interactions only

is exactly symmetric under A↔B at complementary compositions, so it assigns $x = 0.25$ and $x = 0.75$ the same energy and cannot distinguish A_3B from AB_3. On an FCC tetrahedron at 600 K, $E(x{=}0.25) - E(x{=}0.75)$ measures 9.4×10^{-17} eV pair-only — machine zero, an identity rather than a small residual — and 2.41×10^{-2} eV with a triplet ECI.

A triangle is not a subcluster of a pair, so a pair model has nowhere to put one. Triplets supplied with the Pair approximation are *dropped*, with the warning that the numbers below describe a pair-only model and must not be read as the cluster expansion that was fitted. The window itself exposes a single e_{AB}; triplets reach the solver through the Orchestration canvas, whose CVM node supplies them only when the approximation is the tetrahedron.

10.3.3 Long-range order and T_c

Requires

The homogeneous solver cannot produce an order–disorder transition at all. One sublattice means every site is statistically identical and there is no symmetry left to break. Run Cu_3Au through it and you get a perfectly smooth $S(T)$ and $\alpha(T)$ straight through 663 K — not a bug, but the approximation stating its own limits. For a transition, tick **Allow long-range order (4 sublattices)**.

The FCC nearest-neighbour tetrahedron has exactly one vertex on each of the four simple-cubic sublattices FCC decomposes into, so the tetrahedron distribution can be indexed by *position* = sublattice and *value* = species — and, unlike the homogeneous case, it is not symmetric under permuting the positions. That asymmetry *is* the long-range order. Only the overall composition is constrained; the per-sublattice compositions are free, which is what lets $L1_2$ form at $x = 1/4$ with three A-rich sublattices and one B-rich one.

Both branches are solved at every temperature and T_c is located by **which has the lower free energy** — by the free energies crossing, not by an order parameter decaying to zero. The FCC $L1_2$ transition is **first order** and η *jumps*; a search for $\eta \to 0$ would either find nothing or find the spinodal, which is above T_c and is not the transition.

Ticking the box also snaps the composition to the structure's stoichiometry ($L1_2 \leftrightarrow x = 1/4$, $L1_0 \leftrightarrow x = 1/2$) — without which asking for $L1_2$ at $x = 0.5$ correctly reports "no transition" and reads as a broken feature — and raises the minimum temperature, for the reason in §10.3.4.

Cu_3Au. The tetrahedron CVM gives $k_B T_c = 1.9245\, V_2$ for $L1_2$ at $x = 1/4$, against $3.2806\, V_2$ from Bragg–Williams on the same system; $V_2 = 29.69$ meV reproduces the measured 663 K. The mean-field comparison is a *provable* check rather than a tolerance: for $L1_0$ at $x = 1/2$ the Bragg–Williams transition has the closed form $k_B T_c = 4V_2$ exactly — derived by linearising the mean-field equations, and equivalently by Fourier-transforming the FCC nearest-neighbour interaction, whose minimum sits at the X point with $J(X) = -4V_2$ — which the solver returns as 3.99972, and one-sidedly: the mean-field $L1_0$ transition is *second* order, so both the order parameter and the free-energy difference vanish continuously at T_c and the finite thresholds the bisection needs bite a hair below the true temperature. CVM must come out *below* mean field because it includes correlations mean field ignores, and FCC is frustrated enough that it is not merely a little below.

Note

The bracket **MC < CVM < mean field** is the robust statement, and both its ends are provable. The particular literature values often quoted for the middle — about 1.89 for the tetrahedron CVM, about 1.74 for Monte Carlo, and the 25–30 meV window for the Cu–Au

nearest-neighbour V_2 — are recalled from standard literature and are *not* traced to a cited source anywhere in this codebase. Calango measures 1.893 for $L1_0$ at $x = 1/2$ and 1.9245 for $L1_2$ at $x = 1/4$; treat the agreement as encouraging rather than as validation against a reference you can look up from here.

10.3.4 What this is not

Requires

This is not a phase diagram. Neither solver performs a common-tangent construction, and both work at *fixed composition*, comparing homogeneous phases. There are therefore no two-phase fields, no solvus lines, and no composition range over which an ordered phase is stable.

- **Nearest-neighbour interactions only.** No second-neighbour terms, which on FCC are what select between competing ordered structures in several systems. The four-sublattice solver additionally takes pair interactions only.
- **BCC borrows FCC's tetrahedron counting and is not quantitative.** The Kikuchi–Barker coefficients depend on how many pairs, triangles and tetrahedra share a site, which is a property of the lattice and nothing else; BCC needs the *irregular* tetrahedron built from first and second neighbours. Choosing BCC does not fail loudly — it produces a smooth, plausible, wrong entropy — so the run says so in its warnings.
- **Configurational entropy only**; no vibrational, electronic or magnetic contribution. Long-range order is FCC-only ($L1_2$, $L1_0$).

Two pathologies of the truncated Kikuchi expansion on frustrated FCC are reported rather than hidden, and neither is a bug in the iteration.

1. **Negative entropy on the disordered branch.** Well below the transition the *disordered* branch returns $S < 0$ — at $x = 1/2$ it tends to $-0.929\,k_B$. The Kikuchi expansion is not a probability and nothing forces its truncation to stay positive on a lattice this frustrated. It never affects the stable branch or T_c, which is what you read; it does mean the disordered free energy is meaningless deep inside the ordered field.
2. **A domain-mixed fixed point.** At $x = 1/2$, below roughly $0.35\,T_c$, the $L1_0$ start converges onto an equal mixture of the two symmetry-related $L1_0$ domains: exactly the ground-state energy, exactly $S = \ln 2$, and *zero* long-range order, because the point marginals of the two domains average away while their pair correlations do not. It is a fixed point only because zeros in the pair marginals are absorbing under a multiplicative update, and a real crystal would pay a domain wall. Solutions with vanishing order parameter are therefore never accepted as ordered, and hitting one is reported — which is why an $L1_0$ transition must be bracketed from inside the ordered field, and why ticking the checkbox raises the minimum temperature. $L1_2$ at $x = 1/4$ does not suffer from this at any temperature.

10.3.5 Validation

The tests check identities, not previous outputs. The pair approximation is *exact* on a 1D chain, and the 1D Ising chain has a closed-form transfer-matrix solution: Calango matches it to $10^{-9}\,k_B$ in entropy and 10^{-12} eV in energy, measuring residuals of order 10^{-16}. Both signs of J are run, because a sign error in the energy convention passes one and fails the other.

The zero-interaction limit pins the FCC Kikuchi–Barker coefficients $(a_4, a_3, a_2, a_1) = (2, 0, -6, 5)$

— the triangles genuinely vanish — together with the composition multiplier, by requiring S to be *exactly* S_{ideal} when nothing interacts. It is checked at $x = 0.3$, not $x = 0.5$, and measures $1.7 \times 10^{-14}\, k_{\text{B}}$. The composition is the point: an earlier derivation dropped the composition multiplier, giving $w \sim (\prod x)^{7/8}$, which at $x = 0.5$ is completely invisible — every tuple picks up the same factor and normalisation hides it — and wrong everywhere else. The four-sublattice version of the same identity runs at $x = 0.25$ and measures $2.2 \times 10^{-16}\, k_{\text{B}}$.

10.4 EGQCA

Modules ⟩ Alloys ⟩ EGQCA (Alloy Thermodynamics)... is a third route to a binary alloy's mixing free energy, alongside ECI-Fit→CVM above: it solves the Gibbs mixing free energy directly from a small, explicit ensemble of non-equivalent cluster configurations, with no effective model fitted in between.

Working theory: P.N. Ferreira *et al.*, *Ab initio modeling of superconducting alloys*, Materials Today Physics **48**, 101547 (2024). A homogeneous solid solution $A_{1-x}B_x$ is represented as an ensemble of M small, independent supercells falling into J non-equivalent classes by symmetry, each with its own energy E_j, degeneracy g_j and composition n_j. Minimizing $\Delta G = \Delta H - T\Delta S + \Delta A$ over the cluster occurrence probabilities p_j, subject to $\sum_j p_j = 1$ and $\sum_j n_j p_j = nx$, gives a closed-form solution: every p_j is a Boltzmann-weighted power of one Lagrange multiplier η, and η is the unique positive root of one polynomial — solved by bisection, so the whole (x, T) grid is milliseconds even when wide.

"Extended" over the original GQCA: each cluster may carry a phonon density of states, whose harmonic vibrational free energy enters as ΔA — a genuinely temperature-dependent correction beyond the enthalpy and configurational entropy of plain GQCA. Omitting the DOS on every cluster reduces exactly to ordinary GQCA; mixing a vibrational and a non-vibrational cluster in one sum is refused outright, since it is not thermodynamically consistent.

EGQCA is an **analysis layer over a finished Cluster Expansion Calculation**, not a job of its own: it reads that run's `cluster_expansion.json`, requires a degeneracy g_j on every configuration and both pure end-members, and is a strictly **binary** theory — an ensemble with more than two species opens with a note explaining why rather than silently projecting onto two of them. Once loaded it exposes editable reference enthalpies H_A, H_B (defaulted to the ensemble's own end-member energies), a composition range and step count, and a temperature range and step count. Output is two native plots — $\Delta G/M$ against composition, one curve per temperature, and the cluster occurrence probabilities p_j against temperature at the composition nearest $x = 0.5$ — both exportable to CSV, alongside the Kullback-Leibler divergence against the ideal cluster distribution: the paper's own diagnostic for how far the alloy has departed from complete randomness, vanishing exactly when interactions vanish.

10.5 DSIM

Modules ⟩ Alloys ⟩ DSIM (Dilute Solution Interpolation)... computes a binary A-B solid solution's mixing enthalpy $\Delta H_{\text{mix}}(x)$, interpolated between its two dilute-solution limits, from exactly **four** calculations — none at any intermediate composition.

Working theory: L. Seixas *et al.*, *Enthalpies of mixing from dilute solutions to high-entropy alloys* (2026). The subregular-solution model lets the pairwise interaction parameter depend on composition; for a binary this reduces to $\Delta H^{\sigma}_{\text{mix}}(x) = M_{2[1]}\, x(1-x)^2 + M_{1[2]}\, x^2(1-x)$, where

$M_{i[j]}$ — the differential mixing enthalpy of solute i diluted in host j — is the *slope* of ΔH_{mix} at the dilute limit where i is the minority species. Each slope needs exactly **one** calculation: a single substitutional impurity of i in a supercell of pure j, read off at the one composition that single substitution actually realizes, $x = 1/N_{\mathrm{atoms}}$. The whole binary curve therefore comes from four numbers — the two pristine-supercell energies and the two single-impurity supercell energies — and an N-component alloy needs N^2 calculations total (N pristine plus $N(N-1)$ dilute impurities), the paper's own headline result against the exponential cost of Special Quasirandom Structures.

> **Note**
>
> Every energy DSIM consumes is a raw **total** supercell energy, never divided by atom count. An earlier version of this module divided first, reasoning that an intensive-looking $\Delta H_{\mathrm{mix}}(x)$ needed an intensive input; it shipped, was caught by a real MACE Au-Pt run landing $\sim 27\times$ below the working paper's own Fig. 2a, and was fixed. The composition-weighted subtraction inside $M_{i[j]}$ is itself what makes the slope an intensive, size-converged quantity — dividing again on top of that is the bug.

Validity follows the paper: single-phase substitutional solid solutions sharing one crystal structure between end members, in a supercell large enough that one impurity does not interact with its own periodic images (the paper's own 3×3×3 fcc cell, 27 atoms, is Calango's default). Unlike EGQCA and CVM above, which analyze a batch a *prior* job already computed, DSIM needs new calculations of its own: one wizard, one generated script looping over every calculation class internally, one result file — a list of $N \geq 2$ pristine reference structures in, a $\Delta H_{\mathrm{mix}}(x)$ curve (with every $M_{i[j]}$ tabulated) out.

CHAPTER 11

Remote HPC Execution

The **HPC** panel (zone 11) submits calculations to a cluster over SSH, monitors the queue, and brings the results back — without leaving Calango.

Requires

Needs `paramiko` in Calango's Python environment, and SSH access to a cluster running SLURM, PBS or SGE.

Note

All network traffic runs in a separate helper process driven over a JSON protocol, mirroring how local jobs are isolated: the interface never blocks on the network, and a hung connection can always be resolved by killing the helper rather than the application.

Note

The connection is a *session*, not a login per operation: Calango authenticates once and multiplexes every upload, submission, status poll, log tail and download over that one SSH transport. On a cluster with keyboard-interactive two-factor authentication this is the difference between usable and unusable — the one-time code is typed once, not once per queue poll.

11.1 Connection

The **Connection** tab holds the cluster credentials:

Host, Port Hostname and port (default 22).

User Your account name on the cluster.

Key file Path to a private key, e.g. `~/.ssh/id_rsa`. Leave empty to let your SSH agent or default keys authenticate.

Password The password, or the passphrase for an encrypted key.

Remote dir Working directory on the cluster, relative to `$HOME` (default `calango_jobs`).

Connect Opens the session, reports your remote `$HOME`, and auto-detects the scheduler. Pressing it first is optional: any operation issued while disconnected authenticates and then runs.

Disconnect Closes the session. A job already submitted keeps running on the cluster.

> **Note**
> There is no separate **Auth** picker: which method is used is *inferred* from **Key file** and **Password**, freshly, on every connection attempt. A key file present selects key authentication (with **Password**, if also filled, used as that key's passphrase, the same precedence plain SSH uses); a password with no key selects password authentication; both left empty still attempts the connection, with no explicit credential of its own, so the SSH agent and the default key files (`~/.ssh/id_ed25519`, `id_ecdsa`, `id_rsa`) get their usual chance.

Two-factor authentication

If the cluster asks for a second factor, Calango puts the server's own challenge on screen — its wording, its prompts, with the answer masked whenever the server says it must not be echoed — and sends what you type straight back. Clusters that ask for the password and the code in one exchange get the password filled in from the **Password** field, so only the code is asked for.

The code is never stored, which is also why a session opened with 2FA is never re-established silently: if it drops, the panel says so and waits for you to press **Connect**. A session opened with a key or a password has nothing single-use about it and reconnects by itself on the next operation.

> **Requires**
> Passwords and key passphrases are kept in memory only — never written to disk, never placed on a command line where `ps` could see them. One-time codes are not kept even that long. Everything else (host, user, key path, scheduler settings) is remembered between sessions. A host key that *changes* is refused outright rather than pinned over.

11.2 Scheduler settings

The **Scheduler** tab describes the batch job:

Scheduler **SLURM**, **PBS** or **SGE**.

Queue Partition or queue name; empty uses the cluster default.

Nodes × tasks Whole machines requested, and ranks (or cores) on each of them. The total (nodes × tasks) is what SGE is given directly, since it requests slots rather than machines.

Memory / node Per node; zero requests the cluster's own default. SLURM's `-mem` and PBS's chunk memory take this unchanged; SGE's `h_vmem` is per *slot*, so it is divided by the tasks per node first.

Walltime `HH:MM:SS`, default `01:00:00`.

Parallel env SGE only: the site's parallel-environment name (`-pe`). `smp` is single-node shared memory almost everywhere, so a multi-node SGE job needs whatever that cluster called its MPI PE.

Setup Shell lines executed before the payload — module loads, `conda activate`, anything your site needs:

```
module load python
```

```
source ~/venvs/ase/bin/activate
```

VASP POTCAR dir This cluster's PAW pseudopotential library; see §11.3 below. Only matters for VASP jobs, and only needs setting when this cluster's library lives somewhere other than wherever your own machine's copy does.

Command The payload itself, run after Setup. Defaults to running the staged script directly; type a launcher line here for anything else — `mpirun -n 4 python3 run.py`, or a site-specific one like GPAW's own `gpaw python run.py`. A second line runs *after* it, for cleanup that has to happen once the job finishes, e.g. `conda deactivate`.

The following six rows are **SLURM only** — PBS and SGE describe resources differently and have no equivalent, so the **Scheduler** tab hides them unless **Scheduler** is set to SLURM:

Account Billing/allocation account (`-account`).

QOS Quality-of-service name (`-qos`).

CPUs/task Cores per MPI rank (`-cpus-per-task`), for a hybrid MPI+OpenMP job. Distinct from the ranks-per-node above: that is how many ranks share a node, this is how many cores each rank itself gets.

GPUs/node Requested as `-gres=gpu:N`, the gres spelling that works on essentially every SLURM cluster with GPU nodes.

Node list Pin the job to specific node(s) by name (`-nodelist`). Rarely needed; leave empty unless the cluster or the job specifically requires it.

Extra #SBATCH lines Free-form, inserted into the `#SBATCH` block verbatim, after every field above — the escape hatch for a directive none of these controls covers. Include your own `#SBATCH` prefix on each line (or `##SBATCH` for one you want present but deliberately disabled, which SLURM itself already ignores, since it only ever reads a line spelled exactly `#SBATCH`).

The generated wrapper carries the right directives for your scheduler — `#SBATCH`, `#PBS` or `#$` — and always redirects output to fixed file names (`calango_job.out`, `calango_job.err`) so the monitor knows what to tail regardless of how the site templates its jobs. Every field above is saved with the cluster profile (**Connection** tab), the same as the host and credentials, so a second cluster with different modules, a different account, or a different POTCAR library needs entering only once.

11.3 POTCAR resolution

VASP's PAW datasets (POTCARs) are licensed material: Calango never bundles, generates, or transfers one. Every generated VASP script assembles its own `POTCAR` locally, through ASE, from a library that has to already exist on whichever machine the script actually runs on — and for a remote job, that is **the cluster**, not the machine that built the script.

The generated script resolves the library in this order:

1. `CALANGO_VASP_PP_PATH`, if the environment already has it — this is what the **VASP POTCAR dir** field above exports, ahead of your own **Setup** lines, in the job wrapper

submitted to *this* cluster. Set it once per cluster preset and every VASP job submitted there uses it, with no per-job typing.

2. Otherwise, the path configured in **Preferences** ⟩ **External Files** ⟩ **VASP (VASP_PP_PATH)** — baked into the script at generation time, on the machine that built it. Correct for a local run; for a remote one it only happens to work if that path also exists, unchanged, on the cluster — which is exactly what case 1 exists to avoid relying on.

3. Otherwise, whatever `VASP_PP_PATH` the shell environment already carries wherever the script runs — the same fallback ASE itself would use.

Either a flat library (`<dir>/<Element>/POTCAR`) or the nested `potpaw_PBE/potpaw/potpaw_LDA` layout is recognized automatically; a flat one is transparently shimmed with symlinks so ASE's own lookup finds it. Before ever calling into the VASP calculator, the script also checks that **every element the structure actually needs** has a `POTCAR` under the resolved library — not just that the directory exists — and fails immediately, naming the missing element(s) and the exact path searched, instead of a deep ASE traceback minutes into the queue. Calango does not auto-select PAW variants (`_sv`/`_pv`/`_d` suffixes) or switch between the PBE and LDA libraries — point the configured directory at a library that already has the variant you want for every element.

> **Note**
>
> The same check runs, as a local approximation, *before* a VASP job is staged at all: clicking **Run (Local)** or **Run (Remote)** in the calculator wizard validates the path configured in Preferences and warns immediately if it is missing or incomplete. This is necessarily a LOCAL check — it cannot inspect a cluster's filesystem over SSH, so a per-cluster **VASP POTCAR dir** override is validated only when the job actually runs, by the same in-script check described above.

11.4 Submitting and monitoring

Every calculator wizard offers **Run (Remote)** alongside **Run (Local)** on its last stage — there is no separate "remote" dialog. Configure the calculation as usual and click **Run (Remote)** instead of **Run (Local)**; Calango then runs the full sequence:

1. Stage `run.py` and `structure.extxyz` into a local job directory, exactly as a local run would.

2. Generate `job.sh` from the scheduler settings.

3. Upload the directory over SFTP, creating the remote path as needed.

4. Submit with `sbatch` or `qsub` and capture the job ID.

5. Start monitoring.

The **Queue & Logs** tab then shows the job ID and remote directory, the live queue state polled with `squeue` or `qstat` (every **10 s** by default), and the remote standard output and error streamed incrementally into the console (errors in red). PBS and SGE short codes are normalized to readable states (`R` → RUNNING, `qw` → PENDING, `H`/`hqw` → HELD), so the display reads the same whatever the cluster runs. **Cancel Job** issues `scancel` or `qdel`.

11.5 Retrieving results

When the job leaves the queue, Calango downloads the results automatically — structures, trajectories, logs and CSV metric files — into the same local job directory the inputs came from. Trajectories and final geometries open directly in a new tab.

Download Results repeats the transfer manually, which is useful for pulling intermediate output from a long-running job without waiting for it to finish.

The task appears in the process manager like any other, so its directory stays one click away for later analysis.

CHAPTER 12

Electronic Structure

Simulation ⟩ Electronic Structure... computes a band structure along a high-symmetry k-path and, where the backend supports it, an element- and orbital-resolved projected density of states.

> **Requires**
> The GPAW route needs a **baseline ground state**: a completed **Simulation ⟩ Single-point Calculation...** run with the GPAW calculator, which writes `single_point.gpw` into its job directory. The bands run loads that converged density and evaluates non-self-consistently at fixed density — it never re-runs the SCF. Without a baseline the wizard refuses to open and tells you so. Chapter 2 walks through the whole sequence on silicon.

12.1 Setting up the calculation

Baseline SCF density The completed single point whose `single_point.gpw` supplies the charge density. Inheriting it rather than re-converging is what makes the band structure attributable to a specific, inspected ground state.

Backend **Free electrons (ASE — bands only)**, **GPAW (DFT, bands + PDOS)** or **Quantum ESPRESSO (needs pw.x + pseudos)**. The GPAW entry is disabled when the package is not importable in the selected environment.

k-path Pre-filled with ASE's suggestion for the structure's Bravais lattice, e.g. `GXWKGLUWLK,UX`. Edit it freely; leave it empty to accept the suggestion.

k-points Total points along the whole path, default 80.

Valence electrons Electrons per cell, free-electron backend only.

PW cutoff Plane-wave cutoff, default 340 eV.

SCF k-grid Monkhorst-Pack n^3 grid for the self-consistent step, default 4.

Compute element/orbital PDOS (GPAW backend) Adds the projected DOS.

Environment The Conda environment or interpreter that runs the job (Section 9.3).

> **Tip**
> The free-electron backend needs nothing beyond ASE and finishes in seconds. It computes empty-lattice bands — the exact free-electron dispersion folded into your Brillouin zone — which is genuinely useful for checking that a k-path is the one you meant, and for teaching

band folding, before committing a DFT run.

Requires

Real bands need a real solver: `gpaw` installed in the selected environment, or `pw.x` plus pseudopotentials for the Quantum ESPRESSO route. The QE script is a scaffold with an `EDIT ME` line for your pseudopotential set.

The job runs like any other — console, progress markers, process-manager entry — and writes `bands.json` and, when computed, `pdos.json` into the job directory. The viewer opens automatically on completion, and can be reopened any time with **Load Result** in the process manager.

12.2 Reading the plots

The viewer shows the band structure and, when present, the PDOS side by side on a shared energy axis.

12.2.1 Band structure

Energy versus k-path distance, with vertical lines and labels at every high-symmetry point (ASE's `G` is rendered as Γ). Spin channels are drawn in different colours. A dashed horizontal line marks the reference energy at $E - E_{\mathrm{ref}} = 0$.

12.2.2 Projected density of states

Density of states versus the same energy axis, one curve per element and orbital projection — `Si s`, `Si p`, `O p` and so on — accumulated over all atoms of each species and colour-keyed to the legend.

12.3 Controls and export

Fermi level The reference energy, pre-filled from the calculation. Plots show $E - E_{\mathrm{ref}}$, so shifting it moves the zero line — useful for aligning against a reference calculation or a measured spectrum.

E min, E max The energy window, defaults −10 and +10 eV.

Projections A checklist of the PDOS curves. Uncheck the ones that clutter the picture; the vertical scale re-normalises to what remains visible.

Export Bands... CSV or `.dat`: one row per k-point, with the path distance followed by every band (and spin channel).

Export PDOS... CSV or `.dat`: one row per energy, one column per projection.

Export Fatbands... CSV or `.dat`: one row per electronic state. Present only when the run wrote orbital projections — see §12.6.4.

> **Note**
> Exported bands use the same k-path distance as the x axis, so the columns drop straight into an external plotting script and reproduce the figure you see.

12.4 Hybrid band structures with VASP

Selecting a hybrid (HSE06, HSE03, HSEsol, PBE0, B3LYP, Hartree–Fock) on the VASP backend does *not* run the fixed-density band pass every other combination uses. It cannot: the VASP wiki states that "the electronic charge density must not be fixed for any hybrid calculation, i.e., never set `ICHARG=11`!" — for a hybrid the Hamiltonian is not a functional of the density alone, so there is no converged density to diagonalise against.

Calango takes the documented alternative: **one self-consistent hybrid run** on a uniform mesh, carrying the band path in a separate `KPOINTS_OPT` file — "an optional input file to perform an additional one-shot calculation after self-consistency is reached", read automatically when present. `KPOINTS` holds the mesh; the path never enters the SCF. `HFRCUT = -1` is set, the Coulomb truncation the wiki recommends over VASP's default auxiliary functions, which "lead to discontinuities in band-structure calculations" (it converges best for gapped systems; `HFRCUT = 0` is faster for a metal).

> **Requires**
> **Requires VASP 6.3.0 or newer.** `KPOINTS_OPT` arrived in 6.3.0 and an older binary ignores the file *silently* — it would converge the hybrid and write no band path at all. The generated script checks for the result block by name and says so if it is absent.

Three things worth knowing:

Start from a converged semilocal run. Selecting a baseline stages that run's `WAVECAR` — the *orbitals*, not the density — which is what VASP recommends for a hybrid. Without one the hybrid still runs, from scratch, more slowly and with a greater chance of a different local minimum. The baseline must therefore have run with `LWAVE = .TRUE.`

The path is an explicit k-point list, not line mode. VASP returns an explicit list exactly as given — same count, same order, no symmetry folding — so the linear axis and the special-point ticks come from the same `BandPath` object the viewer reads. The script compares what came back against what it asked for and refuses if they differ, since a silent mismatch would draw energies against the wrong k axis.

The PDOS mesh is the SCF mesh. The semilocal route runs a second fixed-density pass for the projected DOS; a hybrid cannot, and a second hybrid SCF would cost as much as the first — so `LORBIT = 11` rides on the run already happening. **PDOS k-mesh** has no effect here.

The eigenvalues land in `vasprun.xml` under `<eigenvalues_kpoints_opt>` and nowhere else: VASP 6.6.1 writes no `EIGENVAL_OPT` or `DOSCAR_OPT` text file, and ASE has no reader for that element, so the generated script parses it directly. That is not on the wiki — it was established by running VASP. The test suite (`vasp_hybrid_bands`) runs the whole chain against a real binary and checks the physics rather than the plumbing: HSE06 opens Si's direct gap at Γ from 2.45 eV (PBE) to 3.56 eV, toward the experimental ∼ 3.4 eV.

12.5 Band symmetry (irreducible representations)

A band structure says where the states are. Their *symmetry* says what they are: which crossings are protected, which optical transitions are allowed, and what a degeneracy is made of. Tick **Assign irreducible representations at high-symmetry k-points** in the setup stage and the run writes `band_symmetry.json` beside `bands.json`; the viewer draws the Mulliken symbol of every band beside its high-symmetry tick.

12.5.1 What is computed

At a wave vector $\mathbf{k}$ the operations $\{R \mid \mathbf{t}\}$ of the space group that satisfy $R^{-\mathsf{T}}\mathbf{k} = \mathbf{k} + \mathbf{G}$ form the *little group*. At a generic $\mathbf{k}$ that is only the identity; at the symmetry points of the zone it can be the whole point group, which is why the classification is worth making there and nowhere else.

For each degenerate multiplet the character

$$\chi(R) \;=\; \sum_{n \in \text{multiplet}} \langle \psi_{n\mathbf{k}} \mid \{R \mid \mathbf{t}\} \mid \psi_{n\mathbf{k}} \rangle$$

is evaluated in the plane-wave representation of the Kohn–Sham states. Writing $\psi = \sum_{\mathbf{G}} c_{\mathbf{G}} e^{2\pi i(\mathbf{k}+\mathbf{G})\cdot\mathbf{x}}$, the operation sends the coefficient at $\mathbf{G}$ to $\mathbf{G}' = \mathbf{G}_0 + R^{-\mathsf{T}}\mathbf{G}$ and multiplies it by $e^{-2\pi i(\mathbf{k}+\mathbf{G}')\cdot\mathbf{t}}$ — one permutation of the coefficient array plus a phase, exact on any grid and for any operation. The characters are then reduced against the character table of the little co-group, computed numerically from its own class-sum algebra, exactly as the Raman module does for phonons (§15.8).

The origin matters and is found rather than assumed. Moving the origin to $\mathbf{x}_0$ replaces every $\mathbf{t}$ by $\mathbf{t} - (1 - R)\mathbf{x}_0$, and the labels at a zone-boundary point are only convention-free once the origin sits at the symmetry centre. spglib reports the operations in whatever cell it was handed, which is routinely not that point — ASE's `graphene()` puts an atom at the origin, where the site symmetry is $\bar{6}m2$, while the $6/mmm$ centre is the hexagon at (1/3, 2/3).

12.5.2 Settings

Also classify the symmetry lines between them Classifies a generic point of each path segment as well as its endpoints. Together these are the *compatibility relations*: the irrep at a point decomposes into the irreps of the lines running out of it, which is how a degenerate level is seen to split and which branch goes where. One extra k-point per segment.

Symmetry tolerance spglib's `symprec` for the space-group operations. Too tight and a relaxed structure loses operations it physically has; too loose and it gains ones it does not.

Degeneracy window Bands closer than this share one character. It has to be a window rather than an exact test: two states degenerate *by symmetry* still emerge from a diagonalization a few μeV apart, and splitting them yields two meaningless one-dimensional characters instead of one correct two-dimensional one.

Energy window Only bands within this distance of E_{F} are classified. Each costs a transform per operation, and the states worth naming are the ones near the gap.

> **Tip**
> Graphene along Γ–K–M–Γ reproduces the classification of Kogan and Nazarov, *Phys. Rev.*

B **85**, 115418 (2012): the two bands meeting at E_F at K form the two-dimensional E'' — that degeneracy *is* the Dirac point — with A_1' and E' below it, and A_{1g}, A_{2u} and E_{2g} among the occupied bands at Γ. This is the `graphene_band_symmetry` integration test.

Requires

At a zone-boundary point of a *nonsymmorphic* group the little-group representations are projective and no ordinary Mulliken symbol applies. The run detects this from the factor system, marks the point, and reports the raw characters rather than inventing a label. The viewer says so too.

Note

Unavailable together with spin-orbit coupling. The spinor bands are a different set of states from the scalar-relativistic ones, in a different number, and classifying them needs the double groups.

12.6 Orbital-projected bands (fatbands)

A PDOS says which orbitals contribute at a given *energy*. A fatband says which orbitals contribute to a given *band, at a given k-point* — the difference between "there is a band crossing E_F" and "the band crossing E_F is Fe *d*, so the magnetism lives there".

Tick **Compute orbital-projected bands (fatbands)** and the run carries the weight $|\langle\phi \mid \psi_{n\mathbf{k}}\rangle|^2$ of the selected orbitals alongside every band energy, writing `fatbands.json`.

12.6.1 Choosing the channels

Each row of the channel table is one projection:

Atoms An element symbol (`Fe`), a 0-based index list (`0, 2, 5-8`), or empty for every atom. Selecting individual indices is what distinguishes a surface layer from the bulk underneath it — the same element, a different question.

Orbital The shell (*s*, *p*, *d*, *f*) or one magnetic sub-level of it (p_z, d_{z^2}, …). Separating p_z from (p_x, p_y) is what separates the π bands of a layered material from its σ bands.

Label The name shown in the viewer; left blank it is generated from the selection.

Leave the table empty for one channel per element and per shell present in the structure — the useful default for a first look. **From structure** refills it with one channel per element on its valence shell.

12.6.2 Reading the plot

The viewer's **Orbital projections** panel lists the channels with their colours and offers three drawing modes:

Line width The classic fatband: the band thickens where the selected orbital contributes.

Colour scale Constant width, with the weight mapped onto the channel's colormap. Stays readable where many bands run close together.

Width + colour Both, and the default.

Several channels can be shown at once, which is the point: seeing metal d and ligand p on the same plot is how hybridization becomes visible. All channels share one normalization, so a channel contributing 2% of a state does not look as strong as one contributing 90%.

12.6.3 Colormaps and superposition

Each channel gets its own *sequential* colormap, assigned in order: Greens, Blues, Reds, Oranges, Greys, Purples — the ColorBrewer maps matplotlib ships under the same names, with one deliberate modification. **The alpha channel ramps linearly with the weight, so the lowest value is fully transparent rather than white.**

That single change is what makes overlapping projections readable. An opaque low end would paint over every channel already drawn *and* over the dispersion itself, so a six-orbital plot would show only whichever orbital happened to be drawn last. With a transparent floor each channel deposits ink only where it actually has weight, and the overlaps composite.

The plot background decides which way each ramp runs. A sequential colormap is a luminance ramp *away from the page*: on a white background it runs towards the saturated dark end, exactly as in matplotlib, and on the dark default plot background its lightness is mirrored so maximum weight is a bright hue instead of near-black. Hue and saturation are untouched, so Blues is blue either way. The channel list shows each channel's swatch, and its tooltip names the colormap — which is the name to put in a figure caption.

12.6.4 Exporting the weights

Export Fatbands... writes the weights themselves, as CSV or as a space-separated `.dat`. The table is *tidy*: one row per electronic state, not one row per k-point.

k_distance	spin	band	energy_eV	C_p_z	C_s
0.000	1	1	−20.14	0.01	0.83
0.000	1	2	−8.72	0.79	0.02

The shape is deliberate. A weight is uninterpretable without the energy of the state it belongs to, and keeping the two together in the wide layout the band export uses would take (channels × bands) columns. As written, the file drops straight into pandas or gnuplot as the (x, y, weight) triples a fatband plot consumes. Energies are absolute, matching **Export Bands...**, not shifted by the viewer's current Fermi reference; channel labels are underscored so a name like `C p_z` stays one column.

> **Note**
> GPAW backend only — the weights are the PAW projector overlaps, read through `gpaw.dos`, which the file-based DFT templates do not expose. Like the symmetry labels, unavailable together with spin-orbit coupling.

12.7 X-ray absorption spectroscopy (XAS)

Electronics ⟩ X-ray Absorption Spectroscopy (XAS)... computes a core-level absorption spectrum with GPAW, following the GPAW XAS tutorial.

An XAS transition starts in a *core* level of one atom, so it needs a PAW dataset with a hole in that level — and none ships with GPAW. The generated script therefore runs three stages rather

than one: it builds the core-hole dataset for the absorbing element, converges a ground state that uses it on the absorbing *atom*, and evaluates the spectrum from those wavefunctions. The dataset is written into the job directory and found through `setup_paths`, so nothing is installed into your GPAW distribution and the run reproduces elsewhere.

> **Note**
>
> **No prior Single-Point Calculation is needed — or usable.** XAS is launched directly on a structure, exactly like Single Point or Geometry Optimization, not chained after a completed run the way Optics, GW, Wannier Functions or Electronic Bands are: the setup wizard shows the full Calculator & Convergence Settings page itself, with no baseline-run selector, because a core-hole ground state uses a PAW dataset an ordinary ground state never used — restarting from one would restart from the wrong basis on the one atom the calculation is about. GPAW is also the *only* calculator XAS supports, so there is no per-engine difference to reconcile.

12.7.1 Absorbing site

Pick the **Element** and then the **Absorbing atom** — one atom, not all atoms of that species. Giving the core-hole setup to every one of them would model a solid in which all are excited simultaneously, which is not what an absorption measurement does. Inequivalent sites each need their own run, and the measured spectrum is their sum; the wizard says so when the structure has more than one candidate.

Edge selects the level: K (1s) is what almost every XAS measurement means, while the L edges (2s, 2p) matter for transition metals.

12.7.2 Core hole

Half hole The transition-potential approximation — 0.5 electrons removed. One calculation gives the whole spectrum, because the final-state relaxation is averaged between initial and final states. This is the tutorial's choice and what most published XAS is.

Full hole The excited final state proper. Better for the first resonance, worse for the rest of the spectrum, and it needs the delta-Kohn-Sham shift to sit on an absolute energy scale.

No hole The unperturbed ground state — what the spectrum would be if the core hole did not pull the excited states down. Worth computing to see how large that effect is.

Unoccupied bands sets how many states above the occupied ones are converged. The spectrum is a sum over them, so too few truncates it — visibly, as a spectrum that stops rather than decays.

12.7.3 Broadening and the energy scale

Broadening (FWHM) is a uniform Gaussian on every transition. **Broaden more at higher energy** ramps it linearly above the edge, which is the physical behaviour (core-hole lifetime plus final-state broadening) and what makes the result comparable with a measurement whose peaks wash out with energy. The default ramp is the tutorial's, chosen for the oxygen K edge — move it to your own.

GPAW produces the spectrum on a *relative* scale whose zero is the first unoccupied state, which is not where a measurement puts it. **Compute the absolute edge position** adds two total-energy calculations (delta-Kohn-Sham) that give the shift, roughly doubling the cost; alternatively type a known **Edge energy** directly.

> **Note**
>
> `gpaw.xas` runs on GPAW's *legacy* engine only. This is the one script Calango generates that clears `GPAW_NEW` and asks for `legacy_gpaw=True` — without it the run stops with "New-GPAW not supported".

12.7.4 Results

The viewer opens automatically when the run finishes and plots the isotropic spectrum — the average over the three Cartesian polarizations, which is what a powder or solution measurement sees. **Show the x, y and z polarizations** adds them individually; for an oriented sample the difference between them is the result. **Show individual transitions** overlays the discrete lines the broadened curve is built from, which is how you tell a genuine shoulder from two peaks the broadening merged.

The header states which energy scale the axis is on, because a relative spectrum plotted as though it were absolute is wrong by hundreds of eV and nothing about the curve reveals it.

Brought up to the same standard as the Optics viewer (Section 13.1) — the two windows draw through the identical plot widget, so styling and export work the same way in both. **Range:** sets the energy window (both boxes at *auto*, the default, fit the data). **Customize Appearance...** opens the same dialog Optics uses — background, curve colour, line width/style, axis font and colour, grid — applied live. **Broadening (FWHM)** re-derives every displayed curve from the run's own discrete transitions at a chosen width, GPAW's own way of folding them; unlike the Optics viewer's Lorentzian η, which can only be *raised*, a transition is a delta function with no width to deconvolve, so this control moves narrower just as freely as wider. It opens at the run's own FWHM and is disabled for a job directory recorded before individual transitions were saved; if the run used **Broaden more at higher energy**, a note says the uniform re-fold will not exactly reproduce the stored ramped curve. **Export CSV...** offers a choice between the broadened spectrum *as computed* and the raw unbroadened transitions; **Export Image...** renders the plot to a high-resolution PNG or JPEG.

12.8 Unfolding a relaxed supercell

Unfolding needs the supercell to be an exact integer multiple of the primitive cell: the projection is defined against that mapping. A *relaxed* doped supercell is not — substituting an atom and letting the cell relax moves its lattice constant by a per cent or so, and the pristine host is no longer $1/n$ of it.

The **Effective Bands** wizard handles that two ways.

Tolerance is how far M may sit from integers before the two cells are called incommensurate. The default (0.02) has room for a relaxation; an unrelated primitive cell misses by 0.1 or more, so the check still catches the mistake it exists for.

Force commensurability by rescaling the unit cell rebuilds the primitive cell as $P = M^{-1}S$, making the relation exact. For a relaxed supercell this is not a fudge: the supercell *is* $M\times$ something, just not $M\times$ the pristine host any more, and this recovers the host cell it actually relaxed into. Only the lattice is rebuilt — the basis rides along at the same fractional coordinates, so every atom stays on its site.

The wizard reports the resulting strain. A couple of per cent is the relaxation; much more means the wrong primitive cell was selected, and it says so.

Note

Worked example — Au_7Pt. A primitive Au fcc cell ($a = 4.078$ Å) and a $2 \times 2 \times 2$ supercell with one Au replaced by Pt, relaxed with GPAW, comes back at $a = 4.1088$ Å: a 0.755% expansion. The deduced $M = 2\mathbf{I}$ is right, but its residual is 0.015 — rejected outright by a pristine-cell tolerance, accepted by the default one. Forcing rebuilds the primitive at 4.1088 Å and the residual falls to 5×10^{-17}.

12.9 Local Density of States (LDOS)

Electronics ⟩ Local Density of States (LDOS)... sums $\mathrm{LDOS}(\mathbf{r}) = \sum_{n\mathbf{k}s \in [E_{\min}, E_{\max}]} w_{\mathbf{k}} \, |\psi_{n\mathbf{k}s}(\mathbf{r})|^2$ over every stored Kohn-Sham state whose eigenvalue falls in a chosen window, weighted by k-point — **not** by occupation, so an "unoccupied near E_F" window is a real, nonzero field. The result is a real-space grid through the ordinary volumetric pipeline (Section 15.10).

Requires

A baseline is **mandatory** on either engine — LDOS is a post-process, with no fresh-SCF fallback. GPAW needs an already-completed `single_point.gpw` written with `mode='all'`; VASP needs a `WAVECAR`, which means the parent ran with `LWAVE = .TRUE.` (the **Write:** row of the VASP calculator settings). A parent without one never appears in the list.

On a **VASP** parent the same menu entry does something structurally different: instead of summing, it hands the selection to VASP's own `LPARD` post-process — "a postprocessing step that requires a pre-converged calculation" in which "no electronic (or ionic) minimization is performed". `NBMOD` chooses the selection mode (-3 adds E_F to the bounds, -2 reads them as absolute) and `EINT` carries the window; the result is a `PARCHG`, converted to the same `ldos.cube` the GPAW route writes. `ICHARG` is deliberately not set, and the parent's `KPOINTS` is copied over verbatim rather than regenerated — for an even mesh a Monkhorst-Pack and a Γ-centred grid of the same size are different meshes, and the orbitals belong to one of them.

Requires

On the VASP route the **energy window is not adjustable afterwards**. The GPAW route keeps every selected state's $|\psi(\mathbf{r})|^2$, so the viewer re-sums a new window for free; VASP recomputes from the `WAVECAR` each time, so a new window is a new job. Two smaller differences: there is no spin-channel selection (no INCAR tag selects one), and a noncollinear parent is refused outright — "LPARD is not supported for noncollinear calculations".

Selecting a baseline reads back its eigenvalue spectrum and shows it as a draggable dual-handle histogram. GPAW needs a real, SCF-free restart for that; VASP needs nothing executed at all, since `EIGENVAL` already holds the eigenvalues, occupations and k-point weights and `DOSCAR`'s sixth line holds E_F. Either way: drag either edge of the shaded band to set the window, or type it directly into **Min**/**Max**. **Relative to Fermi level** keeps the two bounds as offsets from E_F (correct even if the baseline is re-run with a slightly different Fermi level); **Occupied near E_F**/**Unoccupied near E_F** are one-click presets of **Preset width**. **Spin channel** selects **Sum**, **Spin up** or **Spin down**.

Unlike Wannierization, LDOS carries **no "Symmetry: off" pre-condition** — it sums over whatever k-points the baseline stored, weighted by GPAW's own (already symmetry-aware) k-point weights. The job writes `ldos.cube` and `ldos.json`, registered in the Volumetric Data dock **unchecked**, the same "add it, let the user pick" default Wannier orbitals use. With

`LSEPB`/`LSEPK` on, the VASP route writes one `PARCHG` per band and/or k-point and *every* one is converted and registered separately, labelled by its indices.

12.10 Wavefunctions

Electronics ⟩ Wavefunctions... computes individual Kohn-Sham orbitals on a real-space grid — pseudo, or the all-electron PAW reconstruction — as volumetric data. It shares its restart and wavefunction-access layer with LDOS: one implementation of "restart a GPAW baseline and read back a state," not two.

Selecting a baseline fills a table of every stored state (band, k-point, spin, energy, occupation); tick as many rows as needed and each becomes one cube in a single job. **Quantity** offers $|\psi|^2$ (density), the real part, or the imaginary part — there is no separate "signed" option, since the real part of a real orbital already carries both signs and the volumetric pipeline already gives any field with negative values the positive/negative dual-isosurface treatment. **All-electron (PAW reconstruction)** switches from the smooth pseudo-wavefunction to the true, cusped orbital near each nucleus; it needs the *new* GPAW engine and runs on its own **grid spacing**, independent of the SCF's native grid.

> **Note**
> By construction the PAW correction is exactly zero outside each atom's augmentation sphere: pseudo and all-electron agree away from every nucleus and differ only close to one. On water's HOMO this is verified directly (`wavefunctions_h2o`): the density difference between the two reconstructions, integrated near the nuclei, comes out more than 10× the same quantity integrated away from every atom.

Cubes are named `psi_n<band>_k<kpt>_spin-<up|down>_<quantity>.cube` and registered unchecked, like LDOS. A periodic parent's Bloch orbital gets the same periodic-continuation treatment as a Wannier orbital (its isosurface continues across the cell boundary rather than being cut at the faces); a molecular parent's orbital already sits entirely inside its vacuum padding, where that would be meaningless, and is left off.

12.11 Energy Diagrams

Electronics ⟩ Energy Diagrams... is the band-structure readout at the top of this chapter, for a system with no k-path to disperse along: discrete Kohn-Sham levels for a molecule, plus, optionally, the electric-dipole transition moments between them.

> **Requires**
> **Non-periodic (or Gamma-only) only.** A baseline that stored more than one k-point is refused — in the wizard once its spectrum is read back, and unconditionally by the generated script itself.

> **Requires**
> These are **Kohn-Sham eigenvalue-difference transitions** — $E_j - E_i$ from a single ground-state SCF — **not TDDFT or BSE excitation energies**. There is no excited-state density relaxation and no electron-hole binding.

The level diagram draws one bar per state, solid when occupied and dashed when virtual; degenerate levels (within a tolerance, since a real-space grid does not exactly respect a molecule's point-group rotations) are drawn as several parallel bars with a "$\times N$" count rather than one bar picked arbitrarily. HOMO and LUMO are labelled with the gap annotated; spin-polarized baselines get two columns.

Transition dipole moments come from GPAW's own `gpaw.utilities.dipole.dipole_matrix_elements_from_` and the oscillator strength follows the standard atomic-units form $f_{ij} = \frac{2}{3}\Delta E_{ij}|\langle\psi_i|\mathbf{r}|\psi_j\rangle|^2$. A transition is **allowed** when f_{ij} exceeds a threshold and **forbidden** otherwise — a numeric verdict on the computed matrix element, **not** a symmetry label: point-group irrep labeling and symmetry-derived selection rules were investigated and deliberately deferred (see `FUTURE.md` §8 in the source repository for why). **Export Transitions...** writes a CSV, one row per transition.

CHAPTER 13

Optical and Quasiparticle Properties

Four workflows evaluate the response of a converged ground state: the dielectric function, the same response reduced to sheet quantities for a 2D material, the second-order (nonlinear) response, and the G_0W_0 quasiparticle correction.

Requires

The first two and G_0W_0 **inherit a baseline** rather than converging their own ground state — a completed GPAW **Simulation** ⟩ **Single-point Calculation…**, which writes `single_point.gpw`. They will not open without one.

This is not merely a saving. Re-converging inside the job would give a spectrum from a different SCF solution than the one you validated — different smearing, a different k-grid, possibly a different magnetic state — with nothing in the output to say so. Because the ground state is inherited whole, these wizards show no Calculator Settings stage: the cutoff, functional and k-grid all come back from the baseline, and offering to set them again would present knobs that cannot affect the run.

Nonlinear Optics (§13.3) is the exception, and for a reason worth knowing: `gpaw.nlopt` requires point-group symmetry to be off and its band sums need a *converged* empty manifold, neither of which a general baseline has. It therefore converges its own ground state and keeps a Calculator Settings stage.

13.1 Optical properties

Simulation ⟩ **Optics…** evaluates the frequency-dependent dielectric function $\varepsilon(\omega)$ with GPAW's linear-response module and derives the usual optical observables from it.

13.1.1 Settings

Baseline SCF (.gpw) The ground state to inherit. The note beneath it spells out what is being inherited — functional, cutoff, k-grid and environment — because with no Calculator Settings stage this is the only place those values are visible, and they are what decides whether the spectrum is trustworthy.

Broadening η Lorentzian broadening, default 0.1 eV. Smaller values resolve sharper features but need a denser k-mesh in the *baseline* to be meaningful.

Energy window min/max The photon-energy range, default 0–20 eV.

Tetrahedron integration See below.

Number of points Samples on the energy grid.

Polarization directions ε_{xx}, ε_{yy}, ε_{zz}. For an isotropic crystal the three coincide.

13.1.2 Brillouin-zone integration

By default the response is summed over k-points, giving every interband transition a Lorentzian of width η. Anything narrower than η — a van Hove singularity, a 2D absorption edge — is smeared into the broadening rather than resolved.

Tetrahedron integration interpolates the bands linearly within each tetrahedron and integrates analytically, resolving those features on a mesh where point integration is still noisy.

> **Requires**
> Tetrahedron integration requires the *baseline's* k-grid to contain every vertex of the irreducible Brillouin zone — a grid from `gpaw.bztools.find_high_symmetry_monkhorst_pack()`. An ordinary Monkhorst–Pack grid usually does not qualify, and since the grid is inherited, it has to be right when the *baseline* is run; the optics job cannot fix it afterwards. When the grid is unsuitable the run stops and says so, naming the remedy, rather than quietly falling back to point integration and returning a spectrum computed by a different method than the one requested. Which integrator ran is recorded in `optics.json` alongside the spectra.

13.1.3 The viewer

The results window plots one quantity at a time against photon energy: ε_1 and ε_2 together, the absorption coefficient $\alpha(\omega)$, reflectivity $R(\omega)$, the refractive index n and k, and the energy-loss function $L(\omega)$.

The **X axis** selector switches between **Energy (eV)** and **Wavelength (nm)** via $\lambda = hc/E$ with $hc = 1239.84197\,\text{eV nm}$. Every curve and every tick label follows.

> **Note**
> Samples at $E \leq 0$ have no finite wavelength and are omitted from the wavelength view — dropped from the abscissa and from every curve together, so nothing is shifted out of alignment. Since GPAW's energy grid starts at zero, expect the wavelength view to begin one sample in. Very low energies map to very long wavelengths, so a window that starts near 0 eV will stretch far into the infrared; set **Energy window min** above zero if that compresses the part of the spectrum you care about.

Export CSV... writes one row per sample with every quantity for the selected direction; **Export Image...** renders the current plot at high resolution.

13.2 2D optics

Modules ⟩ 2D Materials ⟩ 2D Optics... runs the same response and then divides the vacuum back out.

The reason is that a slab supercell's ε_{3D} is not a property of the material: double the vacuum padding and it moves, because the sheet's polarization is diluted over more cell. The wizard adds one question — **Vacuum axis**, seeded from the cell but worth confirming, since getting it wrong rescales every 2D quantity by the wrong length — and derives three quantities that do not depend on the padding:

Quantity	Definition	Units
Sheet polarizability	$\alpha_{2D} = \frac{L_z}{4\pi}(\varepsilon_{3D} - 1)$	Å
Absorbance	$A(\omega) = \frac{\omega L_z}{c}\,\mathrm{Im}[\varepsilon_{3D}]$	dimensionless
Sheet conductivity	$\sigma_{2D} = -i\,\omega\,\alpha_{2D}$	e^2/h

The -1 in α_{2D} removes the vacuum's own contribution, which is what makes the result a property of the sheet.

The viewer offers these three only when the file actually carries them, and a 2D job opens on the absorbance rather than on ε.

> **Tip**
> Graphene is the reference case. Its low-energy absorbance is the universal $A = \pi\alpha \approx 2.29\,\%$, fixed by the fine-structure constant alone, and its conductivity is the equally universal $\pi/2$ in units of e^2/h. If a graphene run reproduces those, the whole chain — baseline, response, vacuum-axis choice and unit conversion — is working. Reaching the plateau needs a dense mesh: it is the k-sampling near the Dirac point that decides this, not the cutoff or the band count.

13.3 Nonlinear optics

Electronics ⟩ Nonlinear Optics… evaluates the *second-order* response with GPAW's `gpaw.nlopt` module, within the independent-particle approximation. Three quantities are offered:

Second-harmonic generation $\chi^{(2)}(-2\omega;\omega,\omega)$: two photons of energy $\hbar\omega$ in, one of $2\hbar\omega$ out. Reported in pm/V, and for a sheet additionally as $\chi^{(2)}L$ in nm²/V.

Shift current $\sigma^{(2)}(0;\omega,-\omega)$, in A/V²: the DC photocurrent a homogeneous illuminated crystal carries with no junction and no built-in field — the bulk photovoltaic effect.

Linear susceptibility tensor $\chi^{(1)}(\omega)$, the full 3×3 tensor from the same matrix elements, and the $\varepsilon = 1 + \chi^{(1)}$ that follows from it.

> **Requires**
> $\chi^{(2)}$ and the shift current are **odd-rank tensors**: in a centrosymmetric crystal every component vanishes *identically*, by symmetry rather than by smallness. GPAW will still return numbers — the residue of an incomplete cancellation over a finite k-mesh — and they look like a spectrum. The generated script therefore tests the cell for an inversion centre *before* converging anything and warns, and the viewer repeats the warning above the plot. This is the single most common way the module is misused.

13.3.1 Why this one converges its own ground state

Unlike every other workflow in this chapter, Nonlinear Optics does **not** inherit a Single-Point baseline. Three requirements make an ordinary baseline unusable:

- `make_nlodata` asserts that *point-group symmetry is off*. The matrix elements it builds are not invariant under the point-group folding of the k-set, and the assertion arrives with no message — after the SCF has already been paid for.

- The sums run over *intermediate* states, so the band set has to be large. The wizard's ground state uses `nbands="nao"`.
- Those empty bands must be *converged*. An SCF converges occupied states; the unoccupied manifold it leaves behind is noise, and here it is summed over. The generated calculator asks for `convergence={"bands": -10}`.

All three are imposed by the generator rather than left on the Calculator Settings stage, because they are requirements of the method and not preferences. Time reversal is deliberately *kept*: the weaker `symmetry="off"` would satisfy the assertion too, at roughly double the k-points, for nothing.

13.3.2 Settings

Tensor components Three letters from `xyz`, several separated by commas (`yyy, xxy`). There is deliberately no "all 27": most vanish by symmetry, each costs a full sum over bands and k-points, and 27 spectra of which 24 are numerical noise is not a better answer. **Analysis** ⟩ **Symmetry** will name the point group that decides which are allowed.

SHG gauge **Length** (GPAW's default) and **Velocity** are formally equivalent and numerically are not: the velocity gauge carries low-frequency divergences that cancel only for a complete band set, so a truncated sum leaves it visibly wrong as $\omega \to 0$. Running both and overlaying them is the standard convergence test — their disagreement says more about the band summation than any single number does.

Broadening η A second-order spectrum is far more sensitive to it than a linear one: the resonances are sharper and the divergences at ω and 2ω are regularized by exactly this number, so a small η on a coarse mesh produces spikes rather than structure.

Scissors shift A rigid shift of the empty bands, for the semilocal gap error. $\chi^{(2)}$ is more sensitive to the gap than the linear response is, because the two-photon resonance sits at *half* of it. The applied value is recorded in the results and restated by the viewer, so a spectrum never silently carries one.

Vacuum axis As in 2D optics: a supercell $\chi^{(2)}$ is diluted by whatever vacuum was used, and multiplying the thickness back in gives the sheet value the 2D literature quotes.

> **Note**
> The expensive step is `make_nlodata` — the momentum matrix elements — and it runs *once*. Every component and every response reuses the same data, which is saved as `mml.npz` alongside the run. Asking for a second tensor component costs a band sum, not another ground state.

> **Tip**
> Remember that SHG probes the band structure at 2ω as well as at ω. A window ending at 6 eV is sampling transitions up to 12 eV, and the band count has to reach them — which is also why the $\chi^{(1)}$ panel is worth computing: the $\chi^{(2)}$ features of a semiconductor sit at the absorption edge *and* at half of it, and a resonance assigned to the wrong one of those is the standard way to misread an SHG spectrum.

13.4 GW calculations

Simulation ⟩ GW Calculations... computes one-shot G_0W_0 quasiparticle corrections,

$$E^{\mathrm{qp}}_{n\mathbf{k}} = \varepsilon^{\mathrm{DFT}}_{n\mathbf{k}} + Z_{n\mathbf{k}}\left[\Sigma_{n\mathbf{k}}(\varepsilon^{\mathrm{DFT}}) - V^{xc}_{n\mathbf{k}}\right],$$

i.e. the DFT eigenvalue with its exchange–correlation potential replaced by the energy-dependent, non-local self-energy. Nothing here is self-consistent, and the result is only meaningful relative to the baseline it corrects.

13.4.1 Engines

Two engines, each bound to the DFT code that produced its baseline. They are not interchangeable — G_0W_0 perturbs a specific DFT solution, so the engine follows from which baseline exists rather than from preference.

GPAW Corrects a `.gpw`. The run restarts the baseline, adds a generous set of empty bands at fixed density, then applies `gpaw.response.g0w0.G0W0`.

Yambo Corrects a Quantum ESPRESSO `.save` directory. The run converts it with `p2y`, generates the G_0W_0 input, executes it under MPI, and parses the quasiparticle report.

13.4.2 Settings

Screening cutoff The convergence parameter that matters. A G_0W_0 gap is not converged until it stops moving with this.

Frequency treatment **Plasmon-pole** approximates the frequency dependence of the screening with a single pole — much cheaper, and usually good for the gap of a simple semiconductor. **Real axis** performs the full integration.

Bands below/above How many occupied and empty states around the gap receive a correction.

MPI ranks Yambo only.

13.4.3 The viewer

The **GW Viewer** — also on **Results ⟩ GW Viewer...** — reports the DFT and quasiparticle band edges, the gap renormalization as its headline number, and a per-state table of $\varepsilon_{\mathrm{DFT}}$, E_{qp} and the correction between them.

> **Note**
> G_0W_0 essentially always *opens* a semiconductor gap, typically by 0.5–2 eV. A renormalization that is negative or near zero is far more likely an unconverged screening cutoff or too few empty bands than a physical result, and the viewer says so rather than presenting the number neutrally. The per-state corrections are the other thing to read: they should vary smoothly across bands, not jump erratically between neighbouring states.

CHAPTER 14

Boltzmann Transport and Berry Phase

Two modules on **Wannier Functions** that consume a Wannier Hamiltonian $H(\mathbf{R})$ and compute from it directly, in process — no job is launched and no interpreter is started.

> **Note**
> Both are native Calango implementations. Wannier90, postw90 and BoltzWann are references for the formulas, conventions and feature scope; they are not dependencies. No external binary is invoked, nothing is linked against them, and none of their auxiliary output files is required. Reading an `_hr.dat` you already have is done by Calango's own parser — the file is a plain text table of hopping integrals.

Both take their input from the same row, which offers three sources. The normal one is *From a completed run*: a Wannier Functions run finished in this session, whose `wannier_hr.dat`, cell, centres and spreads are read straight from its directory, so nothing outside Calango is involved. The others are opening an `_hr.dat` you already have, and loading one of two built-in demonstration models (a simple-cubic metal, and the two-band Qi–Wu–Zhang Chern insulator). Those are the models the unit tests use, so what the panel reports can be checked against the test output directly.

A run started before Calango wrote $H(\mathbf{R})$ says so when picked, and says to re-run the Wannierization on the same baseline — distinguished from a run that recorded the file but no longer has it, which needs a different fix.

14.1 Boltzmann Transport

Wannier Functions ⟩ Boltzmann Transport…

Electronic and thermoelectric transport in the constant relaxation-time approximation. Everything derives from the transport distribution function

$$\Sigma_{\alpha\beta}(\varepsilon) = \frac{1}{V N_k} \sum_{n,\mathbf{k}} \tau \, v_{n\alpha} v_{n\beta} \, \delta(\varepsilon - \varepsilon_{n\mathbf{k}}),$$

whose Onsager moments $L^{(m)} = \int d\varepsilon \, \Sigma \, (\varepsilon - \mu)^m (-\partial f / \partial \varepsilon)$ give the conductivity $\sigma = e^2 L^{(0)}$, the Seebeck tensor $S = (eT)^{-1} (L^{(0)})^{-1} L^{(1)}$, and the electronic thermal conductivity

$$\kappa_e = \frac{1}{e^2 T} \left[L^{(2)} - L^{(1)} (L^{(0)})^{-1} L^{(1)} \right].$$

That is the *zero-current* thermal conductivity, which is the one zT wants. The subtracted term is the Peltier heat carried by the compensating current; dropping it inflates κ_e and deflates zT.

The panel also reports the power factor $S^2\sigma$, the figure of merit $zT = S^2\sigma T/(\kappa_e + \kappa_L)$, and the carrier concentration.

14.1.1 Band velocities

Velocities come from $\partial H/\partial \mathbf{k}$ in the Wannier gauge, taken analytically. This is not an optimisation: at a band crossing the eigenvalue branches swap, so a finite difference of *sorted* eigenvalues reports a velocity that jumps discontinuously and can have the wrong sign entirely. The matrix gradient has no such problem, because the degeneracy lives in the eigenvectors.

14.1.2 The relaxation time

τ enters σ and κ_e linearly and cancels *exactly* in S, which is a ratio of two moments that each carry one factor of it. A constant-τ Seebeck coefficient is therefore a genuine prediction, while σ and κ_e are only as good as the τ supplied. The lattice thermal conductivity κ_L is a phonon quantity that nothing in an electronic structure determines, so it is a user input and is needed only for zT.

Requires

The smearing that resolves the delta function has to sit between two bounds: *above* the mean level spacing of the k-mesh, or $\Sigma(\varepsilon)$ breaks into isolated spikes, and *below* k_BT, or the smearing acts as extra temperature and the Wiedemann–Franz ratio comes out low. At 300 K, k_BT is 26 meV. The panel reports the measured level spacing against both after every run.

14.2 Berry Phase

Wannier Functions ⟩ Berry Phase…

Quantity	How it is obtained
Berry phase on a closed path	Wilson loop: product of overlap matrices
Berry curvature $\Omega(\mathbf{k})$	Kubo sum over band pairs of $\partial H/\partial k$ elements
Curvature map	the same over a k-plane, drawn with the app's colormaps
Anomalous Hall conductivity	BZ integral of Ω, in SI and in e^2/h
Electric polarization	Berry phase (modern theory), with its quantum
Hybrid Wannier centres	Wilson-loop eigenphases vs transverse k

14.2.1 Gauge

This is the whole difficulty of the subject, so the conventions are fixed explicitly. Berry phases are Wilson loops: an arbitrary phase on an eigenvector enters once as a bra and once as a ket and cancels around the loop, so the result does not depend on what the eigensolver returned. Berry curvature uses the Kubo form, in which every factor is a matrix element between eigenstates, so the phases cancel pairwise and no eigenvector is ever differentiated. $H(\mathbf{k})$ is built in convention I — orbital positions *not* in the phase — so $H(\mathbf{k}+\mathbf{G}) = H(\mathbf{k})$ exactly and a loop closes with no correction factor.

14.2.2 Relation to Topological Invariants

The hybrid Wannier centre flow is the same object **Wannier Functions ⟩ Topological Invariants…** plots. That feature generates a Python script and runs it against a completed Wannierization;

this one evaluates the same quantity in process from $H(\mathbf{R})$. The two now overlap more than they used to: on a **GPAW**-sourced run the generated script reopens the `.gpw` and transports the Bloch states with `gpaw.berryphase`, but on a **VASP**-sourced one it builds a Wilson loop from the same `wannier_hr.dat` this panel reads — so on that route they are the same algorithm reached two ways, and on the GPAW route they remain genuinely independent. Either way this panel reports the Chern number twice, once from the curvature integral and once from the centre winding, so that a disagreement flags under-sampled k.

One thing the $H(\mathbf{R})$ route cannot do, on either side: a wannier90 `_hr.dat` is an $N_w \times N_w$ matrix with *no spin labelling*. For a spinor (`LSORBIT`) run its Wannier functions are spinors and N_w counts them, but the file records neither the up/down decomposition nor the convention used to order it. Neither invariant reads a spin expectation — the Chern winding and the Soluyanov–Vanderbilt largest-gap Z_2 are computed from the centres alone — so this costs a plot series, not a result, and `topology.json` carries `spin: null` rather than an invented one.

14.3 Validation

Both modules are covered by unit tests against results known independently of the code: the Wiedemann–Franz limit $\kappa_e/\sigma T \to \pi^2 k_B^2/3e^2 = 2.44 \times 10^{-8}\,\mathrm{W\Omega/K^2}$, the Seebeck sign for electron and hole doping, the exact τ-scaling identities, the Qi–Wu–Zhang Chern number (± 1 inside $|m| < 2$, zero outside), the Hall conductance quantised at e^2/h, and the SSH Zak phase, which comes out exactly 0 and π in the two phases.

CHAPTER 15

The Analysis Toolbox

Every tool in this chapter opens from the **Analysis** menu. Most compute as soon as they open, run on a worker thread so the interface stays responsive, and export their data as CSV or whitespace-separated `.dat` — the format is chosen by the extension you type.

15.1 Radial distribution function

Analysis ⟩ Radial Distribution Function computes $g(r)$, the probability of finding an atom at distance r relative to a uniform density.

Element pair Two combos, each **Any** or a specific element. **Any**–**Any** gives the total RDF; a specific pair gives the partial.

r max Default 10 Å.

Bins Default 200.

Periodic boundary conditions Enabled and checked when the structure has a cell.

Frames For trajectories: **start**, **end** and **stride**, giving a frame-averaged $g(r)$ over any sub-range.

> **Note**
> With periodicity enabled, all periodic images within $r_{\max}$ are enumerated explicitly rather than relying on the minimum-image convention. The result is therefore valid beyond $L/2$ and correct for triclinic cells — both places where a naive implementation quietly produces artefacts.

15.2 Bond length and angle distributions

Analysis ⟩ Bond Length / Angle Distributions histograms either pair distances or three-body angles j–i–k within a cutoff.

Distribution **Bond length distribution** or **Bond angle distribution (j–i–k)**.

Species (pair / center–neighbor) Two element filters. For angles the first is the central atom.

Cutoff Default 3 Å.

Bins Default 90.

Angles are in degrees, lengths in Å; both handle periodic images exactly.

15.3 Coordination numbers (CN / GCN)

Analysis ⟩ Coordination Numbers (CN / GCN) computes per-atom coordination and generalised coordination numbers.

Neighbor cutoff **Covalent radii × tolerance** (default tolerance 1.15) or **Fixed cutoff radius** (default 3 Å).

Bulk CN reference The cn_{max} normalising the GCN — 12 for fcc/hcp, 8 for bcc, 4 for diamond, or **Auto (max CN found)**.

Results appear as a per-atom table (index, element, CN, GCN) with a summary line giving the ranges and means. Two buttons push the result onto the structure: **Color Viewport by CN** and **Color Viewport by GCN**.

> **Note**
> The generalised coordination number weights each neighbour by its own coordination, which distinguishes sites that plain CN cannot: a terrace atom, a step-edge atom and a vertex atom on a nanoparticle can share the same CN but have clearly different GCN. This is what makes GCN such a good descriptor for catalytic activity.
> Periodic images are enumerated explicitly, so a primitive fcc cell correctly reports CN = 12 even though all twelve neighbours are images of the single atom in the cell.

15.4 Static structure factor

Analysis ⟩ Structure Factor S(q) computes $S(q)$ from the (frame-averaged) pair distribution via a Lorch-windowed Fourier transform.

q range Defaults 0.3 to 12 $Å^{-1}$.

q points Default 400.

g(r) r max Upper bound of the Fourier integral, default 10 Å. Larger values improve low-q fidelity.

g(r) bins Default 400.

Frames Start/end/stride, as for the RDF.

15.5 X-ray diffraction

Analysis ⟩ X-Ray Diffraction (XRD) simulates a powder pattern with the Debye scattering equation.

Wavelength Presets for Cu Kα (1.54056 Å), Co Kα, Mo Kα, Cr Kα, or **Custom** to type your

own.

2θ range Defaults 10–90°.

Points Default 800.

Supercell repeat Periodic structures are repeated before the Debye sum, default 3. More repeats sharpen the Bragg peaks, but the cost grows as N^2 in atoms.

Unlike the other analysis dialogs, this one waits for you to press **Simulate**. The status line then reports which form factors were used: Waasmaier-Kirfel factors when every species is tabulated, otherwise a constant $f = Z$ approximation — in which case peak *positions* remain exact but high-angle *intensities* are approximate.

Two exports: **Export Curve…** writes the 2θ–intensity pattern, and **Export Peaks…** writes detected peaks (those above 2 % of the maximum) with their d-spacings from $d = \lambda/(2\sin\theta)$.

15.6 Warren-Cowley chemical order

Analysis ⟩ Warren-Cowley analysis quantifies chemical short-range order in multicomponent alloys through

$$\alpha_{ij} = 1 - \frac{p_{ij}}{c_j},$$

where p_{ij} is the probability that a neighbour of an i-type atom is of type j, and c_j is the overall concentration of j.

$\alpha = 0$ The ideal random alloy.

$\alpha < 0$ Ordering — unlike pairs preferred.

$\alpha > 0$ Clustering or segregation of like species.

Set **First shell cutoff** (default 3.2 Å) and, optionally, **Second shell cutoff** (default 4.8 Å). The dialog reports the concentrations, the mean coordination of each shell, and the full α_{ij} matrix for every ordered species pair, exportable as CSV.

> **Tip**
> A quick sanity check: a perfectly ordered B2 (CsCl-type) binary gives $\alpha_{AB} = -1$ in the first shell and $+1$ in the second, while a random decoration of the same lattice gives values near zero.

15.7 Local entropy

Analysis ⟩ Local Entropy Analysis computes the per-atom pair-entropy fingerprint of Piaggi and Parrinello, in units of k_B:

$$s_S^i = -2\pi\rho \int_0^{r_c} \left[\, g_i(r)\ln g_i(r) - g_i(r) + 1 \,\right] r^2 \, \mathrm{d}r,$$

with $g_i(r)$ the Gaussian-smoothed radial distribution around atom i.

Cutoff Integration cutoff, default 5 Å — three or four coordination shells works well.

Broadening σ Gaussian width, default 0.15 Å.

Radial grid points Default 100.

Average over neighbors Smooths s_i over each atom's neighbourhood, sharpening the contrast between ordered and disordered regions.

The result is stored as the per-atom field `local_entropy` and mapped onto the atoms immediately, with a distribution histogram in the dialog. *Lower* (more negative) values mark more ordered, crystal-like environments; higher values mark disordered, liquid-like ones. This makes it an effective order parameter for melting, solid-liquid interfaces and grain boundaries.

15.8 Raman modes

Analysis ⟩ Raman Modes performs a Γ-point factor-group analysis: which optical phonons are Raman-active, IR-active or silent, by symmetry alone.

The method uses no hard-coded character tables. spglib supplies the space group operations of the primitive cell; the character table of the isogonal point group is computed numerically from the class-sum algebra; the mechanical representation $\chi(R) = (\pm 1 + 2\cos\theta)\, N_{\text{unmoved}}(R)$ is reduced into irreducible representations; the acoustic translations are subtracted; and activity follows from the vector (IR) and symmetric polarizability (Raman) representations.

The dialog reports the space group, point group, number of atoms in the primitive cell, and a table of irreps with their degeneracy, optical mode count and activity.

> **Tip**
> For silicon in the diamond structure the result is the textbook $\Gamma = T_{2g} + T_{1u}$: one triply degenerate Raman-active optical mode, and the acoustic branch.

> **Note**
> Mulliken subscripts are assigned from Cartesian axis geometry. In low-symmetry orthorhombic groups the subscript convention can be permuted relative to a particular textbook's axis choice — the activities and degeneracies are unaffected.

> **Requires**
> Needs `spglib` and a periodic structure.

15.9 Magnetic space group

Analysis ⟩ Magnetic Space Group determines which of the 1651 magnetic space groups (MSGs) the structure realizes, from its coordinates *together with* its magnetic moments.

An ordinary space group asks only where the atoms are. Once the atoms carry moments, an operation that maps a moment onto its own reverse is no longer a symmetry — unless it is combined with *time reversal* T. Writing the full group as $\mathcal{M} = \mathcal{G} + \mathcal{A}$, with $\mathcal{G}$ the unitary

operations and $\mathcal{A}$ the antiunitary ones, the Belov–Neronova–Smirnova (BNS) classification has four types:

Type I $\mathcal{A}$ is empty. Every symmetry is unitary; time reversal is broken outright. 230 of them, one per space group.

Type II $\mathcal{A} = T\mathcal{G}$, the "grey" groups. T alone is a symmetry, which is only possible when every moment vanishes: these are the *non-magnetic* structures. 230 of them.

Type III $\mathcal{A} = Tg_0\mathcal{G}$ with g_0 a spatial operation that is *not* a pure translation. Half the parent group stays unitary; the rest survives only combined with T. 674 of them.

Type IV g_0 *is* a pure translation — an *anti-translation*. The magnetic unit cell is a supercell of the crystallographic one: the classic two-sublattice antiferromagnet. 517 of them.

The classification follows Watanabe, Po and Vishwanath, *Sci. Adv.* **4**, eaat8685 (2018).

15.9.1 Supplying the moments

Magnetic moments from loads whichever moments the structure carries — the converged `magmoms` of a finished calculation, or the `initial_magmoms` it was seeded with. Those answer different questions: the guess says what was *asked* for, the result says what the calculation *found*, and an ordering that collapsed during the SCF is exactly the case where they disagree.

Whichever is loaded, the moment table stays editable. Flip a sublattice, cant a moment out of the axis, or type in an ordering the file never carried, then press **Determine**. Asking "what would the magnetic space group be if this sublattice flipped?" is the normal way to use this module.

A collinear structure (all moments along z) is analysed as collinear; any transverse component switches to the non-collinear treatment, in which the moments are axial vectors that rotate with the operations.

15.9.2 Reading the result

BNS number The label $S.L$, e.g. `221.97`: S is one of the 230 ordinary space groups and L distinguishes its magnetic descendants. This is the identifier the tables are keyed by.

Type I–IV, with a one-line statement of what it means for the crystal, and — for type IV — the anti-translation itself.

Parent space group The S of the BNS label: the space group of the *unitary* subgroup.

Crystallographic space group What the same structure would be with the moments *ignored*. The comparison is the physics: magnetic order can only lower the symmetry, and the difference between these two is exactly what the magnetism broke.

Magnetic ordering Non-magnetic, ferromagnetic, ferrimagnetic or antiferromagnetic, read off $|\sum_i \mathbf{m}_i|$ against $\sum_i |\mathbf{m}_i|$, both reported.

Magnetic class In the moment table: the symmetry-equivalence class of each atom *under the magnetic group*. Two atoms of the same element in different classes are the two sublattices of an antiferromagnet.

The **Symmetry operations** tab lists every element of the group. Those marked $1'$ are the antiunitary ones — the spatial operation alone is not a symmetry, only its combination with time reversal is.

15.9.3 Tolerances

The positional tolerance is spglib's usual `symprec`. The moment tolerance is separate and in μ_B, because a moment converged to $1.98\,\mu_B$ against its neighbour's $-2.02\,\mu_B$ is one antiferromagnet, not two inequivalent sublattices, and only a tolerance in μ_B can say so. Raise it to absorb SCF noise; lower it to resolve a genuine ferrimagnetic inequivalence.

> **Tip**
> The same cell, three answers. Take Fe at $(0,0,0)$ and $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$ in a cube — geometrically bcc, $Im\bar{3}m$ (229). Zero moments give the grey type II group. Parallel moments give type I, still with parent 229. Antiparallel moments give `221.97`, type IV: the body-centring translation now maps a moment onto its reverse, so it survives only with T, and the unitary group drops to the simple-cubic $Pm\bar{3}m$ (221). This is the `magnetic_space_group` integration test.

> **Requires**
> Needs `spglib` (with its magnetic tables) and a periodic structure. A structure carrying no moments at all is answered as a grey type II group, correctly — the dialog says so rather than leaving the reading unexplained.

15.10 Volumetric data

Analysis ⟩ **Volumetric Data** visualises 3D scalar fields: charge densities, electrostatic potentials, wavefunctions, ELF.

15.10.1 Loading fields

Load Field A... and **Load Field B...** read Gaussian `.cube` (including files written by Quantum ESPRESSO's `pp.x`), VASP `CHGCAR` / `LOCPOT` / `PARCHG` / `ELFCAR`, and XCrySDen `.xsf` grids. Each is reported with its grid dimensions and value range.

Field A defines the geometry; Field B is optional and colours it.

15.10.2 Render modes

Edit Volumetric Render offers two modes: **Isosurfaces** and **Color Slice**. Potential-map colouring used to be a third; it is now a group inside **Isosurfaces**, because it always was one — the same surface, extracted the same way at the same isovalue, differing only in what colours it.

15.10.3 Isosurfaces

The isovalue slider and numeric box re-extract the surface in real time. Extraction uses a marching-cubes-family algorithm on a tetrahedral decomposition of each grid cell, which is topologically unambiguous, with smooth normals from the field gradient and periodic stitching so surfaces close correctly across cell boundaries.

15.10.4 Potential map colour

The **Potential Map Color** group under **Isosurfaces** makes an EPM: the isosurface (say the electron density) shapes the geometry, and the field chosen under **Color by** (say the Hartree potential) is interpolated at every surface vertex and mapped through the **Color map**. **Invert palette** flips the ramp.

Custom range pins the colour scale to an explicit min/max instead of the colouring field's own extremes. A potential runs from deeply negative at the nuclei to nearly flat in the vacuum, so on the full range everything interesting sits in a sliver of the ramp — and a fixed window is also what lets two molecules be compared on one scale.

Because the colouring is an option on the surface rather than a mode, it applies to *every* ticked dataset at once: two orbitals can be shown side by side, both painted by the same potential.

> **Tip**
> The classic recipe: load the charge density as Field A, the local potential as Field B, pick an isovalue on the density that traces the molecular surface, and choose **Coolwarm** — the diverging palette puts the neutral potential at white, so electron-rich and electron-poor regions read immediately.

15.10.5 Slice planes

Color Slice cuts through the volume with a colour-mapped plane, oriented by Miller indices $(h\,k\,l)$ against the grid's own lattice and swept along its normal by the **Offset** slider.

Extent sets how far the plane is drawn, in unit cells across. **This unit cell only** keeps it inside the box the field was computed in — the honest extent, and the one to use when reading values off it. The replicated options tile the periodic field over the neighbouring cells, which is what makes a surface reconstruction or an adsorbate pattern legible instead of showing one cell floating on its own. **Outline the slice** draws its boundary, useful where the field has gone to zero and the quad would otherwise fade into the background.

Custom color range pins the ramp the same way the potential map's does: a few outlier voxels (a nuclear cusp, a spike at a boundary) otherwise compress everything else into one end of it.

15.10.6 Export

Export Isosurface (OBJ)... writes the triangle mesh with normals for external rendering; **Export Slice (CSV)...** writes the sampled plane as x, y, z, value.

15.11 Born charges and charged defects on VASP and Quantum ESPRESSO

Two modules that used to be GPAW-only now run on VASP and Quantum ESPRESSO as well, and in the Born case by a better route.

Born effective charges take the DFPT path on both: `LEPSILON = .TRUE.` for VASP, `ph.x` with `epsil = .true.` for QE. That is one linear-response run instead of $6N$ displaced SCFs, and it is an *analytic* derivative — there is no displacement amplitude to trade off against SCF noise and no linearity assumption left to check. The macroscopic dielectric tensor comes free with the same run. Both write the same `born_charges.json` the GPAW path writes, so the viewer and the phonon LO–TO block are unchanged.

> **Note**
> Both routes need an insulator or semiconductor. For a metal the macroscopic polarization is not defined: VASP writes no Born block and `ph.x` refuses `epsil`. The scripts say that rather than failing obscurely.

Charged defects read total energies and band edges from `vasprun.xml` or the `pw.x` output, and run one fixed-geometry SCF per charge state — `NELECT` for VASP (an *absolute* electron count, so $q = +1$ is one electron fewer) and `tot_charge` for QE (which follows the physical sign directly).

> **Requires**
> The FNV correction on these two engines is *delegated to pymatgen* (`pymatgen-analysis-defects`), reading `LOCPOT` for the potential alignment. Where pymatgen is not installed the run reports **uncorrected** energies and says so loudly. That is deliberate: uncorrected is a legitimate mode — it is exactly what a supercell-convergence study needs — whereas a hand-rolled correction that has never been checked against a reference is a plausible number of the right magnitude and the wrong value, applied silently to every point of the diagram.

15.12 Elastic properties

Electronics ⟩ Elastic Properties... computes the elastic stiffness tensor C_{ij} ($i, j = 1..6$ Voigt) by the finite-strain method: apply a small homogeneous strain to a relaxed reference cell and read off either the **stress** or the **energy** at each strained point, then fit. It reuses exactly the strain-generation core the Piezoelectric Tensor wizard uses to differentiate polarization over the same Calango-precomputed deformation gradients — here the same strained structures feed a stress or energy readout instead, so the strain stencil, the 2D vacuum-axis handling, and the point-group symmetrization are shared code. Unlike Piezoelectric Tensor, a ground-state baseline is **optional**: the Berry phase needs GPAW specifically, but stress and energy are available from any ASE calculator (EMT, MACE, VASP, Quantum ESPRESSO, ...) — pick "(none)" in the baseline combo to strain the current structure with the ordinary calculator settings page instead.

Stress-strain vs. energy-strain. Stress-strain (primary) reads $\sigma_i = C_{ij}\varepsilon_j$ directly off the stress tensor at each strained point: one component sweep already fills a whole matrix column at once, and a single strain magnitude ($\pm\delta$, the central difference) suffices. Energy-strain (fallback, for a calculator with no `get_stress()`) fits the diagonal term from the energy curvature, $C_{jj} = V_0^{-1}\, d^2E/d\varepsilon_j^2$, a quadratic fit needing at least 3 points; the off-diagonal C_{jk} additionally needs a combined two-component strain (Ravindran *et al.*, *J. Appl. Phys.* **84**, 4891 (1998)) — 4 extra evaluations per off-diagonal pair, 60 extra points for the full 6-component bulk case, shown in the wizard's cost estimate before launching. **Method** offers **Auto** (probe the reference point's `get_stress()`; fall back to energy-strain for the *whole* run if it raises, rather than mixing methods across components, which would not be physically consistent), or either method explicitly.

Clamped-ion vs. relaxed-ion follows the same terminology and trade-off as Piezoelectric Tensor below.

A monolayer is detected the same way as in the Piezoelectric Tensor wizard and restricts the strain set to the in-plane Voigt components. As with e_{ij}, raw C_{ij} (GPa) is vacuum-thickness-dependent for a slab, so a 2D structure additionally reports the in-plane block in **N/m** (area-normalized), $C_{ij}^{\mathrm{2D}} = C_{ij}^{\mathrm{3D}} \times L_{\mathrm{vacuum}}$, plus the derived 2D layer modulus, in-plane Young's modulus and Poisson's ratio, and the 2D restriction of the Born stability criterion, $C_{11}C_{22} - C_{12}^2 > 0$ and $C_{66} > 0$.

Born stability runs two independent checks: the general criterion (every eigenvalue of C positive, always computed) and, when the point group falls in one of five Laue classes (cubic, hexagonal, tetragonal, trigonal, orthorhombic), the crystal-class closed form of Mouhat & Coudert, *Phys. Rev. B* **90**, 224104 (2014). Unlike the piezoelectric tensor, C_{ij} is never forced to zero by inversion

symmetry, so there is no centrosymmetric refusal here.

Moduli. Voigt (uniform-strain, upper bound), Reuss (uniform-stress, lower bound, via $S = C^{-1}$) and Hill (arithmetic mean — the standard practical isotropic estimate) bulk and shear moduli; isotropic Young's modulus and Poisson's ratio follow from the Hill averages. Reuss and Hill are left unavailable, rather than silently reported as zero, when the tensor does not invert cleanly.

The viewer shows the 6×6 C_{ij} tensor (raw or symmetrized, GPa), both stability criteria with pass/fail per inequality, the moduli (or, for a 2D result, the layer modulus and in-plane Young's/Poisson's ratio), and — per requested component — the stress-or-energy-vs-strain points behind that column.

15.13 Piezoelectric tensor

Electronics ⟩ **Piezoelectric Tensor** computes

$$e_{i\alpha} = \frac{\partial P_i}{\partial \varepsilon_\alpha} \qquad (i = 1..3,\ \alpha = 1..6 \text{ Voigt}),$$

the piezoelectric tensor, by the finite-difference strain-polarization method: apply a small homogeneous strain to a relaxed reference cell, evaluate the total polarization P by the modern (Berry-phase) theory at each strained point, and differentiate. It reuses exactly the GPAW Berry-phase evaluation Born Effective Charges differentiates over *atomic displacements* — here the same evaluation is differentiated over *cell strain* instead, and only GPAW is offered for the same reason. Like Born Effective Charges, the run starts from a completed **Simulation** ⟩ **Single-point Calculation** `.gpw`, which supplies both the reference geometry and the calculator every strained run is rebuilt from.

The strain stencil. **Strain Components** picks which of the six Voigt components to differentiate (all six by default); **Strain magnitude** δ is the smallest sample point (0.5% by default); and **Points per component** offers the exact central difference (2, at $\pm\delta$) or a least-squares fit over four points ($\pm\delta, \pm 2\delta$) for a better-conditioned slope — the viewer plots the points behind every column, so the linearity the method assumes can be checked directly.

> **Note**
>
> **The polarization quantum.** P is defined only modulo eR/Ω, and the Berry phase computed at two different strain points routinely lands on different branches. The generated script resolves this with `np.unwrap` across each component's ordered strain series before differencing — the classic pitfall of this method.

Clamped-ion vs. relaxed-ion. Clamped-ion (the default) leaves the internal coordinates at their strain-scaled positions and re-converges only the SCF. Relaxed-ion additionally minimizes the internal coordinates at each fixed, strained cell before reading off the polarization — the physical, zero-stress response, at the cost of one geometry optimization per strain point.

Proper vs. improper, and symmetry. The raw finite difference is the *improper* (clamped-cell-shape) tensor, which carries a pure volume-definition artefact the physical *proper* response does not have. The script applies the correction from Vanderbilt, *Berry-phase theory of proper piezoelectric response*, J. Phys. Chem. Solids **61**, 147 (2000) (arXiv:cond-mat/9903137, Eq. 15), symmetrized in the strain pair. **Use point-group symmetry** runs spglib on the reference structure first: if the point group contains the inversion, the run refuses outright — piezoelectricity is forbidden by symmetry for every centrosymmetric point group, exactly rather than approximately.

Otherwise the assembled tensor is symmetrized by averaging the general rank-3 tensor transform over every point-group operation, which both zeroes forbidden components and cleans up numerical noise in the allowed ones.

Supplying an elastic stiffness tensor C (Voigt, GPa) in the wizard turns on the piezoelectric *strain* tensor $d_{i\alpha} = \sum_\beta e_{i\beta} S_{\beta\alpha}$ with $S = C^{-1}$, reported in the conventional pm/V alongside e_{ij}.

15.14 Raman and infrared spectra

Electronics ⟩ Raman and IR Spectroscopy... computes both spectra of the Γ-point phonons. They describe the *same* modes and differ only in which electronic response couples to them:

Infrared intensity is the change in macroscopic *polarization* a mode produces, $\left|\sum_\kappa Z^*_\kappa \cdot e_\kappa / \sqrt{M_\kappa}\right|^2$. In a periodic crystal there is no molecular dipole to differentiate, so the Born effective charges Z^* are the only route to it.

Raman activity is the Placzek invariant $45a'^2 + 7\gamma'^2$ built from $\partial\chi/\partial Q$, the change in *polarizability* — the same response the Optics module evaluates, taken in the static limit.

Three engines are offered, and what one run of each can produce differs by orders of magnitude. This is the choice the wizard's cost estimate exists to inform, so it is worth stating plainly:

GPAW Finite displacements throughout: $6N$ force evaluations for the Hessian, and $6N$ further self-consistent runs (each followed by six dielectric evaluations) for $\partial\alpha/\partial u$. Z^* is *inherited* from a completed **Born Effective Charges** run rather than recomputed, because GPAW's own route to it is another $6N$ Berry-phase runs for a quantity the Raman spectrum never uses. Supplying one is therefore what turns the IR column on; without it the phonons and the Raman spectrum are computed as usual and every IR intensity is reported as zero, with `ir.computed = false` recording which happened.

VASP One DFPT run — `IBRION = 8` with `LEPSILON = .TRUE.` — returns the force constants, every ion's Z^* and ε_∞ together, so the *infrared* spectrum costs a single job and nothing has to be selected for it. The Raman half has no such shortcut: VASP computes no Raman tensor, so $\partial\alpha/\partial u$ comes from differencing ε_∞ over $6N$ displaced `LEPSILON` runs.

Quantum ESPRESSO One `ph.x` run at $q = 0$ returns all of it. `epsil` gives the force constants, Z^* and ε_∞; `lraman` adds the Raman tensor as an analytic *third-order* response — no displacement amplitude to trade off against SCF noise, and no linearity assumption left to check.

> **Requires**
>
> `lraman` is implemented for **norm-conserving** pseudopotentials only. With an ultrasoft or PAW set `ph.x` declines rather than approximating, and the run stops with a message naming that as the cause. The infrared half is unaffected — turn the Raman spectrum off and the same job still produces it. Edit the pseudopotential map in the generated script before running: the guess is the conventional `<Symbol>.UPF` naming, which a real library rarely matches.

All three engines share one physics core — the mass-weighted diagonalization, the two contractions, the Stokes prefactor and the broadening — and write the same `raman_ir.json`, so one viewer serves all of them. The file records which route produced it: `engine` and `method` name the

workflow, and `displacement_A` is 0 for a DFPT result, because there was no displacement to report.

> **Note**
> The laser wavelength, temperature, broadening and frequency window change only the *reported* spectrum, not the physics. Mode frequencies and activities are computed once and the experiment-specific factors applied afterwards, so trying a different laser line does not mean another run — reopen the viewer instead.

15.15 Partial charges

Analysis ⟩ **Partial Charge Analysis...** partitions a converged electron density among the atoms. Three schemes are offered — **Bader** (topological zero-flux basins), **Voronoi** (nearest-atom cells) and **Hirshfeld** (stockholder weights against a promolecule) — and all three are native: they run on a standardized 3D grid and know nothing about the code that produced it.

15.15.1 Density source and engine

Charge-density source lists the completed processes whose directories hold a density; **Density format** decides how it is read. `Auto-detect` covers the normal case and the explicit entries exist for a directory holding output from more than one engine, where guessing would be a coin toss.

GPAW `get_all_electron_density()` from the run's `.gpw`.

VASP `AECCAR0` + `AECCAR2` when the run wrote them (`LAECHG = .TRUE.`), otherwise `CHGCAR`.

Quantum ESPRESSO `pp.x` (`plot_num = 0`) exports the density to a cube first. Set `CALANGO_PP_X` if `pp.x` is not on `PATH`.

> **Requires**
> **Bader needs the all-electron density.** Partitioned on the pseudo-valence density alone, its basins follow the wrong topology and the charges come out systematically small. For VASP that means `AECCAR0` and `AECCAR2`; given only a `CHGCAR` the run says so in its log rather than returning quiet nonsense.

15.15.2 Scope

Current structure only partitions the displayed frame. **All structures in the trajectory** partitions every frame — and it is not a loop over one density. A charge is a property of a *converged density*, so each frame needs its own; the run looks for one density file per frame and reports what it found rather than reusing a single density as though it described every geometry.

15.15.3 Reading and keeping the results

Load Results... fills the per-atom table and a summary line above it: the net charge, the range, and the mean per element — the number people actually quote, and the one a 200-row table does not give up easily.

> **Tip**

The net charge is the check to read first. For a neutral cell it must come out near zero; anything larger is density the integration lost, and it bounds how much any single per-atom value can be trusted. The summary flags a residual above 0.05 e on its own.

Write into Trajectory stores the charges on the structure as `initial_charges` — the array name ASE reads and writes — through the document's undo stack. Saving as extended XYZ then carries them as a column of the file, and they appear in **Edit Structure...** (§7.2.2). The same field tints the viewport, so the colours and the file cannot disagree.

15.16 Charge density difference

Analysis ⟩ Charge Density Difference (CDD)... computes

$$\Delta\rho = \rho_{A+B} - \rho_A - \rho_B,$$

the redistribution of charge caused by putting two fragments together. Each of the three densities on its own is dominated by the atomic cores and shows nothing; the difference shows the bond.

15.16.1 Step 1 — density source

Pick a completed **Simulation ⟩ Single-point Calculation**; that run supplies ρ_{A+B} and, more importantly, the calculator both fragments are rebuilt from. Choose whether to difference the **All-electron density** or the **Pseudodensity**. The pseudodensity is usually the clearer field to read: it carries no nuclear cusps, so the isosurface scale is set by the bonding features rather than by the cores.

GPAW, VASP and Quantum ESPRESSO are all supported, and the engine is taken from the parent run's provenance rather than asked for again — $\Delta\rho$ only means something if all three densities were computed the same way.

GPAW Restarts from the `.gpw` and reads the exact calculator back out of it. This is the strongest of the three guarantees: there is nothing left to re-specify.

VASP Reads `CHGCAR`, or `AECCAR0` + `AECCAR2` for the all-electron form (which needs `LAECHG = .TRUE.` on the parent run). Each fragment is re-run with the parent's settings.

Quantum ESPRESSO Exports each density through `pp.x` (`plot_num = 0`) as a Gaussian cube, since QE keeps its density in a binary file with no public reader. Set `CALANGO_PP_X` if `pp.x` is not on `PATH`.

Requires

For VASP and Quantum ESPRESSO the fragment runs pin the FFT grid explicitly to the parent's (`NGXF`/`NGYF`/`NGZF`, `nr1`/`nr2`/`nr3`). Both codes choose that grid from the cutoff *and* the cell contents, so a fragment with fewer atoms in it can otherwise land on a different one — and two densities sampled on different grids cannot be subtracted at all. GPAW needs none of this because the restart already fixes the grid.

15.16.2 Step 2 — subsystem partition

Two columns list the atoms of the selected run. Move atoms across with the arrow buttons or by double-clicking a row. The split is exhaustive — every atom belongs to exactly one side — because $\Delta\rho$ only integrates to zero if A and B together are the whole system.

15.16.3 What the run does

Both fragments are recomputed in the parent's own unit cell, at the parent's geometry, with the parameters read back out of its `.gpw`: cutoff, XC functional, grid, k-points and convergence cannot drift between the three terms. *Nothing is relaxed.* $\Delta\rho$ is defined at one geometry, and letting a fragment relax would mix charge transfer with structural rearrangement into a quantity nobody can read off an isosurface.

> **Note**
> Cutting a closed-shell molecule in half leaves open-shell fragments, and an isolated atom with a partly-filled degenerate shell often will not converge under the settings that converged the molecule in one pass. The run therefore tries the parent's exact parameters first, then adds occupation smearing, then spin polarization, and reports in the log which was needed. The reported `net` charge is the check: A and B together hold exactly the electrons $A{+}B$ does, so it must come out at zero.

15.16.4 Result

The difference is written as `cdd.cube` and loaded automatically into the **Volumetric Data** dock (§15.10), where it renders in the main 3D viewport like any other grid. A **Coolwarm** colormap is the natural choice: accumulation and depletion sit on opposite sides of white. The status bar reports how much charge was redistributed, from `cdd.json`.

15.17 Adsorption and catalysis

Analysis ⟩ **Adsorption & Catalysis** finds high-symmetry adsorption sites on a slab and populates them.

15.17.1 Site detection

Opening the dialog detects sites on the top surface automatically and reports a census, for example *4 top, 12 bridge, 4 fcc, 4 hcp*:

`top` Directly above a surface atom.

`bridge` Midway between nearest-neighbour surface atoms.

`fcc`, `hcp` Threefold hollows, distinguished by whether a second-layer atom sits directly underneath (hcp) or not (fcc).

`hollow` A threefold hollow that could not be classified, typically because there is no second layer.

Neighbour vectors are enumerated over periodic images, so the census is correct even for a small $p(1 \times 1)$ cell where surface atoms bond to their own periodic copies. The table lists every site with its in-plane coordinates and can be filtered by type.

15.17.2 Placing adsorbates

Adsorbate `OH`, `O`, `H`, `CO`, `CHO`, `H2O`, `N2`, `O2`, `NH3`, `CH4`, or any `ase.build.molecule` name or chemical formula you type.

Height Anchor-atom height above the surface layer, default 2 Å.

Place on Selected Sites Adds one copy per selected table row.

Molecules are anchored by the chemically sensible atom and oriented upright: O down for `OH` and `H2O`, C down for `CO` and `CHO`, N down for `NH3`.

15.17.3 Coverage series

The **Coverage series** group generates a systematic set in one step: choose a **Site family** and check the coverages you want — 0.25, 0.50, 0.75, 1.00 ML. Calango places evenly strided subsets of that site family and opens one labelled tab per coverage, ready for a batch of adsorption energy calculations.

15.18 Brillouin zone and k-path builder

Analysis ⟩ **Brillouin Zone / k-Path** shows the first Brillouin zone as a Wigner-Seitz cell of the reciprocal lattice, with high-symmetry points labelled from ASE's Bravais lattice detection.

Drag to rotate, Shift+drag to pan, wheel to zoom; **Orthographic projection** removes perspective foreshortening when you need to read the zone geometry.

15.18.1 Building a path

Click high-symmetry points in the 3D view to append them to the path. The list shows the sequence with fractional coordinates.

Suggested Loads ASE's suggested path for the lattice.

Break Starts a new discontinuous section, giving paths like $\Gamma \to X \,|\, M \to R$.

Undo, Clear Remove the last point, or start over.

Points per segment Sampling density, default 40.

15.18.2 Export

Export k-Path... writes the path for an external code:

Format	Typical file
VASP `KPOINTS` (line mode)	`KPOINTS`
Quantum ESPRESSO `K_POINTS crystal_b`	`kpath_qe.in`
CASTEP `SPECTRAL_KPOINT_PATH`	`kpath_castep.cell`
SIESTA `BandLines`	`kpath_siesta.fdf`
ASE / Python script	`kpath_ase.py`

Export Figure (PNG/SVG)... renders the zone at high resolution; the SVG output is vector and drops straight into a paper figure.

CHAPTER 16

Publication Output

16.1 Still images

File ⟩ **Import / Export** ⟩ **Export Image** (Ctrl+E) renders the current view off-screen at any resolution, independent of the window size, with 8× multisample anti-aliasing.

Resolution preset 720p, 1080p, 4K, or custom width and height up to 8192 px.

Background **Transparent (PNG only)**, **Solid white**, or a custom colour.

Format PNG or JPEG, chosen by the extension.

> **Tip**
> Transparent PNG is the right choice for figures that will sit on a coloured background in a paper or slide. Combine it with an orthographic, axis aligned view (O, then an alignment button) for a clean, reproducible result.

16.2 Animations

File ⟩ **Import / Export** ⟩ **Export Animation (GIF/MP4)** writes either a turntable rotation of a static structure or a playback of a trajectory.

Source **Turntable rotation (360°)** or the current trajectory.

Resolution preset Up to 4K.

Frames per second Playback rate.

Background Including **Transparent (GIF only)**.

Format Animated GIF or H.264 MP4.

> **Requires**
> GIF export needs `pillow`; MP4 needs `imageio` and `imageio-ffmpeg`. MP4 has no alpha channel — asking for a transparent MP4 falls back to a solid background.

16.3 Ray-traced rendering

File ⟩ Import / Export ⟩ Ray-Traced Render exports the active scene — camera, lights, representation, colours, multiple bonds, unit cell — to an external ray tracer for publication-quality output with true shadows and reflections.

Engine **POV-Ray** or **Tachyon**.

Width (px), Height (px) Default 1920 × 1440.

Background **Solid white** or **Viewport color**.

Renderer binary Path to the executable; the bare names `povray` and `tachyon` are resolved on `PATH`. The path is remembered per engine.

Two ways to use it:

Save Scene File... Writes just the scene (`.pov` or `.dat`) for you to render elsewhere, or to edit by hand before rendering.

Render... Writes the scene next to your chosen PNG output and invokes the renderer, streaming its output into the log pane so you can watch progress and diagnose failures.

> **Requires**
> Needs POV-Ray or Tachyon installed separately. If the binary cannot be started, the log says so explicitly rather than failing silently.

16.4 Data exports at a glance

Nearly every analysis tool exports its numbers, so figures can be regenerated in your own plotting stack.

Source	Formats	Typical file
RDF	CSV, DAT	`rdf.csv`
Bond length / angle	CSV, DAT	`bond_lengths.csv`
Structure factor	CSV, DAT	`structure_factor.csv`
XRD pattern	CSV, DAT	`xrd_pattern.csv`
XRD peak list	CSV, DAT	`xrd_peaks.csv`
Warren-Cowley	CSV	`warren_cowley.csv`
Job metric series	CSV, DAT	`energy.csv` ...
Band structure	CSV, DAT	`bands.csv`
PDOS	CSV, DAT	`pdos.csv`
Isosurface mesh	OBJ	`isosurface.obj`
Volumetric slice	CSV	`slice.csv`
Phonon bands / DOS	CSV	`phonon_bands.csv`
k-path	5 codes	Section 15.18
ML datasets	extxyz, ASE db	Section 9.7
Brillouin zone figure	PNG, SVG	`brillouin_zone.svg`

CHAPTER 17

Reference

17.1 Keyboard shortcuts

17.1.1 Viewport

Single-letter shortcuts, no modifier. They yield to text fields while you are typing.

Key	Action
R	Rotation mode
T	Translation (pan) mode
S	Selection mode
I	Insertion mode
D	Distance measurement mode
A	Angle measurement mode
O	Toggle orthographic / perspective
F	Frame structure
Tab / Shift+Tab	Next / previous structure tab
Delete / Backspace	Delete selection (Selection mode)

Note

Every row above except Delete/Backspace is remappable in **Edit** ⟩ **Preferences** ⟩ **Hotkeys**: a table with one key-capture cell per action, live conflict detection against both this list and the Application shortcuts below, and a per-row **Reset** plus a **Reset All to Defaults**. Tab/Shift+Tab cycling wraps around and is a no-op with a single tab open; both keys yield to the viewport losing focus, same as the letter shortcuts above.

17.1.2 Molecular Design

Live only while the Molecular Design window (Section 6.1) has focus, which is what lets both it and the viewport use single letters without either seeing the other's keys. All remappable, under **Molecular Design** in **Edit** ⟩ **Preferences** ⟩ **Hotkeys**.

Key	Tool
V	Selection
1 / 2 / 3	Single / double / triple bond
4 / 5	Wedge (bold) / hashed stereo bond
C	Chain
L	Atom label
X	Caption
P	Formal charge
E	Eraser
B	Ring template
Y	Tidy the drawing
Ctrl+Return	Send to 3D Viewport

Undo, redo, copy, paste and select-all use the application's own bindings inside the sketcher too, so remapping Ctrl+Z remaps it everywhere.

17.1.3 Application

Shortcut	Action
Ctrl+N	New workspace
Ctrl+O	Open structure
Ctrl+T	Open trajectory
Ctrl+Shift+O	Open project
Ctrl+S	Save project
Ctrl+Shift+S	Save structure as
Ctrl+Shift+T	Save trajectory as
Ctrl+E	Export image
Ctrl+W	Close tab
Ctrl+Q	Quit
Ctrl+Z	Undo
Ctrl+Shift+Z	Redo
Ctrl+Shift+A	Add atom
Ctrl+B	Bond editor
Ctrl+P	Preferences
Ctrl+R	Single-point calculation

On macOS, Ctrl in this table is Cmd.

17.1.4 Mouse

Gesture	Action
Left drag	Depends on interaction mode
Middle drag	Pan (any mode)
Shift+left drag	Pan (any mode)
Wheel	Zoom
Left click	Pick atom
Ctrl/Cmd+click	Toggle atom in selection
Double click	Frame structure

17.2 Menu map

File New Workspace · Open (Structure, Trajectory) · Save (Structure As, Trajectory As) · Import / Export (Image, Animation, Ray-Traced Render) · Project Workspace (Open, Save, Save As)

· Close Tab · Quit

Edit Undo · Redo · Add Atom · Selection (Change Element, Translate) · Delete Selected Atoms · Bond Editor · Unit Cell (Create Supercell) · Preferences

View Orthographic · Overlays (Show Unit Cell) · Alignment (Frame Structure, XY, XZ, YZ) · Visual Effects · panel toggles

Build Nanomaterial Builder · Surface Slab · Metallic Nanoparticle (Wulff) · Special Quasirandom Structure (SQS) · Normal Modes / Phonon Builder · From Database · Structure Perturbation / Noise

Simulation New Calculation (ASE) · New Remote Calculation · Electronic Bands / PDOS · Dataset Manager (MLIP)

Analysis Structure Factor S(q) · X-Ray Diffraction · Radial Distribution Function · Bond Length / Angle Distributions · Coordination Numbers (CN / GCN) · Warren-Cowley analysis · Local Entropy Analysis · Raman Modes · Volumetric Data · Adsorption & Catalysis · Brillouin Zone / k-Path

Help Documentation · GitHub Repository · About Calango

17.3 Python dependencies by feature

Package	Needed for
`ase`, `numpy`	Everything (structure I/O, building, jobs)
`spglib`	Symmetry detection, Raman modes
`paramiko`	HPC panel
`pillow`	GIF animation export
`imageio`, `imageio-ffmpeg`	MP4 animation export
`mace-torch`	MACE machine-learning potentials
`gpaw`	DFT band structures and PDOS
`icet`	Preferred SQS backend (optional)

External binaries: `pw.x` (Quantum ESPRESSO), `siesta`, VASP, `orca`, `povray` or `tachyon`.

17.4 Job directory contents

A typical job directory under `.calango_tmp/` (Section 3.4.1):

File	Written by
`run.py`	The generated (possibly hand-edited) script
`structure.extxyz`	The staged input geometry
`opt.traj`	Geometry optimization trajectory
`md.traj`	Molecular dynamics trajectory
`optimized.extxyz`	Final relaxed geometry
`bands.json`, `pdos.json`	Electronic structure results
`phonon_bands.csv`, `phonon_dos.csv`	Phonon results
`perturbed.extxyz`	Noise ensemble checkpoint
`job.sh`	Scheduler wrapper (remote jobs)
`calango_job.out`, `calango_job.err`	Remote stdout/stderr

17.5 Progress markers

Generated scripts communicate with the interface by printing markers to standard output. If you hand-edit a script or write your own, emitting these keeps the Job panel live:

Marker	Effect
`CALANGO_PROGRESS step total`	Progress bar
`CALANGO_ENERGY step value`	Energy plot (eV)
`CALANGO_TEMP step value`	Temperature plot (K)
`CALANGO_FMAX step value`	Force plot (eV/Å)
`CALANGO_PRESSURE step value`	Pressure plot (GPa)
`CALANGO_TARGET_TEMP value`	Thermostat reference line
`CALANGO_TARGET_PRESSURE value`	Barostat reference line
`CALANGO_CELL` + `CALANGO_FRAME`	Live trajectory streaming
`CALANGO_RESULT ...`	Final summary line

17.6 Troubleshooting

Structures will not open; job features are disabled. ASE is not importable. Run `calango --probe-python` to see which interpreter is being used, then either install ASE there or set `CALANGO_PYTHON` to an environment that has it (Section 1.2).

The space group is `P1` for an obviously symmetric crystal. Numerical noise in the coordinates. Raise **Tolerance** in the **Structure** panel — a relaxed cell often needs 0.01 Å or more.

Bonds are missing or spurious. Adjust **Covalent cutoff multiplier** in the bond editor (Section 7.4), or add and suppress individual bonds by hand. Metallic and ionic systems rarely match a covalent-radius rule.

Double and triple bonds do not appear automatically. By design. Assign them with the **Bond order** buttons (Section 7.5).

A job fails immediately with an import error. The job is running in a different environment from Calango itself. Check the **Execution Environment** status line in the calculation dialog (Section 9.3).

The GPAW backend is greyed out. GPAW is not importable in the selected environment. Point the **Environment** selector at a Conda environment where it is installed.

The remote panel reports a connection failure. Test the same host, port, user and key with `ssh` from a terminal first. Remember that with a **Key file** set, the **Password** field doubles as that key's passphrase rather than a login password (Chapter 11 explains the full inference).

Adsorption site detection finds no bridge or hollow sites. The surface cell is too small to have in-plane neighbours. Build a supercell of the slab first.

The viewport looks soft after enabling depth of field. Expected — the effect trades multisample anti-aliasing for the blur pass (Section 4.7). Disable it before exporting still images.

Animation export fails. Install `pillow` for GIF, or `imageio` and `imageio-ffmpeg` for MP4, in Calango's Python environment.

17.7 Citing Calango

Help ⟩ **About Calango** has a **Citations** tab, right after **Third-Party Licenses**: it groups the references worth citing for a Calango-produced result into **Core** (Calango itself, and ASE, which every generated script builds on), **Databases** (Materials Project, PubChem, C2DB — whichever a structure actually came from), **Calculators** (the DFT/MD engine a wizard generated a script for — cite the ones a given result actually ran on, not the whole list), and **Libraries** (spglib, NumPy, phonopy, icet, dftd4 — customary to cite where a venue expects it). Each entry has a **BibTeX viewer...** button opening a read-only, monospaced, copyable BibTeX record, so citing a result correctly never means retyping a reference by hand or guessing a citation key.

17.8 Further reading

`README.md` Feature overview and build instructions.

`docs/sphinx/` The Sphinx manual — the primary, most frequently updated reference, covering every builder, wizard, calculator and analysis viewer in more depth than this guide.

`docs/packaging.pdf` Building the macOS and Linux installers.

`https://github.com/seixas-research/calango` Source, issues and releases.

`https://wiki.fysik.dtu.dk/ase/` ASE documentation — the reference for anything in a generated script.