CutiePie v1.6-003 — A User's Guide

CutiePie (a.k.a. QtPy) is a desktop viewer for **live nuclear-physics spectra** served by SpecTcl. It shows one or more histograms ("spectra") in a grid of panels, refreshes their counts in real time, and gives you the everyday analysis tools: gates (graphical cuts), summing regions, peak finding, curve fitting, data export, and reference-image overlays.

This guide is task-oriented: each section tells you **what to click and what you will see**. It assumes you already know, roughly, what a spectrum, a gate, and a fit *are* — the goal here is to teach *this program's* buttons.

> **Before you start, you need:** a running **SpecTcl** instance with some bound spectra, reachable over its **REST** port and **mirror** port. If you don't have one, ask whoever runs your DAQ for the host name and those two port numbers. Without a live server CutiePie opens fine but shows no data.

> *All button and menu names in this guide were checked against the program's source. If a label on your screen differs, your install may be a different version — tell your instructor.*

How the window is organized

There are three things to know your way around:

1. **The config strip** — the single row of controls across the top. There is **no menu bar**; everything global lives here.
2. **The pad grid** — a tabbed area below, divided into panels ("pads"). You add a spectrum to a pad.
3. **The "Extra" window** — a separate popup (titled *Special Functions*) that holds Fitting, Peak Finder, Jupyter export, image Overlay, and a few option checkboxes. Open it with the gold **Extra** button.

Read the config strip left to right: **Connect** → **Geometry** (rows, columns, Apply Geometry, More) → **Spectrum** (the spectrum list, Add, Update, refresh interval, Colormap) → **Gate / Sum. Region / Integrate** → **Extra / Exit**.

---

1. Getting started & connecting

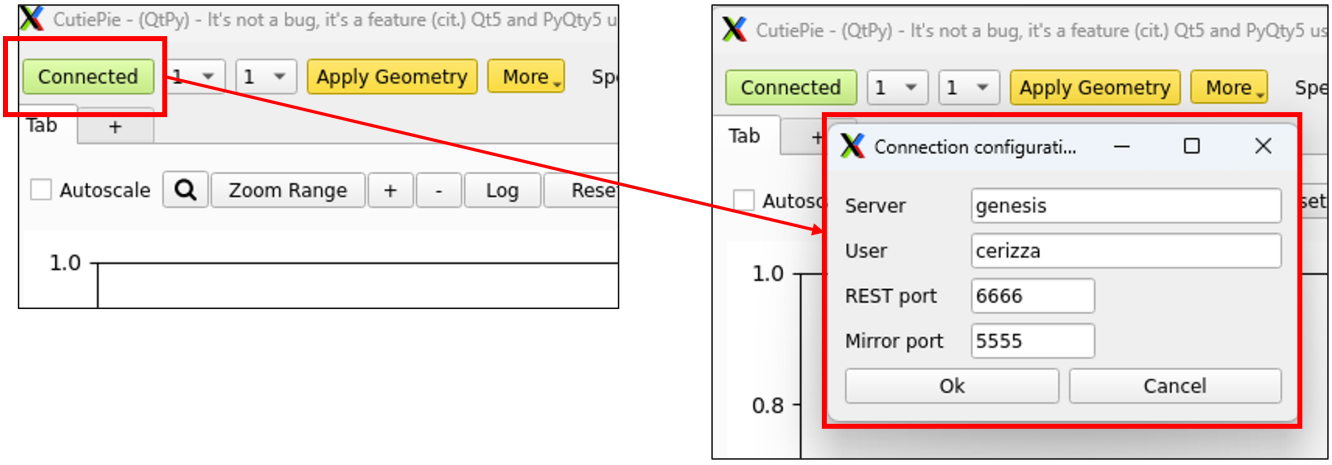
CutiePie is launched from its `gui/` folder by running `Main.py` (your install may wrap this in a launcher script or menu entry — check with your instructor).

To connect to your SpecTcl server:

1. Click the red **Connect** button (far left of the config strip).
2. A small **Connection configuration** window opens with four fields:
    - **Server** — the host name where SpecTcl runs (e.g. `localhost` or a machine name).
    - **User** — your username (pre-filled; usually leave as is).
    - **REST port** — SpecTcl's REST/control port number.

• **Mirror port** — SpecTcl's shared-memory mirror port number.

3. Fill in the host and the two port numbers, then click **Ok**.



*The **Connected** button (far left) and the **Connection configuration** dialog — Server, User, REST port, and Mirror port.*

Watch the Connect button change color — this is your connection status:

Color & text	Meaning
Red — "Disconnected"	not connected
Amber — "Connecting to mirror…"	handshake in progress (button briefly disabled)
Green — "Connected"	live; spectra and counts are flowing

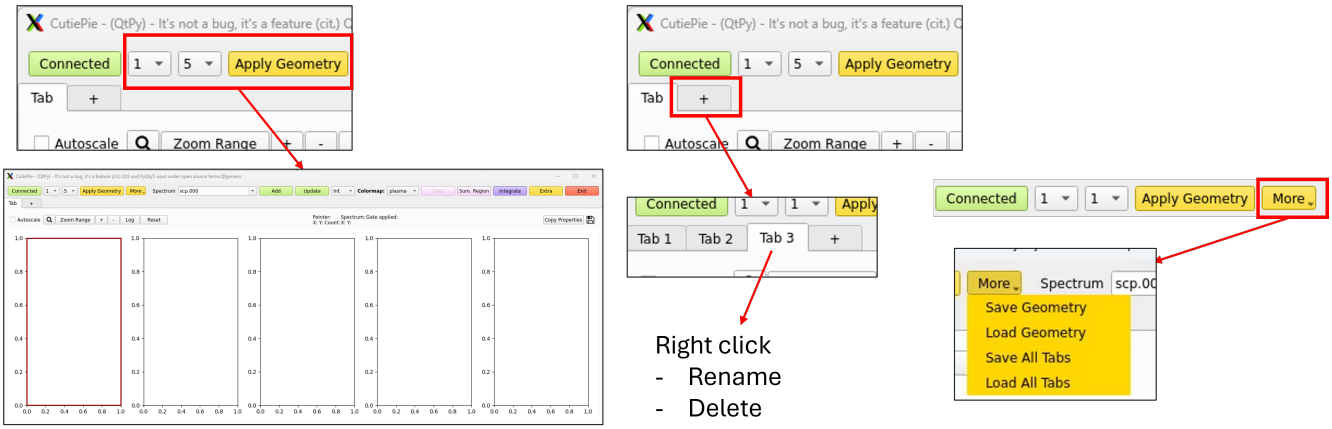
> *Shortcut:* if the REST and mirror ports are already filled with valid numbers
> when the program starts, CutiePie may connect automatically — you'll see the
> button already green.

Nothing showed up? Check, in order: the server name and both port numbers are correct; SpecTcl is actually running; and the **mirror** service (not just REST) is up — the counts arrive over the mirror port, so a REST-only connection shows spectrum *names* but no live data.

---

2. Laying out the workspace: geometry & tabs

A "pad" is one panel that holds one spectrum. You first choose how many pads to show, in a rows × columns grid.



*Left: a 1×5 grid built with **Apply Geometry**. Middle: add tabs with the **+**, and right-click a tab to **Rename** / **Delete**. Right: the **More** menu — Save/Load Geometry and Save/Load All Tabs.*

Set the grid:

1. In the **Geometry** block, pick a number of rows in the first small dropdown and columns in the second (each goes 1–9).
2. Click **Apply Geometry** (the gold button). The current tab rebuilds into that grid of empty pads.

Select a pad: single-click inside a pad. A **red dashed rectangle** marks the selected pad — that's where the next **Add** (Section 3) will drop a spectrum, and it's the pad most tools act on.

Working with tabs (each tab is an independent pad grid):

- **New tab:** click the **+** tab at the right of the tab bar.
- **Rename or delete: right-click** a tab → **Rename** or **Delete**.
- *Gotcha:* you can't delete the tab you're currently on — switch to another tab first, then delete it.
- **Reorder:** drag a tab left or right (the **+** always stays last).

Saving and restoring layouts — click the **More** button (in the Geometry block) for a dropdown:

- **Save Geometry** / **Load Geometry** — save or reload *the current tab's* layout as a `.win` file (which spectra are in which pads, their zoom and log settings).
- **Save All Tabs** / **Load All Tabs** — save or restore your **whole multi-tab session** at once. Loading replaces everything currently open.

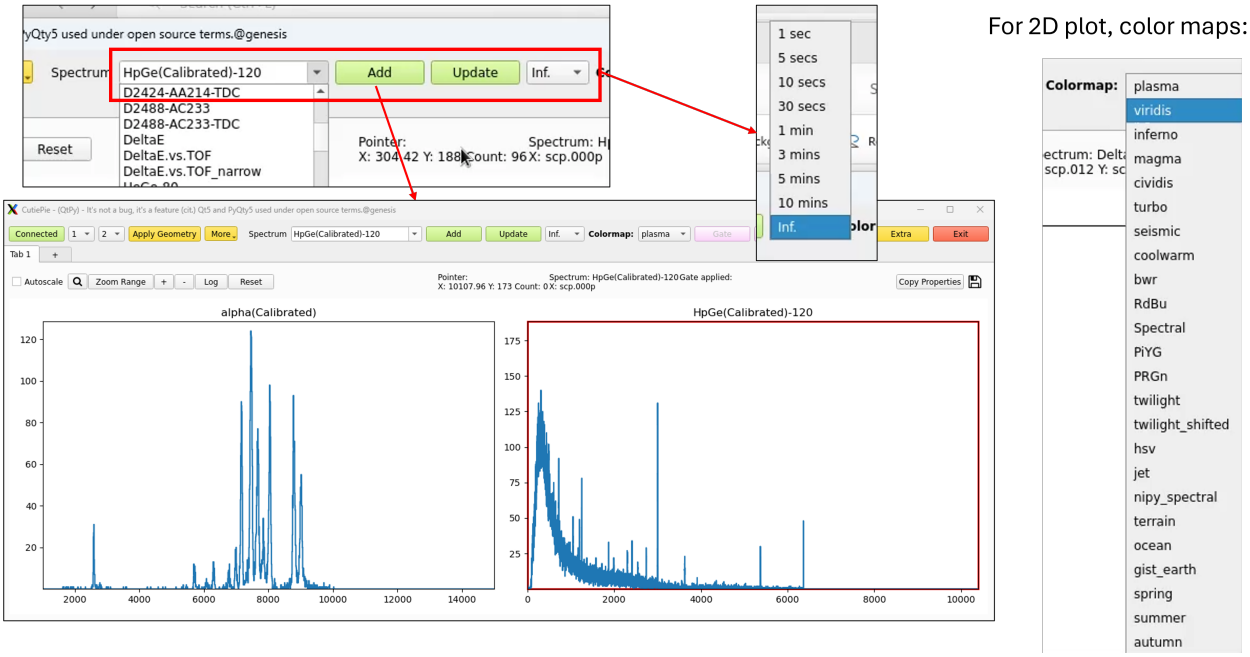
> *If you mix them up:* **Load Geometry** now recognizes when you hand it a
> multi-tab session file by mistake — it tells you how many tabs the file holds
> and asks whether to load them all (which replaces your current tabs) rather
> than failing. Point it at something that isn't a geometry file and it warns
> *"Not a readable geometry file — nothing was changed."*

---

3. Adding & viewing spectra

Once you're connected (green button) and have a pad grid:

1. Click the **Spectrum** dropdown and start typing — it filters the list of bound spectra by any part of the name (not just the start).
2. Pick the spectrum you want.
3. Single-click the pad you want it in (red dashed rectangle appears).
4. Click **Add**. The spectrum draws in that pad.



For 2D plot, color maps:

*The **Spectrum** dropdown (type to filter), **Add** / **Update**, the refresh-interval dropdown (1 sec … Inf.), and the **Colormap** list for 2D spectra.*

Keeping data live:

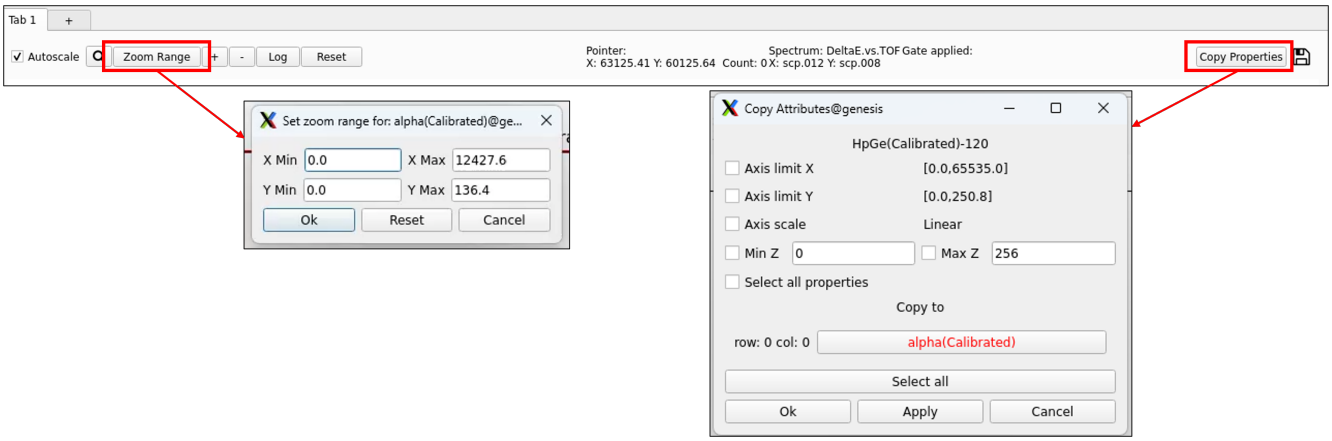
- **Update** redraws the selected pad on demand (a one-shot refresh).
- The **refresh-interval dropdown** (next to Update) controls automatic refreshing: choices are **1 sec, 5 secs, 10 secs, 30 secs, 1 min, 3 mins, 5 mins, 10 mins,** and **Inf.**
- *Gotcha:* it starts on **Inf.**, which means **no auto-refresh**. To watch counts grow live, choose an actual interval (e.g. *5 secs*).

Colormap (2D spectra): the **Colormap** dropdown (in the Spectrum block) recolors 2D histograms — `plasma`, `viridis`, `jet`, and many more, plus **custom** (loads a colormap from a file).

---

4. Reading & navigating a plot

Every pad has its own small toolbar. From left to right it holds: **Autoscale**, the **zoom (magnifier)** button, **Zoom Range**, **+**, –, **Log**, **Reset**, then the cursor readout labels, **Copy Properties**, and the **Save** figure button. Each is described below.



*The pad toolbar; **Zoom Range** opens the* Set zoom range *dialog (exact X/Y/Z limits); **Copy Properties** opens the* Copy Attributes *dialog to copy display settings onto another pad.*

• **Autoscale** (checkbox) — when ticked, the pad rescales its y-axis (1D) or color/z-axis (2D) to fit the data automatically. Untick it to keep a fixed scale.

• ■ **Zoom (magnifier) button** — *left-click* it to arm rubber-band zoom, then drag a rectangle on the plot to zoom into it. *Right-click* the button instead for **Set Zoom Range Manually**, which opens a dialog (see Zoom Range below).

• **Zoom Range** — opens **Set zoom range for: <spectrum>**, where you type exact limits: **X Min / X Max / Y Min / Y Max** (and **Z Min / Z Max** for 2D). Then **Ok** applies them, **Reset** clears the zoom, **Cancel** backs out.

• **+ / –** — zoom in / out in fixed steps (raise/lower the y or z ceiling). The keyboard **+** and – keys do the same on the selected pad.

• **Log** — *left-click* toggles a logarithmic axis for this pad. *Right-click* for **Log all** / **unLog all** (apply to every pad at once).

• **Reset** — *left-click* restores this pad's full range ("home"). *Right-click* for **Reset all**.

• **Readout labels** (Pointer / Spectrum / Gate) — these just display the cursor position, counts under the cursor, and any applied gate. Not clickable.

• **Copy Properties** — copies display settings (axis limits, log, etc.) from one pad onto another. Opens a dialog where you tick which properties, pick the target pad, and click **Apply** (or **Ok** to apply and close).

• ■ **Save** — the standard matplotlib "save figure to image file" button.

> **Enlarging a pad — remember this one. Double-click** a pad to blow it up
> to fill the tab (double-click again to return). Enlarging isn't just for a
> better view: **you must enlarge a pad before you can draw gates or summing > regions on it** (Sections 5 and 6). The **Gate** button stays greyed out until
> a pad is enlarged.

---

5. Gates (graphical cuts)

A gate is a region you draw on a spectrum to select events. In CutiePie you draw gates directly on the plot and they're saved back to SpecTcl.

> ■■ **Enlarge first.** Gates only work on an enlarged pad. **Double-click the > pad** you want to gate. If the **Gate** button is greyed out, you haven't

> enlarged a pad yet.

Create a gate:

1. Double-click a pad to enlarge it.
2. Click **Gate** (pink button, config strip). The **Gate Editor** window opens:
    - **Create** radio is selected.
    - **Name** — pre-filled `gate-001` (you can type your own name).
    - **Type** — filled automatically from the spectrum: a **1D** spectrum gives s (a slice); a **2D** spectrum gives c (contour) or b (band).
    - **Points** — a readout that lists the points as you place them.
3. **Place the gate on the plot by left-clicking:**
    - **1D:** two left-clicks set the low and high edges. (Only the last two clicks count, so you can re-click to adjust.)
    - **2D:** left-click each vertex of your polygon; a contour auto-closes back to the first point. **Right-click** removes the last vertex if you misplace one.
4. Click **Ok**. The gate is saved to SpecTcl and drawn on the pad.

*Creating a gate: (1) double-click a pad to enlarge it, (2) the **Gate** button becomes enabled, (3) the **Gate Editor** opens (Create, Name, Type, Points) while you left-click the shape on the plot.*

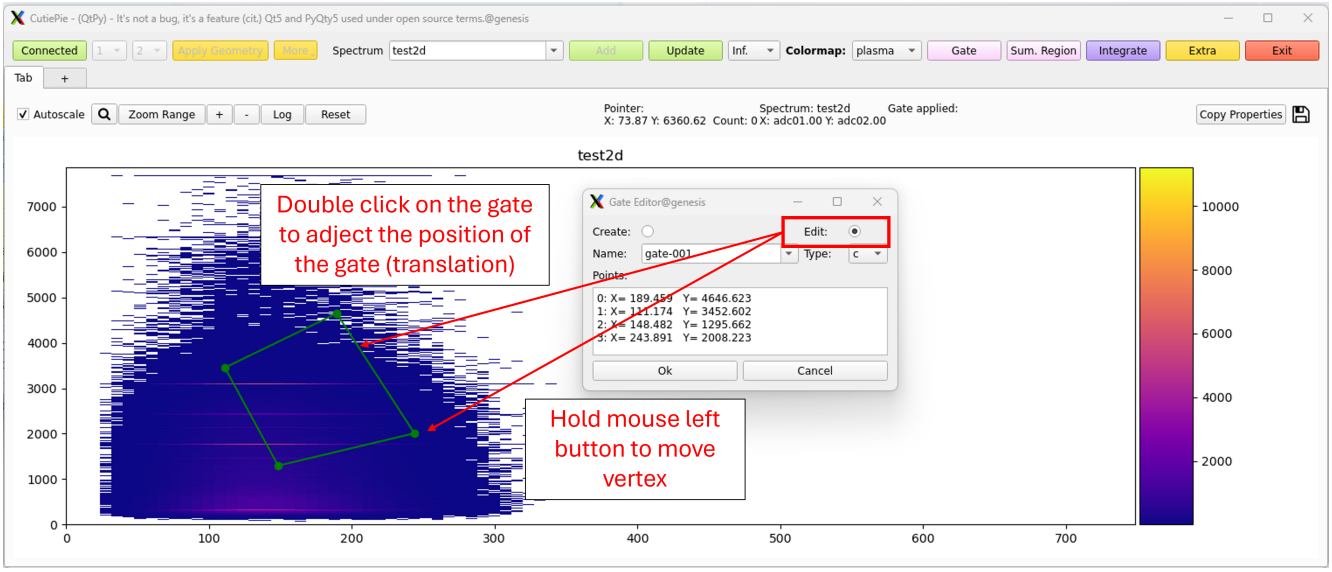
Naming rules: if the name already exists you'll be asked to confirm **overwrite** (Yes / Cancel). Names cannot contain _-_. **Cancel** discards a gate in progress and closes the editor.

Gate display options live in the **Extra** window's **Other options** panel (not in the Gate Editor):

- **Gate annotation** — label each gate on the plot.
- **Hide gate** — hide all drawn gates.
- **Disable gate edition** — locks gates so you can't accidentally edit them.

Editing an existing gate:

1. In the Gate Editor, click the **Edit** radio.
2. Pick a gate from the **Name** dropdown — it redraws in green with markers, and the **Points** box becomes editable.
3. To reshape it on the plot, hold **Alt**: **Alt + left-click** inserts a vertex, **Alt + right-click** deletes one. (You can also edit the numbers in the Points box directly.)



*Editing: switch the **Edit** radio and pick a gate — it redraws in green with draggable vertices. Double-click the gate to translate it; hold left-click on a vertex to move it.*

---

6. Summing regions & integration

A summing region is like a gate you draw purely to **add up counts** in an area.
Same drawing mechanics as gates; the payoff is the **Integrate** report.

> ■■ **Enlarge first**, exactly as with gates — double-click the pad.

Create a summing region:

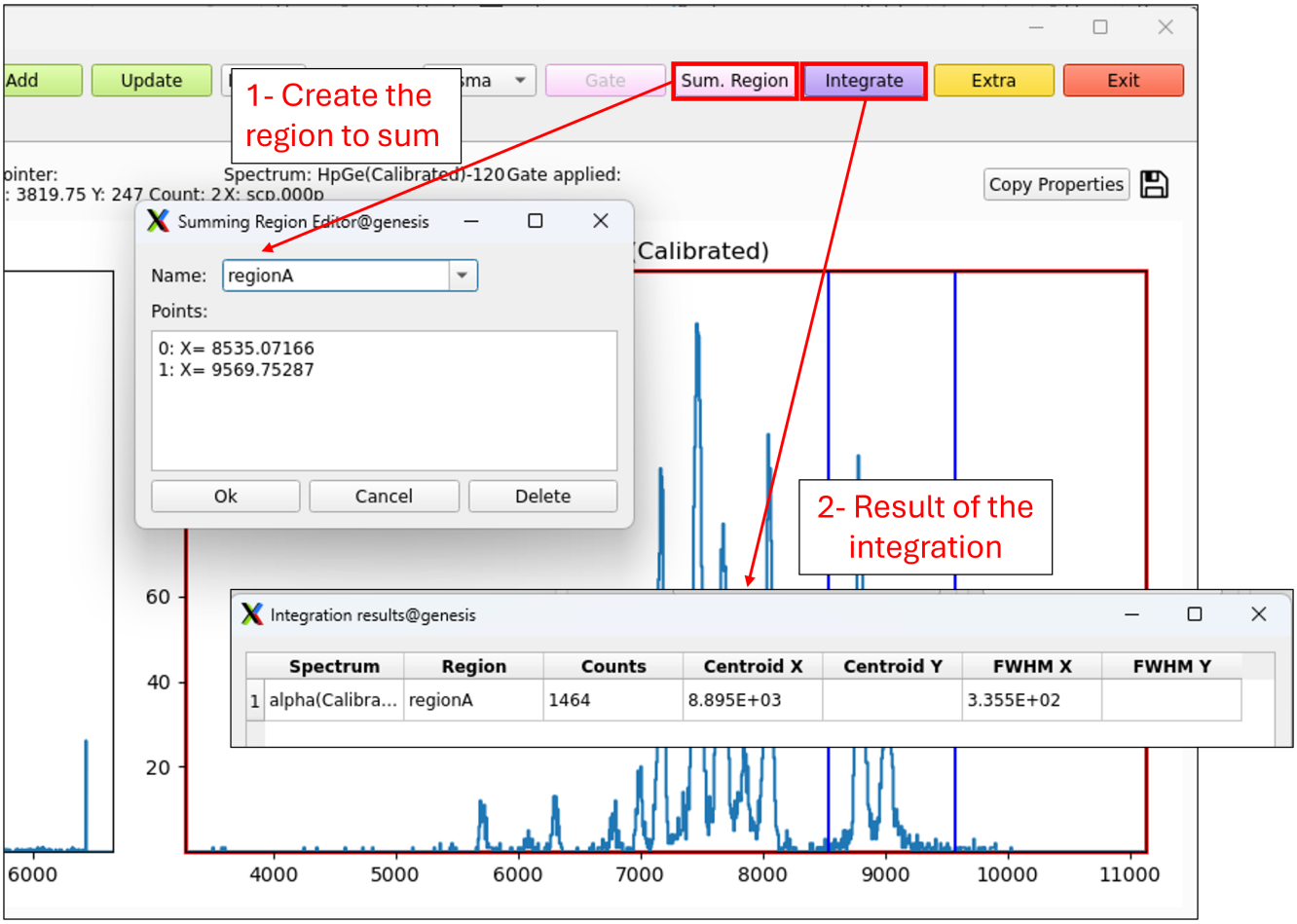
1. Double-click a pad to enlarge it.
2. Click **Sum. Region** (pink button). The **Summing Region Editor** opens with a **Name** box, a **Points** readout, and **Ok / Cancel / Delete** (there's no type dropdown and no edit mode — regions are just delete-and-redraw).
3. Draw it by left-clicking, same as a gate:
    - **1D:** two clicks for the low/high range.
    - **2D:** click polygon vertices (auto-closes); **right-click** to back up a point.
4. Click **Ok** to save it (overwrite prompt if the name exists). **Delete** removes a region by name.

Integrate:

1. With regions (and/or gates) saved on the pad, click **Integrate** (purple button, config strip).
2. The **Integration results** table opens with columns: **Spectrum, Region, Counts, Centroid X, Centroid Y, FWHM X, FWHM Y** (1D spectra fill only the X

columns).

3. **To copy the numbers out:** click any cell — the *entire* table is copied to your clipboard as tab-separated text, ready to paste into a spreadsheet or notebook.



*(1) Draw a region in the **Summing Region Editor** (Name, Points, Ok/Cancel/Delete), then (2) click **Integrate** to get the **Integration results** table — Counts, Centroid, and FWHM per region.*

---

7. Fitting a 1D spectrum

Curve fitting lives in the **Extra** window.

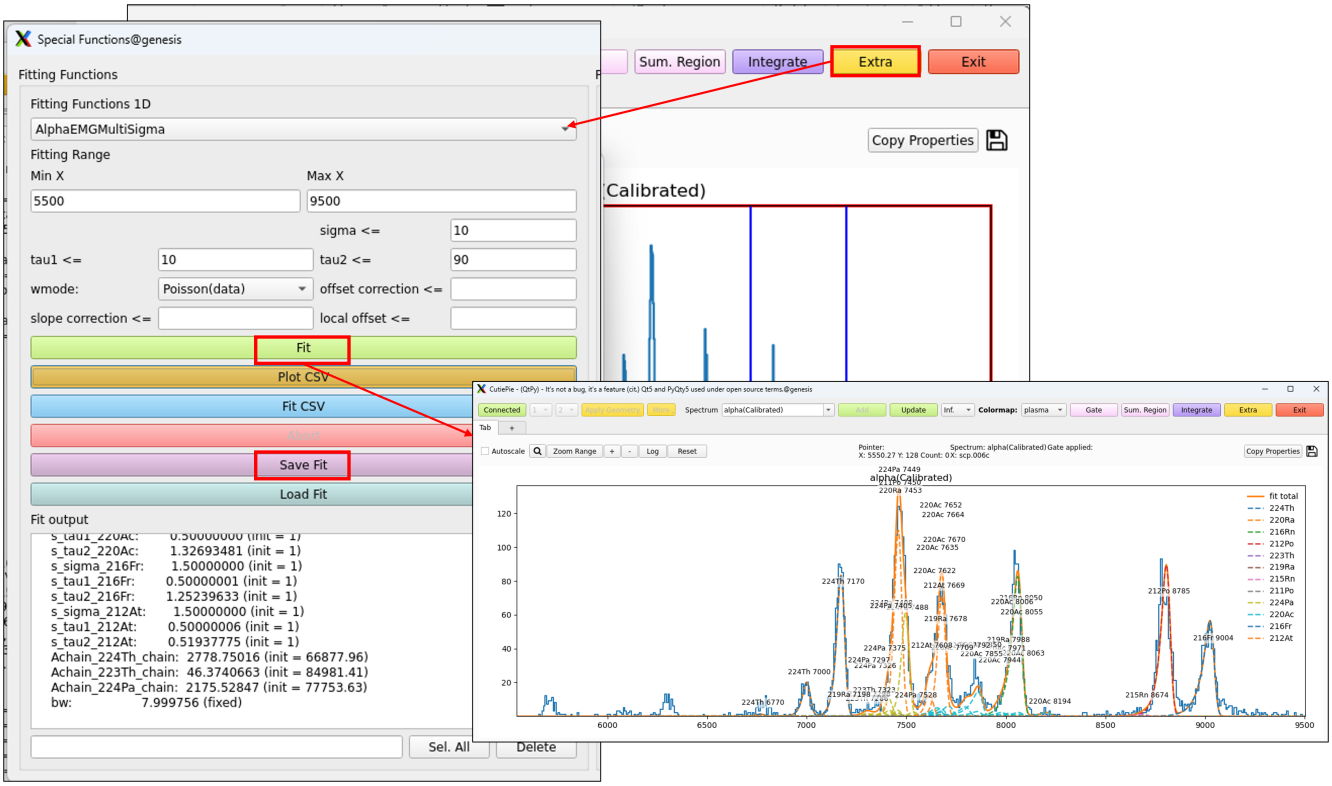
1. Click **Extra** (gold button) to open **Special Functions**. The **Fitting Functions** panel is on the left.
2. Single-click the **1D** pad you want to fit (fitting acts on the selected pad).
3. Choose a model from the fit dropdown. The everyday models are:
    - **Gauss** — a Gaussian peak.
    - **Exp** — an exponential.
    - **Pol1**, **Pol2** — straight line, parabola.
    - **G+Pol1**, **G+Pol2** — a Gaussian on a linear / quadratic background.

(The dropdown also lists AlphaEMG… models — specialist nuclear line-shapes used for alpha-decay work. They're beyond this tutorial; ignore them unless your project needs them.)

4. Set the fit window: **Min X** and **Max X**. (If you enter Min > Max, CutiePie warns you and just swaps them.)

5. *(Optional)* enter starting guesses in the **p0, p1, …** fields — these relabel to match the model (for Gauss: **A** amplitude, μ mean, σ width). Leave them blank to let CutiePie seed them automatically.

6. Click **Fit**. The fit curve draws on the pad, results append to the **Fit output** box, and a separate **Fit results — <model> : <spectrum>** window opens with the full parameters.



*The **Fitting Functions** panel (model, range, bounds, **Fit** / **Save Fit** / **Load Fit**, Fit output) and the drawn fit — here an AlphaEMGMultiSigma fit with the total curve plus per-isotope components in the legend.*

Other fit controls:

• **Abort** (red) — stop a long-running fit. The output notes *"[abort] Fit stopped by user."*

• **Sel. All** then **Delete** — **Sel. All** fills the index box with the numbers of all fit lines on the pad; **Delete** removes the ones you list. Deleting a fit removes the *whole* fit at once — for a multi-line fit that means the total curve, every component curve, its peak labels, and the legend all disappear together.

• **Plot CSV** / **Fit CSV** — plot an external `x, y` CSV file, then fit it with the same range fields. (Fitting with no CSV loaded warns *"No CSV loaded"*.)

• Fitting a **2D** spectrum is not supported — you'll get *"Sorry 2D fitting is not implemented yet."*

Saving a fit and re-using it on another plot:

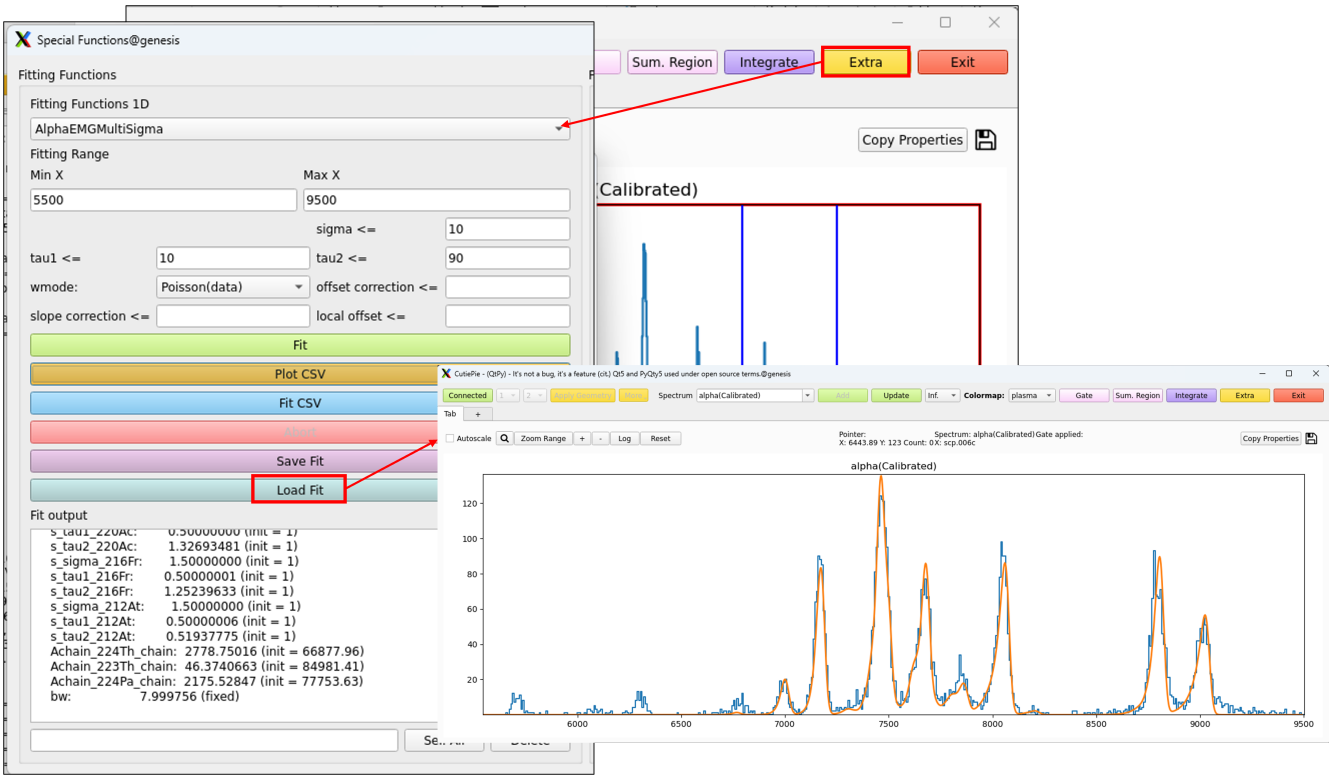
Once you've run a fit you can save its curve and draw it back later — handy for comparing a reference fit against a fresh spectrum.

• **Save Fit** (mauve) — writes the *last* fit's curve to a `.csv` (a file dialog lets you pick the name; `.csv` is added if you leave it off). For most models this is a three-column `x, y_data, y_fit` file — one row per histogram bin, giving the bin centre, the raw counts, and the fitted value — so you can compare fit against data straight in Excel. (If the raw data isn't available it falls back to a two-column `x, y_total` file.) Clicking Save Fit before you've run any fit warns *"No fit to save."*

• **Load Fit** (teal) — select the pad you want to draw on, click Load Fit, and choose a saved file. The curve is drawn on that pad as a normal fit line, so **Sel. All** / **Delete** manage it like any other fit.

• *Heads-up:* Load Fit does **not** rescale the pad — it draws inside the current view. If you don't see the curve, it may be sitting outside your zoom; reset or widen the view (Section 4).

• Loading with no pad selected warns *"No plot selected."*



Reloading a saved fit: ***Load Fit*** *draws the saved curve back onto the selected pad (here the total curve overlaid on the spectrum).*

Saving all components (AlphaEMGMultiSigma). When you save an AlphaEMGMultiSigma fit, the file keeps *every* component — the total, one summed column per isotope, **and one column per individual peak** — organised by decay chain. The `.csv` is built to be readable and Excel-friendly:

• A **per-peak parameter table** near the top — a plain CSV block (`chain,isotope,E_keV,A,mu,sigma,tau1,tau2,eta`, one row per peak) that imports straight into Excel/pandas as a table.

• Below it, the sampled **data table**, one row per histogram bin. A `y_data` column carries the raw counts alongside the fit columns, which are named by their place in the hierarchy — `fit total`, then `<chain>/<isotope>` for an isotope sum and `<chain>/<isotope>/<energy>` for a single peak — so you can

chart each peak, or the fit against the data, directly in Excel.

On **Load Fit**, every component is drawn at once (total solid, isotope sums dashed, individual peaks dotted — all one deletable fit) and a **Loaded fit components** panel opens. The panel **stays open** so you can add or remove components live:

- It's a tree with *fit total* on top, then each decay **chain**, its **isotopes** (the summed curve), and each isotope's individual **peaks** indented beneath — everything ticked.
- **Untick** a box to remove that curve from the pad; **re-tick** it to add it back — the plot updates immediately, no re-loading.
- Ticking/unticking an isotope's **(sum)** box toggles all of its peaks at once.
- **All** / **None** add or remove everything; **Close** leaves the currently shown curves on the pad.

A new **Load Fit** (or running a new fit) closes the panel and starts fresh.

---

8. Finding peaks

The **Peak Finder** panel in the **Extra** window searches the visible range for peaks in one pass and marks everything it finds.

Peak Finder (scan)

An automatic peak finder: it searches the visible X range in one pass and marks everything it finds, rather than fitting one peak per click.

1. Click **Extra**, then find the **Peak Finder** panel. Select a **1D** pad.
2. Set **Peak Width (in bins)** to roughly the FWHM of the peaks you expect, in bins.
3. Pick an **Algorithm** (or keep the default):
   - **Mariscotti (2nd difference)** — the default. The classic nuclear-spectroscopy search; it cancels smooth backgrounds, so it is the best choice when your peaks sit on a slope (e.g. an exponential continuum). Good everywhere else too.
   - **Smoothed (Savitzky-Golay)** — smooths the counts before searching and requires a peak to beat the statistical noise; the best pick for low-statistics spectra. Accepts peaks down to half the entered width.
   - **Raw counts (legacy)** — the original CutiePie search, run directly on the raw counts. Kept for continuity with old results; on noisy spectra it reports many spurious "peaks".
   - **Raw counts (legacy, fast)** — identical results to the legacy search, just faster on very large spectra.

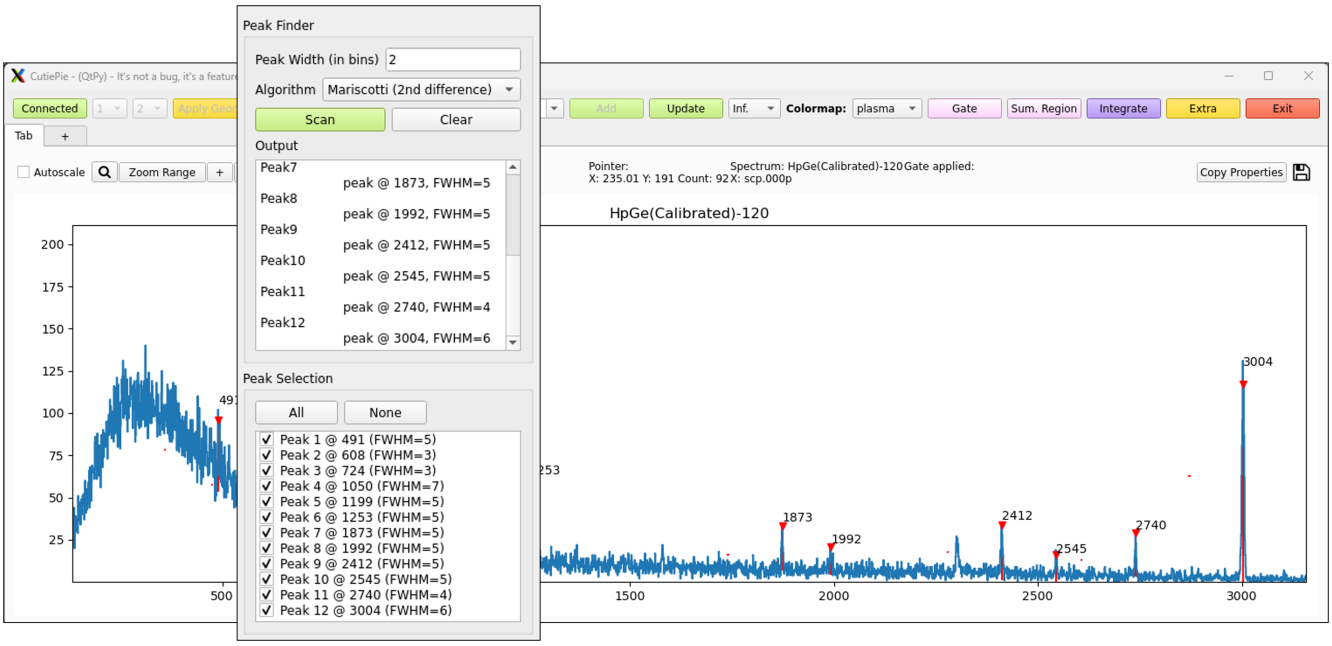
> *Gotcha (legacy algorithms only):* for the two **Raw counts** entries the
> width acts as a strict minimum-width filter — with the default **20** on
> narrow peaks the scan finds **nothing**; lower the number and scan again.
> Mariscotti and Smoothed are forgiving about the width setting.

4. Click **Scan**. Found peaks are marked in red (a v marker plus guide lines), and the **Output** box lists each one:

```

Peak1

peak @ <position>, FWHM=<width>

```

5. Only peaks in the **currently visible X range** are found — zoom into a region first (Section 4) to focus the scan there.

Managing the markers: every found peak (no limit on how many) appears as a checkable row in the **Peak Selection** list, labeled `Peak N @ <position> (FWHM=<width>)` — untick a row to hide that peak's markers, re-tick to show it again. **All** / **None** check or uncheck every row at once. **Clear** wipes the markers, the list, and the Output box.



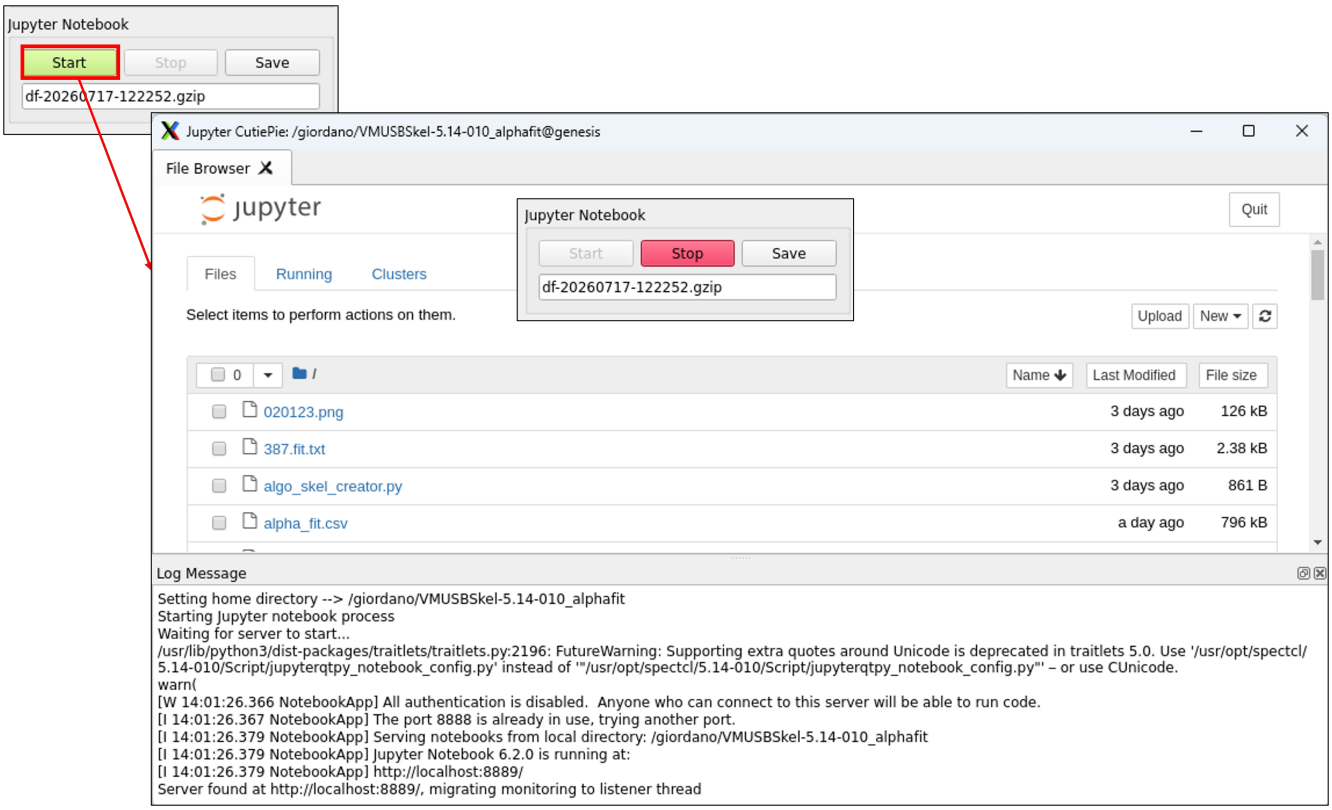
*The **Peak Finder** panel — Peak Width, Algorithm, **Scan** / **Clear**, the **Output** list, and the checkable **Peak Selection** list — with each found peak marked (red ∨) and labelled on the spectrum.*

---

9. Exporting data (Jupyter / pandas)

CutiePie can dump every displayed spectrum to a compressed CSV for analysis in Python. This is in the **Extra** window's **Jupyter Notebook** panel.

- **Filename field** — pre-filled with `df-<date-time>.gzip`; edit it to choose where the file goes.
- **Save** — writes the CSV **right now**, without launching anything. Use this when you just want the data file.
- **Start** — writes the CSV *and* launches a Jupyter notebook server with a docked browser window. **Stop** shuts the server down again.



*The **Jupyter Notebook** panel (**Start** / **Stop** / **Save** and the `df-…gzip` filename); **Start** opens the docked Jupyter file browser, with the server log at the bottom.*

What you get: two files that belong together — the CSV you named, and a counts file beside it with `-counts.npz` in place of the extension. Saving `df-20260711-101500.gzip` also writes `df-20260711-101500-counts.npz`. Copy or move them as a pair.

The CSV has one row per spectrum, with columns:

```
name, dim, binx, minx, maxx, biny, miny, maxy,
xunderflow, yunderflow, data, xoverflow, yoverflow, parameters, type
```

`xunderflow`/`yunderflow` and `xoverflow`/`yoverflow` are the under/over-range statistics (blank/`NaN` where the server doesn't report them, e.g. the y-values for a 1D spectrum). The `data` column holds a short key such as `s0` pointing into the counts file — the counts themselves are no longer written into the CSV, which is what makes a large 2-D export take under a second instead of several.

Read it back in Python:

```python
import sys
sys.path.append("<path to>/PyQtGUI/gui")
from services.dataframe_export import read_spectrum_export

df = read_spectrum_export("df-20260711-101500.gzip")
print(df[["name", "dim", "xunderflow", "xoverflow"]])
counts = df["data"].iloc[0] # a real NumPy array, 1-D or 2-D
```

```

`read_spectrum_export` opens both files and puts the counts back in the `data` column as NumPy arrays. It also opens **older** exports, where the counts were inside the CSV, so old files keep working. If the counts file has been renamed or left behind, it says so — `counts sidecar missing for …` — rather than failing quietly.

**Plotting a spectrum.** The stored array has `binx + 1` entries: one more than the number of bins. CutiePie's own fitting code drops the first entry and pairs the rest with bin centres, and that is what you want for analysis:

```python
import numpy as np
import matplotlib.pyplot as plt

row = df[df["name"] == "my_spectrum"].iloc[0]
edges = np.linspace(row["minx"], row["maxx"], int(row["binx"]) + 1)
centres = edges[:-1] + 0.5 * np.diff(edges)
plt.step(centres, row["data"][1:], where="mid")
```

A 2-D spectrum goes straight into `imshow`:

```python
plt.imshow(row["data"], origin="lower", aspect="auto",
extent=[row["minx"], row["maxx"], row["miny"], row["maxy"]])
```

Without importing anything. The counts file is a plain `.npz`, so you can read the pair by hand:

```python
import numpy as np
import pandas as pd

meta = pd.read_csv("df-20260711-101500.gzip", compression="gzip")
z = np.load("df-20260711-101500-counts.npz")
counts = z[meta["data"].iloc[0]] # the cell holds the key, e.g. "s0"
```

The keys are **positional** — `s0`, `s1`, … in CSV row order, not spectrum names. That is deliberate: spectrum names carry spaces, brackets and slashes, and a slash would turn an archive entry into a path. Always look the key up from the CSV's `data` column rather than guessing it from a name.

> **If you have a notebook that does** `pd.read_csv(...)` **on an export:** it still
> runs, but its `data` column now holds the `s0`-style keys rather than counts.
> Switch that line to `read_spectrum_export` to get the arrays back. Everything
> else about the table is unchanged.

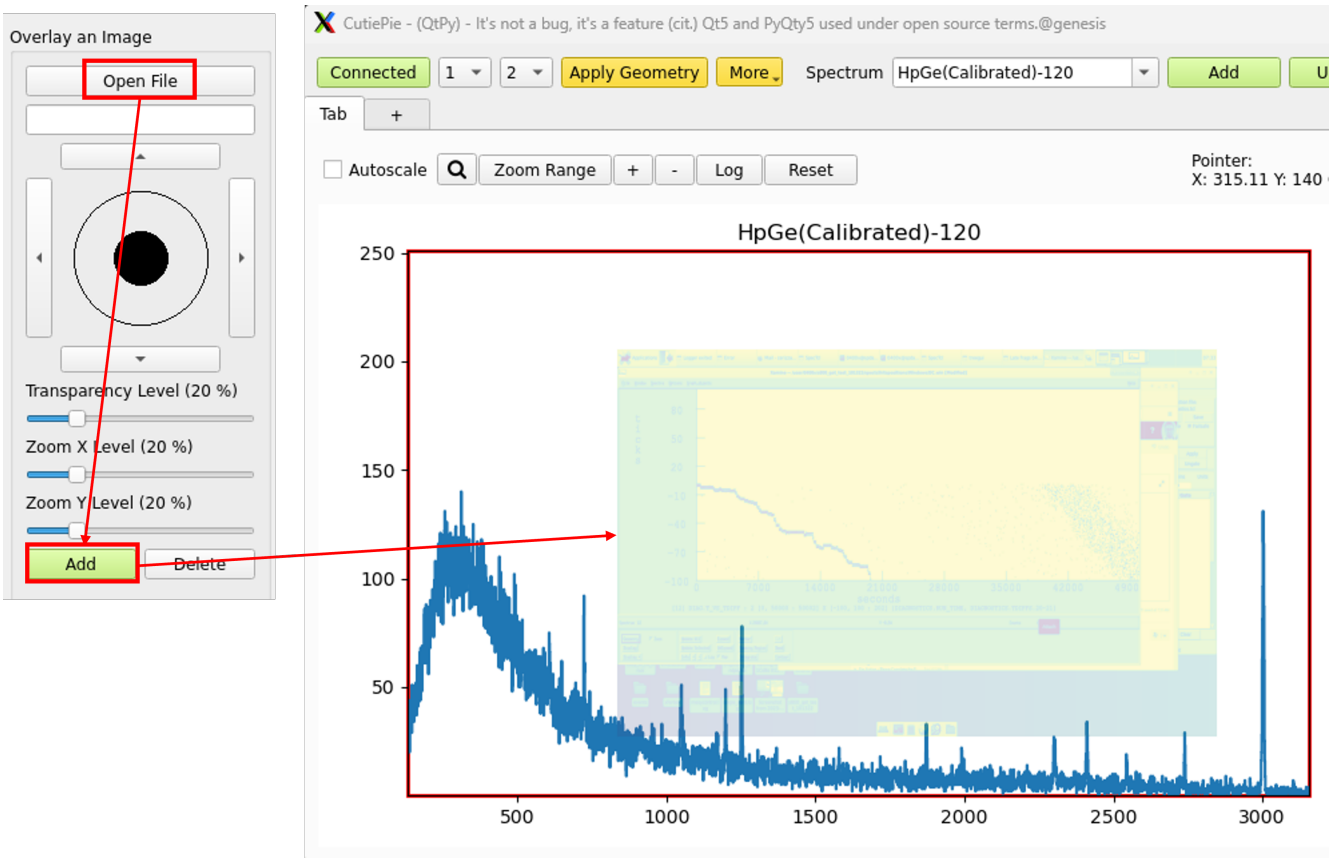
---

10. Overlaying a reference image (LISE)

You can lay a reference image (e.g. a LISE prediction) over a pad to compare it
```

with your data. This is in the **Extra** window's **Overlay an Image** panel.

1. Click **Open File**, pick a `.png` or `.jpg`. Its path appears in the field.
2. Single-click the pad you want it on, then click **Add** — the image appears over that pad.
3. **Position it:**
   - Drag the **Joystick** widget to move the image (bigger drag = bigger step).
   - Use the four fine **arrow** buttons for small, precise nudges.
4. **Adjust it** with the sliders: **Transparency**, **Zoom X**, **Zoom Y**.
5. **Delete** removes the overlay.



*The **Overlay an Image** panel — **Open File**, the joystick and arrow nudges, and the Transparency / Zoom X / Zoom Y sliders — with a reference image overlaid on the spectrum after **Add**.*

---

11. Quick reference

Most-used buttons:

Button (where)	Does
Connect (config strip)	open the connection dialog / show status by color
Apply Geometry	build the rows × columns pad grid
More	Save/Load Geometry, Save/Load All Tabs
Add	put the chosen spectrum in the selected pad
refresh-interval dropdown	how often pads auto-refresh (**Inf. = never**)
Gate / **Sum. Region**	draw a gate / summing region (needs an enlarged pad)
Integrate	tabulate counts in gates & regions

Button (where)	Does
Extra	open the Fitting / Peak / Jupyter / Overlay window
pad **Log / Reset**	log scale / reset view (right-click for "all")

Common gotchas:

• **Gates and summing regions need an *enlarged* pad** — double-click the pad first (the Gate button is greyed out otherwise).

• **No live counts?** The refresh interval is on **Inf.** by default — pick a real interval.

• **Peak scan finds nothing?** With a **Raw counts (legacy)** algorithm the **Peak Width** default of 20 bins is too wide for narrow peaks — lower it, or switch to the default **Mariscotti (2nd difference)** algorithm.

• **Can't delete a tab?** You can't delete the tab you're on — switch away first.

• **Gates/regions need a live SpecTcl connection** (green Connect button).

---

12. Appendix

• **Exit** (config strip, red) closes CutiePie. If a Jupyter server is running it is shut down for you.

• **Saved files:** geometry layouts are `.win` files; Jupyter data dumps are `.gzip` CSVs; saved fits (Section 7's **Save Fit**) are `.csv` files — an `x, y_data, y_fit` table (bin centre, raw counts, fit), or, for AlphaEMGMultiSigma, a multi-column file with one column per isotope plus a `# meta` header. All saved to wherever you point the file dialogs / filename field.

• **If something looks wrong:** confirm the Connect button is green; confirm your refresh interval isn't on Inf.; and remember the enlarge-first rule for gates/regions. If a label in this guide doesn't match your screen, your install may be a different version — check with your instructor.