MLIGLON / RESEARCH SOFTWARE

INTEGER GATE LOGIC LABORATORY

COMPREHENSIVE USER GUIDE

Version 1.3.0 | WordPress plugin | WebGL 3D explorer



INCLUDES

- 16 Boolean + 4 graded presets
- Five-region activation guide
- Exact & Fixed INT arithmetic
- Six-panel operator reference
- 3D inspection and PNG export
- 10 experiments + troubleshooting

SOURCE-VERIFIED TECHNICAL EDITION · 08 NOV 2026

Prepared from the v1.3.0 plugin distribution. Intended for visitors, instructors, software engineers, researchers, and WordPress administrators.

How to use this guide

This reference documents the features that ship in MLiglon Interactive IGL Laboratory version 1.3.0. The descriptions and formulas were checked against the bundled PHP, HTML, JavaScript model, WebGL renderer, README, and test report. Screenshots were rendered from the supplied standalone v1.3.0 application and the WebGL preview.

> **Scientific scope and interpretation**
>
> This is an educational, client-side demonstrator for a single two-input IGL node. It is not a multi-layer network simulator, full chain-isolation trainer, or a source of hardware performance measurements. Its visual surfaces are sampled renderings of the numerical model.

Reading paths: New visitors can start with Sections 1–4 and the guided experiments. Technical readers should study the activation function, quantization equations, and truth-table definitions before using the 3D explorer. Site administrators can move directly to installation, compatibility, and troubleshooting. The appendices provide reference tables and test provenance.

Documentation conventions

Notation	Meaning
X1, X2	Two input signals; user controls range from 0 to 1.
W1, W2	Learnable-style weights used here as editable parameters.
B	Editable bias (offset) in the weighted sum.
P	Positive activation scale, chosen from seven discrete values.
S	Weighted sum entering the five-region activation.
Y / Y/P	Raw output Y; normalized response Y/P between 0 and 1.
Exact / real	Direct evaluation of fractional input and parameter values.
Fixed INT	Quantized integer calculations at activation scale P.

The labels “Boolean”, “graded”, and “exact” describe responses to the four binary input combinations in the truth table. Unless stated otherwise, continuous plots show outputs across all combinations of real inputs X1 and X2 in [0,1].

What is new in v1.3.0

- Preserves the v1.2.0 laboratory: 20 presets, presentation controls, and the original panel 5 Canvas graph.
- Adds a resizable WebGL explorer with orbit, zoom, pan, point inspection, PNG export, six palettes, four mesh densities, projections, slices, axes, labels and wireframe. No remote 3D framework is needed; IBM Plex font loading is optional.

Contents

How to use this guide 2

Documentation conventions . . . 2

What is new in v1.3.0 . . . 2

PART I | Purpose, installation and orientation 5

1. Why the IGL Laboratory exists . . . 5
2. System requirements and operating environment . . . 6
3. Installing or upgrading the WordPress plugin . . . 6
4. Interface tour: from top toolbar to six panels . . . 7
5. First session: a ten-minute route . . . 9

PART II | Mathematics and interpretation 10

6. The two-input IGL node . . . 10
7. The patented five-region activation shape . . . 10
8. Boolean truth tables: why there are exactly 16 . . . 11
9. The four graded example presets . . . 13
10. Fixed INT: the implemented arithmetic path . . . 13
11. Activation scale P, slider ranges and rescaling . . . 14

PART III | Complete interface reference 15

12. Display toolbar, presets, and learning guide . . . 15
13. Panel 1: node parameters . . . 15
14. Panel 2: five-region activation plot . . . 17
15. Panel 3: live truth table and classification . . . 17
16. Panel 4: arithmetic trace . . . 18
17. Panel 5: the original projected 3D response surface . . . 18
18. Panel 6: illustrative local candidate search . . . 19

PART IV | The WebGL interactive 3D explorer 20

19. Opening, positioning and closing the explorer . . . 20
20. Camera controls: orbit, pan, zoom and reset . . . 21
21. Projection, color palettes and surface resolution . . . 21
22. WebGL visibility controls . . . 22
23. Interactive point inspection and pinning . . . 23
24. Cross-section tools and spatial interpretation . . . 23

25. Resizing, fullscreen, export and model synchronization . . . 24
26. Performance and rendering considerations . . . 24

PART V | Guided experiments and lesson plans 25

27. Experiment 1: reproduce XOR with one node . . . 25
28. Experiment 2: exact XNOR and the effect of bias . . . 25
29. Experiment 3: AND, NAND, NOR and OR . . . 26
30. Experiment 4: asymmetry versus symmetry . . . 26
31. Experiment 5: inspect graded logic and continuous inputs . . . 26
32. Experiment 6: Real versus Fixed INT . . . 27
33. Experiment 7: study activation scale and parameter rescaling . . . 27
34. Experiment 8: greedy search toward a different gate . . . 27
35. Experiment 9: use cross-sections to find non-monotonic behavior . . . 28
36. Experiment 10: prepare a reproducible technical observation . . . 28

PART VI | Accessibility, troubleshooting and administration 28

37. Accessibility and presentation tips . . . 28
38. Troubleshooting: the main laboratory . . . 29
39. Troubleshooting: the WebGL explorer . . . 30
40. Security, privacy and network considerations . . . 30
41. WordPress developer and file-structure reference . . . 31
42. Verification results and known gaps . . . 31
43. Frequently asked questions . . . 32

Appendices | Quick reference and reproducibility 33

Appendix A. Complete gate matrix . . . 33
Appendix B. Default settings and saved preferences . . . 34
Appendix C. Symbol and term glossary . . . 35
Appendix D. Numerical verification checklist . . . 35
Appendix E. Source and scope of this manual . . . 36

The headings and page references are generated from the finished document. Most headings are navigable via the PDF outline/bookmark sidebar.

PART I | Purpose, installation and orientation

1. Why the IGL Laboratory exists

The laboratory gives visitors an immediate way to investigate the core behavior of Integer Gate Logic rather than relying on a static description. A user can place four binary input combinations into different activation regions by changing only two weights and a bias, watch the normalized outputs change, and inspect the full response over nonbinary inputs. The intention is explanatory and exploratory: to turn the architecture into something visible, testable, and discussable.

For software engineers, the most useful characteristic is the connected view of parameters, arithmetic, and behavior. A change to W1, W2, or B updates the weighted sum, activation curve, truth-table rows, step-by-step arithmetic, and response surfaces. For students, this is a concrete introduction to non-monotonic piecewise functions, signed weights, discrete logic behavior, and quantization. For evaluators, it provides an accessible starting point for framing more rigorous questions.

What it can demonstrate

- All 16 possible truth tables of two Boolean inputs, each represented exactly at the four binary input corners by a preset node.
- Four examples of graded responses that do not exactly reproduce a Boolean truth table.
- The five-region activation function and its relationship to weighted sums.
- Differences between direct real-number evaluation and a specified scaled-integer arithmetic model.
- How the input-response surface changes with model parameters and where its activation regions lie.
- A simple derivative-free local candidate search against a selectable Boolean truth table.

What it does not establish

- It does not demonstrate training of a deep or convolutional IGL network.
- It does not reproduce the complete patented chain-isolation optimization process.
- It provides no measured claims about test accuracy, power, latency, memory, or benchmark speedups.
- It does not prove that a higher-resolution triangle mesh improves the mathematical accuracy of the single-node evaluator.

> **Suggested visitor narrative**
>
> Begin with XOR. Examine why its four truth-table outputs are 0, 1, 1, 0. Move a fractional input and inspect the grade between the Boolean corners. Then use Fixed INT, open WebGL, and compare the effect of sampling and quantization.

2. System requirements and operating environment

Component	Requirement / recommendation
Website	WordPress installation with permission to install and activate a plugin.
Page	A published page containing exactly one [igl_demo] shortcode.
Browser	Modern JavaScript, SVG, Canvas 2D, and pointer/keyboard event support.
Optional 3D	WebGL 1 compatible browser, GPU/driver or software rendering support.
Internet	Required to load the website; Google Fonts is optional for IBM Plex presentation.
Storage	Local browser storage, if permitted, retains theme/font/font-size only.
Input methods	Mouse, keyboard; touch input including pinch in the WebGL viewer.
Export	Browser clipboard and local file-download capabilities, subject to browser permissions.

The core laboratory and the original 2D-projected chart remain available even if the browser cannot create a WebGL context. WebGL features can be blocked by security policies, graphics drivers, browser settings, remote desktops, privacy extensions, or browser-level GPU restrictions. Unsupported behavior should be checked on the specific deployment environment.

3. Installing or upgrading the WordPress plugin

1. Back up the existing WordPress site, or install on a staging copy.
2. Open Dashboard → Plugins → Add New Plugin → Upload Plugin.
3. Choose the file mliglon-igl-demo-1.3.0.zip and upload it.
4. If WordPress identifies a previous version, select the option to replace it with the uploaded archive.
5. Confirm that “MLiglon Interactive IGL Laboratory” version 1.3.0 is active.
6. Verify that the Demo page contains the [igl_demo] shortcode; the existing shortcode is unchanged.
7. Purge WordPress, performance-plugin, reverse-proxy, and CDN caches where applicable; hard-refresh the page.
8. Load a preset and verify all six panels update. Then open panel 5 → Explore interactive 3D and confirm the WebGL surface renders.

Shortcode

[igl_demo] — Insert into a Shortcode block or other WordPress location that evaluates shortcodes. The distribution intentionally allows only one laboratory instance per page.

If the plugin appears unchanged after uploading

Check the installed plugin version, clear any HTML/JavaScript caches, force-refresh the browser, and review the page source or browser network requests for the 1.3.0 assets. The PHP bootstrap registers assets and can preload them when it encounters the shortcode in post content or when the current page slug is “demo”. A duplicate copy of the shortcode yields a one-instance warning rather than two running laboratories.

Rollback strategy

If a theme, plugin, or browser incompatibility is found after upgrading, restore your backed-up plugin installation and remove any stale caches. The normal laboratory controls and shortcode were retained from 1.2.0, reducing migration requirements; a staging test remains the appropriate way to verify compatibility with local WordPress themes and security policies.

4. Interface tour: from top toolbar to six panels

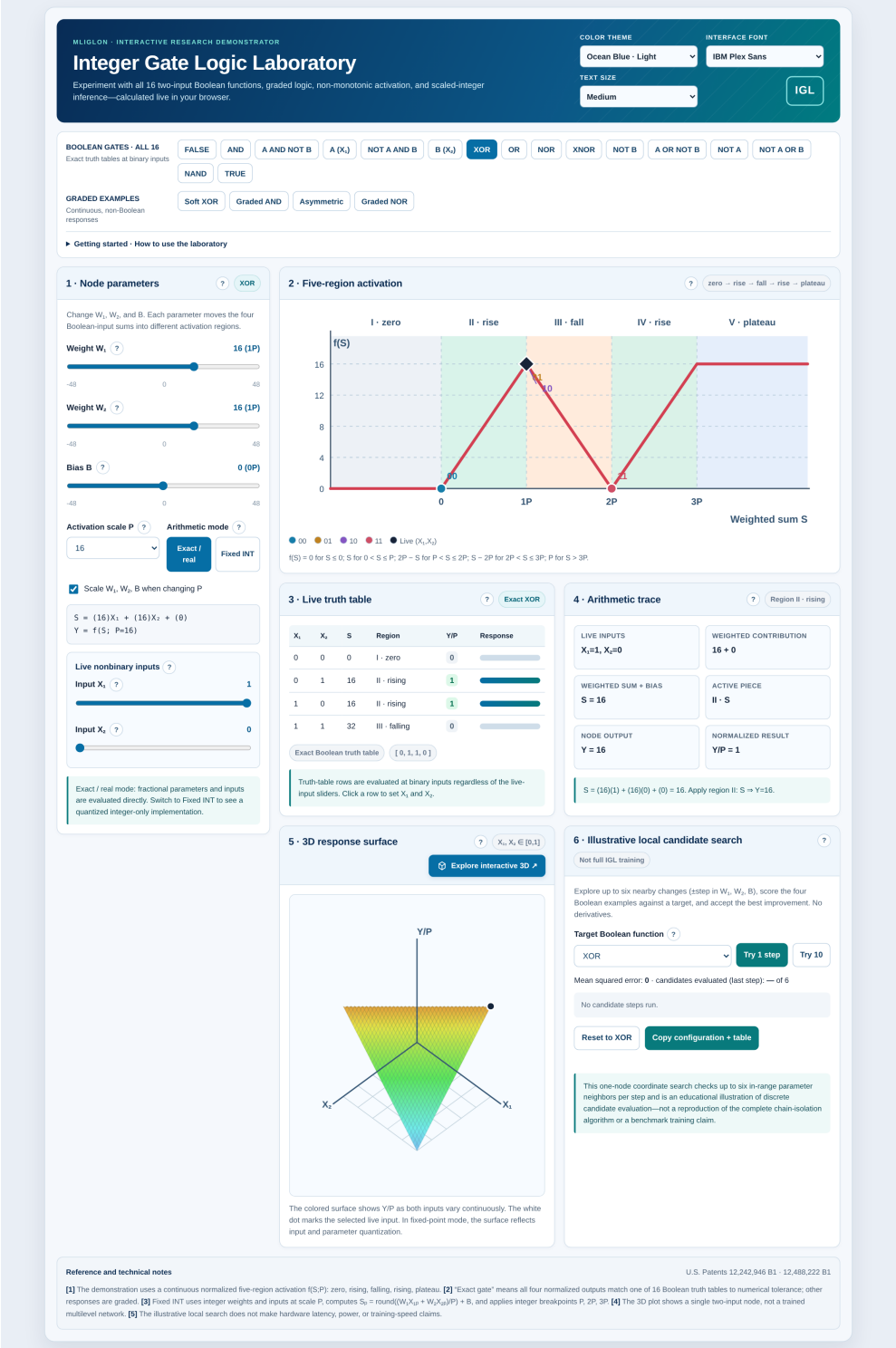


Figure 1. Actual v1.3.0 laboratory captured from the standalone application. The banner, 20 presets, six numbered panels, and technical notes form one linked workspace.

The banner contains visual preferences. Beneath it, the preset strip separates exact Boolean gates from graded examples and includes a collapsible “Getting started” guide. The numbered sections serve different analytic purposes: panel 1 is the parameter editor; panel 2 plots the five-region transfer curve; panel 3 shows four binary truth-table results; panel 4 traces one live evaluation; panel 5 shows a continuous response surface; and panel 6 tries a small local search against a chosen target gate.

Area	Primary use
Display toolbar	Color theme, interface font, and text size.
Boolean presets	Select one of 16 exact two-input truth tables.
Graded presets	Select one of four continuous non-Boolean examples.
1 · Node parameters	Edit W1, W2, B, P, mode, and live inputs.
2 · Five-region activation	Read f(S), regions, and point markers.
3 · Live truth table	Compare binary-corner outputs and exact/nearest gate.
4 · Arithmetic trace	Follow S, active piece, Y, and normalized result.
5 · 3D response surface	View full input/output behavior; open WebGL viewer.
6 · Local search	Inspect a pedagogical coordinate-search attempt.

5. First session: a ten-minute route

1. Open the Demo page and confirm that XOR is initially selected. The initial values are W1=16, W2=16, B=0, P=16, Exact/real mode.
2. Look at the truth table: Y/P should be 0, 1, 1, 0 for rows (0,0), (0,1), (1,0), (1,1).
3. Click the row (1,0). Confirm that the live X1 and X2 sliders and arithmetic trace update.
4. Move X1 to 0.45 and X2 to 0.40. The real-mode weighted sum for the default XOR weights is 13.6; output Y/P=0.85.
5. Look at panel 5 to see how the white live marker moves on the surface.
6. Click “Explore interactive 3D”. Rotate the surface and change the color palette.
7. Hover on the mesh to read X1, X2, S, Y, Y/P, and region; click a point to pin it.
8. Close the explorer and return to panel 1. Switch to Fixed INT and observe input snapping and integer computations.
9. Try XNOR or NAND and compare its truth table and surface with XOR.
10. Return to XOR with “Reset to XOR” in panel 6.

The example X1=0.45, X2=0.40 assumes the default XOR weights. Because 0.45+0.40=0.85, S=16(0.85)=13.6; the activation is in region II, where Y=S.

PART II | Mathematics and interpretation

6. The two-input IGL node

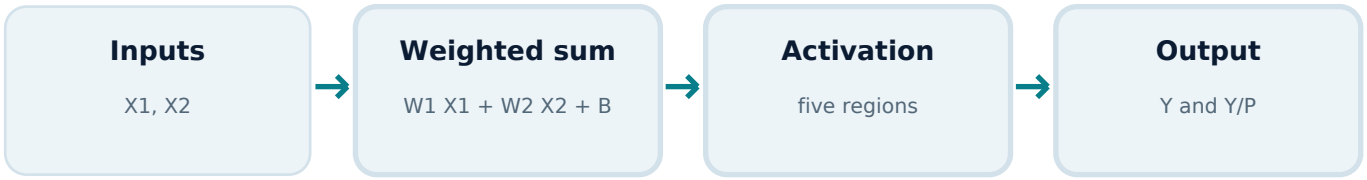


Figure 2. The shared signal path is input values → weighted sum → five-region activation → raw and normalized output. The same numerical engine feeds the truth table, trace, Canvas chart, WebGL surface, and search score.

> **Real arithmetic**
> S = W1 × X1 + W2 × X2 + B. Then Y = f(S; P), and the displayed normalized output is Y/P.

The weights are signed, the bias may be positive or negative, and P is a positive scale. In the interface, the weights and bias are bounded to the interval −3P through +3P; this UI range is a laboratory design choice, not a theorem about all IGL implementations. Live inputs occupy the interval [0,1].

Outputs for the standard truth table are evaluated only at the four binary corners. The four rows use the fixed ordering (X1,X2)=(0,0),(0,1),(1,0),(1,1). These four points are independent of whichever fractional values are currently selected on the live sliders.

Normalized outputs

The activation produces Y in [0,P]. Dividing by P gives the normalized response Y/P in [0,1]. Therefore, a truth-table value of 1 represents Y=P, not a literal raw activation of one when P is 16 or 64. The classification system uses the four normalized values, not only rounded labels.

7. The patented five-region activation shape

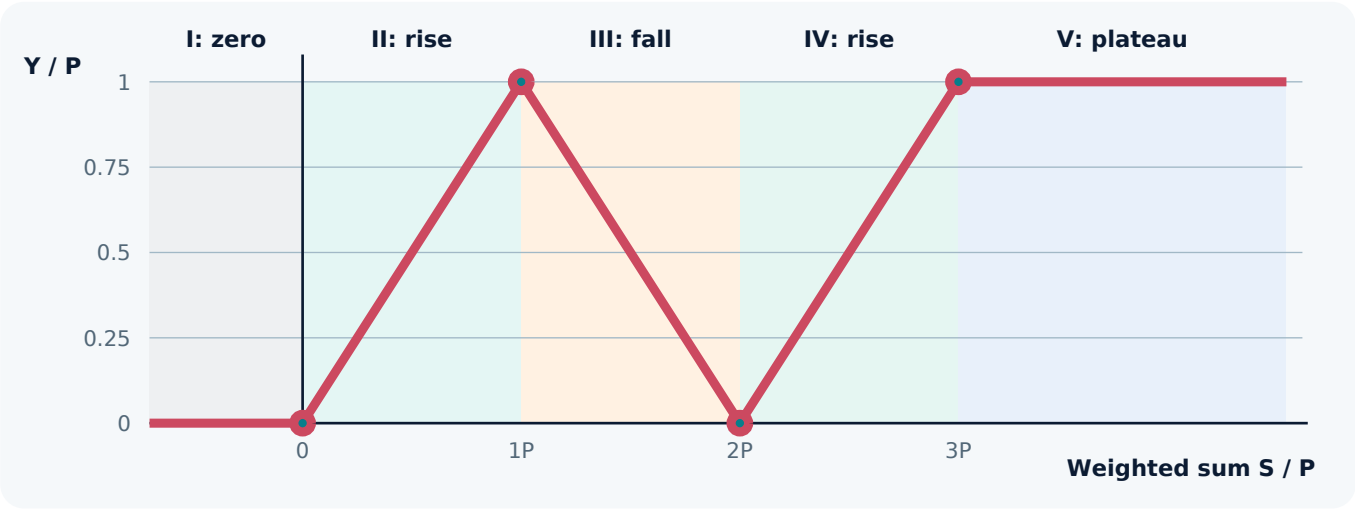


Figure 3. Vector-drawn five-region piecewise activation. The horizontal axis is weighted sum S expressed in units of P, and the vertical axis is normalized Y/P.

Region	Weighted-sum interval	Output rule
I · zero	$S \le 0$	$Y = 0$
II · rising	$0 < S \le P$	$Y = S$
III · falling	$P < S \le 2P$	$Y = 2P - S$
IV · rising	$2P < S \le 3P$	$Y = S - 2P$
V · plateau	$S > 3P$	$Y = P$

Unlike a monotonic clipped ramp, this function rises from zero to P, decreases from P back to zero, then rises again and saturates. The decreasing section is essential to the single-node XOR example: the combined input (1,1) can produce a larger weighted sum than either single-high input while its activated output returns to zero.

Boundary tests

S	Y	Y/P at P=16
−P	0	0
0	0	0
P/2	P/2	0.5
P	P	1
1.5P	0.5P	0.5
2P	0	0
2.5P	0.5P	0.5
3P	P	1
4P	P	1

The function is continuous at region boundaries but has changes in slope at its corner points. Its slope within each non-flat interval is +1, −1, or +1 (in raw Y versus raw S coordinates). Calling this transfer shape “nowhere differentiable” would be incorrect; the corners are the nondifferentiable locations.

8. Boolean truth tables: why there are exactly 16

Two Boolean inputs yield four possible input combinations. For each of these four positions, a Boolean function independently selects output zero or one. Consequently, there are 2^4 = 16 distinct two-input Boolean functions. The v1.3.0 catalog includes all 16, not just the six familiar AND/OR/XOR/NAND/NOR/XNOR names.

The bit sequence in the following table is ordered 00, 01, 10, 11. For example XOR is 0110, meaning outputs 0,1,1,0. The parameter triples below are normalized coefficients (W1/P, W2/P, B/P). Loading a preset multiplies these coefficients by the current P, with integer rounding when Fixed INT mode is active.

Function	Bits	W1/P, W2/P, B/P	Meaning
FALSE	0000	0, 0, 0	Constant false: 0 at all four input combinations.
AND	0001	1, 1, -1	1 only when both inputs are 1.
A AND NOT B	0010	1, -1, 0	1 when X_1=1 and X_2=0.
A (X_1)	0011	1, 0, 0	Copy the first input X_1.
NOT A AND B	0100	-1, 1, 0	1 when X_1=0 and X_2=1.
B (X_2)	0101	0, 1, 0	Copy the second input X_2.
XOR	0110	1, 1, 0	1 when exactly one input is 1.
OR	0111	2, 2, -1	1 when either or both inputs are 1.
NOR	1000	-1, -1, 1	1 only when both inputs are 0.
XNOR	1001	1, 1, 1	1 when the two inputs match.
NOT B	1010	0, -1, 1	Invert the second input X_2.
A OR NOT B	1011	2, 1, 1	1 when X_1=1 or X_2=0.
NOT A	1100	-1, 0, 1	Invert the first input X_1.
NOT A OR B	1101	1, 2, 1	1 when X_1=0 or X_2=1.
NAND	1110	-2, -2, 3	0 only when both inputs are 1.
TRUE	1111	0, 0, 1	Constant true: 1 at all four input combinations.

Exactness criterion

An “Exact [gate]” label appears when all four normalized outputs agree with one gate’s ideal 0/1 truth-table vector within the model’s numerical tolerance. Otherwise, the interface reports a graded/non-Boolean response and the nearest gate by mean absolute error.

A note on gate names

A corresponds to X1, and B in a gate name corresponds to the second Boolean input X2. This Boolean-input “B” should not be confused with the node’s bias parameter B. For clarity, this guide normally says X1/X2 for inputs and bias for the constant offset.

9. The four graded example presets

Preset	Normalized coefficients	Educational purpose
Soft XOR	0.75, 0.75, 0	Partially graded XOR-like response.
Graded AND	0.5, 0.5, 0	Gradual increase toward AND-like output.
Asymmetric	1, 0.5, 0	Unequal weights give asymmetric responses.
Graded NOR	-0.5, -0.5, 1	Soft decrease resembling NOR.

The graded examples deliberately give fractional outputs at one or more Boolean input corners. They are useful for examining threshold sensitivity, asymmetric inputs, and continuous behavior. “Soft XOR” is not identical to the exact XOR Boolean function, and “Graded AND” is not a hard Boolean AND until parameters are changed to an exact configuration.

Evaluating a graded example

1. Choose Soft XOR. Verify that the status badge no longer identifies an exact Boolean function.
2. Read the four normalized truth-table outputs rather than treating a response bar as a binary label.
3. Move X1 and X2 over [0,1] and observe that the continuous surface is not just a flat binary map.
4. Compare Exact/real and Fixed INT at a chosen P; watch for rounding effects.

10. Fixed INT: the implemented arithmetic path

The Fixed INT mode is a specific educational numerical model. It rounds the weight and bias values to integers, represents each live input by an integer $X_p = \mathrm{round}(\mathrm{clip}(X,0,1) \times P)$, calculates an integer dot product, divides by P and rounds, adds the integer bias, and applies the same five-region activation with integer-valued breakpoints.

> **Fixed INT model**
>
> $X1p = \mathrm{round}(P \times X1)$; $X2p = \mathrm{round}(P \times X2)$. $D = W1 \times X1p + W2 \times X2p$. $S = \mathrm{round}(D / P) + B$. $Y = f(S;P)$. Display $Y/P = Y / P$.

In the JavaScript code, rounding follows Math.round. In particular, a positive half-integer rounds upward; negative half-integers round toward positive infinity, which differs from some processor-specific “ties to even” rounding modes. This implementation detail can matter at exact boundary cases. The laboratory does not simulate fixed-width overflow, saturation, SIMD instructions, energy use, or any particular microcontroller’s rounding behavior.

Worked comparison: W1=16, W2=8, B=0, P=16, X1=0.45, X2=0.40

Step	Exact / real	Fixed INT
Input representation	X1=0.45, X2=0.40	X1p=round(7.2)=7; X2p=round(6.4)=6
Weighted sum	16(0.45)+8(0.40)=10.4	D=16(7)+8(6)=160
Scale and bias	S=10.4	S=round(160/16)+0=10
Region	II (rising)	II (rising)
Raw output Y	10.4	10
Normalized Y/P	0.65	0.625

This example isolates an important distinction: the same visible input request may be represented differently in Fixed INT mode, and therefore produce different values even when both cases occupy the same activation region. Be careful not to confuse mesh-resolution artifacts with integer quantization; the exact numerical readout is authoritative.

11. Activation scale P, slider ranges and rescaling

Item	Implemented setting
Allowed P values	1, 2, 4, 8, 16, 32, 64
Default P	16
Parameter range	Each of W1, W2, bias from −3P to +3P
Real-mode parameter slider step	P/16
Fixed-INT parameter slider step	1
Real-mode live-input step	0.05
Fixed-INT live-input step	1/P
Rescale checkbox	Checked by default: multiply parameters by new P / old P

When “Scale W1, W2, B when changing P” is checked, the laboratory attempts to preserve the ratios W1/P, W2/P and B/P. When unchecked, absolute parameter values remain fixed if they fit the new ±3P slider range; out-of-range values are clamped to match the slider controls. In Fixed INT mode, rounding and input snapping can still change the numerical result. The selected preset name is cleared if parameter ratios cease to match the loaded preset.

Why presets sometimes change P automatically

Some graded presets contain halves or quarters of P. When loading one in Fixed INT mode at too small a scale to represent its fractional coefficients as integers, the application chooses P=16 and shows a notice. This keeps the example’s intended integer parameter ratios representable. It is not a silent training step.

PART III | Complete interface reference

12. Display toolbar, presets, and learning guide

The upper-right toolbar offers three independent presentation controls. These affect appearance only; they do not change W1, W2, B, P, target, or the numerical evaluation.

Preference	Available selections
Color theme	Ocean Blue Light; Evergreen Light; Warm Sand Light; Glacier Steel Light; Ivory & Gold Light; Midnight Dark; Graphite Dark; Electric Cobalt Dark; Amethyst Dark; High Contrast Dark.
Interface font	IBM Plex Sans (default); IBM Plex Sans Condensed; IBM Plex Serif; System Sans; Arial/Helvetica; IBM Plex Mono.
Text size	Small (0.91×), Medium (1.00×, default), Large (1.14×), Extra Large (1.30×) - interface-level multipliers.

The user’s color theme, font, and font-size preferences are saved under browser localStorage keys prefixed “mliglon-igl-” when local storage is allowed. They may reset on other devices, profiles, or private sessions. IBM Plex fonts are requested from Google Fonts, but a system fallback is used if unavailable.

The twenty preset buttons

There are two separately labeled rows: 16 exact Boolean gates and four graded examples. Clicking a preset updates the model parameters, active badge, status, tables, plots, and search history. A preset is not applied as an image or lookup table: its W1, W2, B parameters are loaded, and the node is evaluated by the common core.

Help and tooltips

Many headings and controls have a small “?” help button. Hover with a pointer, focus by keyboard, or activate the button to read its explanation. Tooltips close on blur, pointer exit, scroll, outside interaction, or Escape. The banner also contains a collapsible “Getting started” section with a five-step orientation.

13. Panel 1: node parameters

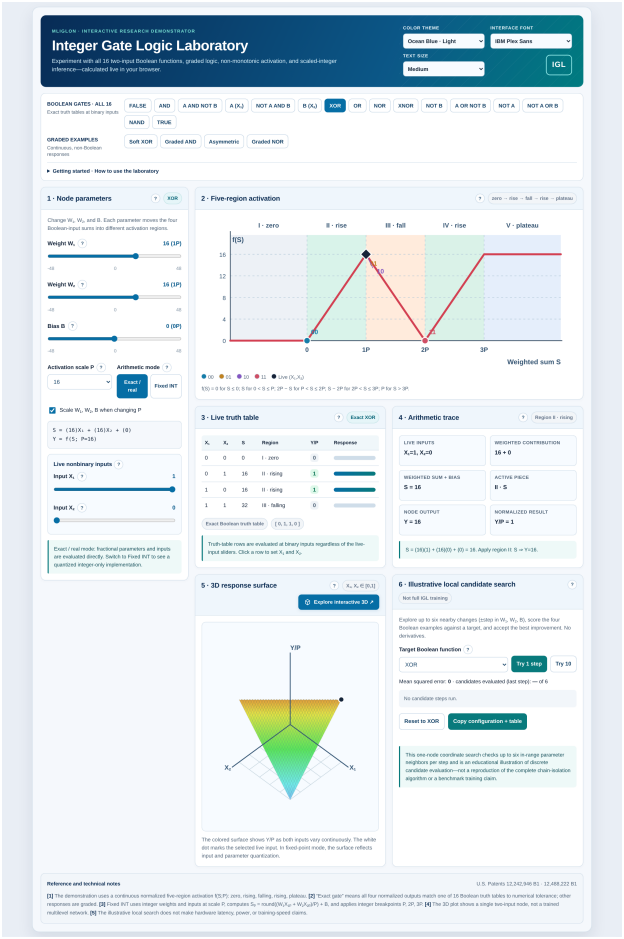


Figure 4. The parameter panel is part of the complete v1.3.0 interface. Zoom the PDF or the live page to read individual control labels.

Control	Behavior and interpretation
Weight W1	Signed contribution from X1. Slider can cross zero to change response direction.
Weight W2	Signed contribution from X2. Unequal weights produce asymmetric behavior.
Bias B	Constant offset added to the weighted input sum.
Activation scale P	Chooses integer scale and breakpoints 0, P, 2P, 3P.
Exact / real	Direct fractional evaluation; preserves continuous values.
Fixed INT	Quantized input, weight and bias arithmetic at scale P.
Rescale parameters	Attempts to preserve normalized parameter ratios when P changes.
Live input X1, X2	Moves the current selected point on response plots; not the truth-table corners.
Formula card	Displays the current equation and scaled implementation context.

Reading parameter labels

Parameter outputs show the raw numerical value and, in parentheses, its ratio to P. For example “16 (1P)” indicates W1=16 with P=16. This presentation helps you recognize that changing P while preserving normalized coefficients can leave the four Boolean truth-table responses unchanged.

Recommended exploration sequence

1. Choose XOR and look at the four weighted sums before moving anything.
2. Increase W2 or reduce W1 and compare the asymmetry.
3. Move the bias and watch which inputs cross the region boundaries.
4. Use a graded example, then compare it with a nearby exact gate.
5. Return to a preset before trying a new concept to avoid accumulating unrelated parameter changes.

14. Panel 2: five-region activation plot

This SVG chart draws the transfer curve and the five activation regions, with region-colored backgrounds. Four colored circular markers show the weighted sums and outputs for input pairs 00, 01, 10, 11. A distinct diamond marker shows the currently selected live fractional-input point.

The horizontal axis is weighted sum S, and the vertical axis is the raw activation f(S). Changing P stretches the breakpoint positions; changing weights or bias moves the markers horizontally and may move their output values to another region. The graph recalculates directly from the model rather than from a precomputed screenshot.

Interpreting coincident markers

Two or more Boolean input combinations can generate the same weighted sum or output. When this happens, the markers overlap; the interface offsets marker labels to make coincident points easier to distinguish. This is common for symmetric functions such as XOR with W1=W2.

> **Reading a breakpoint**
>
> At S=P, Y=P. At S=2P, Y=0. At S=3P, Y=P. The continuous curve changes slope at these locations. The vertical axis is raw Y, whereas panel 3 and WebGL’s output scale use normalized Y/P.

15. Panel 3: live truth table and classification

Column	Meaning
X1, X2	The binary input combination used for that row.
S	Weighted sum generated by that binary pair.
Region	The active five-region piece of the activation function.
Y/P	Normalized value of the node response for the pair.
Response	Graphical bar representing the normalized output magnitude.

The panel always evaluates four rows at inputs 00, 01, 10, 11, even if the live-input sliders show 0.45 and 0.40. If all four normalized values equal exactly one gate's ideal vector within the model's numeric tolerance, an "Exact [gate]" label is shown. Otherwise, the interface identifies the nearest Boolean gate by mean absolute error (MAE) and prints the actual four-value vector.

Interactive row inspection

Click a row—or focus it and press Enter or Space—to copy that row's binary X1 and X2 values into the live-input controls. The arithmetic trace, live point, and WebGL marker update accordingly. This is a useful way to verify which region is responsible for each Boolean corner.

16. Panel 4: arithmetic trace

The arithmetic trace is the most direct numerical debugging view. It identifies the selected X1/X2 values, their weighted contribution, weighted sum with bias, active region and piece, raw output Y, and normalized Y/P. In Fixed INT it also reflects the quantized arithmetic path.

1. Select a truth-table row or move the fractional live-input sliders.
2. Find the displayed weighted contribution. For Exact/real this corresponds to W1X1+W2X2.
3. Confirm the computed S after adding bias.
4. Locate the active region and its formula in panel 2.
5. Confirm the raw Y and normalized Y/P values.
6. If Fixed INT is selected, use the integer trace to identify input quantization and the rounded division by P.

The arithmetic trace lets users distinguish an incorrect assumption about activation mathematics from a visual interpretation error. If the surface appears surprising, verify the trace before adjusting mesh density, projection, or color palette.

17. Panel 5: the original projected 3D response surface

Version 1.3.0 retains the original 2D Canvas graph, which uses a fixed pseudo-3D projection of Y/P over the input square X1, X2 in [0,1]. It is automatically updated as model parameters change and displays a white dot at the live-input location. A 2D canvas implementation does not require WebGL and is intentionally preserved as an accessible fallback.

The Canvas region was enlarged in v1.2.0 to keep the plotted axes and corners within its panel. It serves as an always-visible overview. The WebGL popup does not replace the 2D graph; the button in the panel heading provides a deeper way to inspect the same model.

> **Panel 5 heading control**
>
> Click "Explore interactive 3D" to open the resizable, in-page WebGL explorer. It does not open a browser tab and normally does not trigger popup blockers. The existing graph remains behind the modal window and is restored immediately when the viewer closes.

18. Panel 6: illustrative local candidate search

This panel accepts a target among all 16 two-input Boolean functions and evaluates up to six neighboring parameter changes. The candidate set consists of ±delta changes to W1, W2 and bias, one parameter at a time. Each eligible candidate is tested on the four binary corners; only the candidate with the best strict decrease in mean squared error is accepted.

Search objective

MSE = (1/4) × sum over 00, 01, 10, 11 of (current Y/P − target bit)^2. A step is accepted only when the best tested candidate improves MSE.

Control or indicator	Purpose
Target Boolean function	Select which of the 16 truth tables the search should approximate.
Try 1 step	Evaluate up to six one-parameter neighbors; accept the best improvement.
Try 10	Repeat up to ten greedy steps, stopping early if no improvement occurs.
Mean squared error	Current model error on four binary input rows, relative to target.
Candidates evaluated	Actual number of in-range neighbors checked in the last step (maximum six).
Search log	Reports accepted change, before/after MSE, or a no-improvement condition.
Reset to XOR	Load exact XOR preset at current mode and scale.
Copy configuration + table	Copy current parameters and truth-table rows as plain text.

Step size and boundaries

The real-mode candidate step is delta=P/4. In Fixed INT the step is delta=1. A candidate is skipped when the resulting parameter would fall outside ±3P, so fewer than six candidates may be evaluated. The algorithm is greedy, does not combine simultaneous changes, and can become trapped at a plateau or local minimum. Running ten steps does not guarantee a correct target gate.

Correct interpretation of the search result

The local search is an educational illustration of derivative-free candidate evaluation. It should not be cited as a benchmark of the complete patented training algorithm, as proof of convergence, or as a demonstration of production-grade network optimization.

PART IV | The WebGL interactive 3D explorer

19. Opening, positioning and closing the explorer

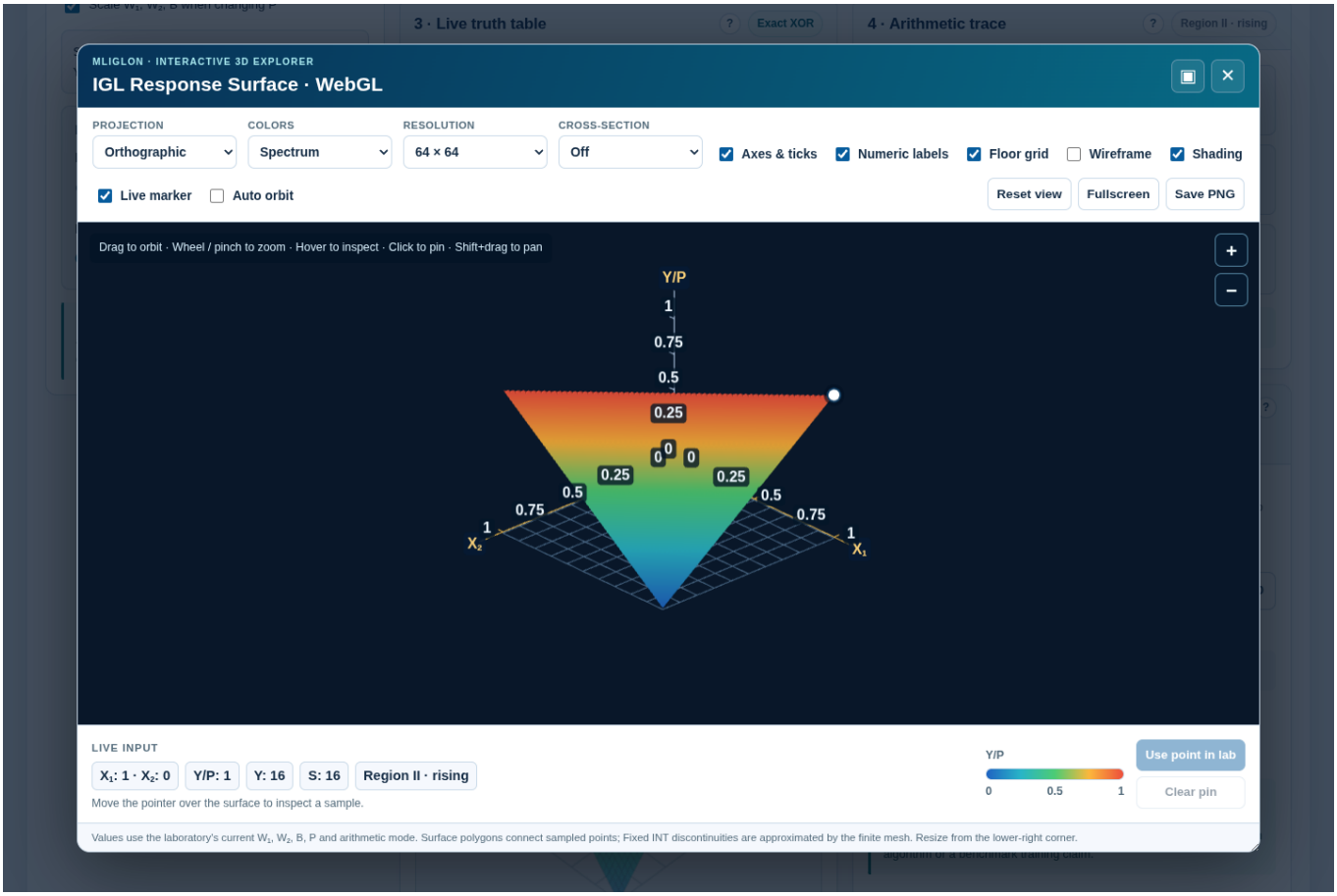


Figure 5. Actual WebGL popup in Chromium: toolbar, rotatable surface, normalized-axis labels, live readout, color legend, and action buttons.

The separate explorer is loaded only after the panel 5 button is activated. Its rendering uses WebGL 1 with built-in vertex and fragment shaders and no third-party 3D library. The overlay contains a header, controls, a large drawable viewport, numeric inspection readouts, and a footer with limitations.

1. Click Explore interactive 3D in panel 5.
2. Wait for the colored surface to render. If a WebGL support message appears, use the original panel 5 chart instead.
3. Drag from the middle of the surface to orbit; use the mouse wheel or the +/− controls to zoom.
4. Change projection, palette or resolution without modifying the node’s actual W1, W2, B, P values.
5. Resize by dragging the lower-right corner on desktop; Maximize/Restore or Fullscreen can expand the viewing area.
6. Press Escape, click the close button, or click outside the dialog to return to the laboratory.

The viewer synchronizes with live laboratory parameter changes. Its text theme and font follow the main laboratory appearance; its camera, palette and feature switches are independent presentation settings for the popup.

20. Camera controls: orbit, pan, zoom and reset

Interaction	Result
Mouse/touch drag	Orbit around the response surface (azimuth/elevation).
Shift + mouse drag	Pan the view relative to its current center.
Mouse wheel	Zoom in or out around the current view.
Two-finger pinch	Zoom on supported touch devices.
+ and − viewport buttons	Increase or decrease the camera zoom.
Double-click	Restore the initial camera view.
Reset view button	Restore initial azimuth, elevation, zoom and center.
Canvas arrow keys	Orbit while the WebGL canvas has focus.
Canvas + / − keys	Keyboard zoom while the canvas has focus.
Canvas 0 key	Reset the view from the keyboard.
Escape	Close the popup.

When using keyboard shortcuts, click or tab to the surface canvas first. The viewer applies bounded elevation and zoom to prevent unbounded zoom or camera inversion. Auto orbit is disabled by default; enabling it makes the camera rotate automatically, which can be useful for presentations but not for careful static inspection.

21. Projection, color palettes and surface resolution

Projection

Orthographic projection preserves parallel lines and makes relative height and planar geometry easier to compare without perspective foreshortening. Perspective projection introduces depth cues and is often preferable for intuitive spatial viewing. Changing projection does not change any underlying IGL output value.

The six WebGL color palettes

Palette	Use case
Spectrum	High-contrast multicolor range; useful default.
Ocean / Cyan	Cool blue-to-cyan rendering for the MLiglon technology style.
Thermal	Dark-violet to warm-yellow progression; useful for height gradients.
Terrain	Green/brown/natural hues; emphasizes elevation-like structure.
Graphite	Restrained grayscale-to-blue highlights for technical prints.
Cobalt / Gold	Cool-to-warm corporate alternative with gold highlights.

Every palette maps normalized Y/P from 0 to 1. The color legend helps decode this mapping; colors are not separate class probabilities or measured physical temperatures. For print interpretation or comparison across images, retain numerical labels and the Y/P legend.

Mesh resolution

Setting	Meaning	Practical choice
32 × 32	Coarse sampled input grid.	Fast exploration and weaker devices.
64 × 64	Default sampled grid.	Balanced standard viewing.
96 × 96	More detailed sampled mesh.	Improved shape readability.
128 × 128	Highest built-in sampling.	Detailed visuals; more rendering cost.

Increasing mesh density creates more triangles and approximates steep changes with finer spacing, but the model is not being trained or made inherently more accurate. Particularly in Fixed INT mode, finite triangles interpolate between samples even when the underlying function is step-like. The inspection readout evaluates the core directly at the selected point.

22. WebGL visibility controls

Toggle	Effect
Axes & ticks	Draws input and output axes and tick marks.
Numeric labels	Displays tick values next to selected axes.
Floor grid	Shows a reference plane beneath the response surface.
Wireframe	Overlays mesh edges, revealing triangle sampling.
Shading	Enables directional diffuse lighting for shape cues.
Live marker	Shows the laboratory’s selected X1/X2 position on the surface.
Auto orbit	Continually rotates the camera until switched off or the viewer closes.

For an instructional screenshot, a useful default is axes on, numeric labels on, floor grid on, wireframe off, shading on, and live marker on. Turn on wireframe to show how geometry is sampled; turn it off again when emphasizing the mathematical shape rather than the sampling grid.

23. Interactive point inspection and pinning

The cursor is ray-picked against the visible triangles. The popup then evaluates the resulting input coordinates through the shared numerical IGL core and displays the live X1, X2, S, raw Y, Y/P, and activation region. A click pins the inspected location so that the values remain visible while the cursor moves elsewhere.

1. Hover on the visible colored mesh, not merely on the empty dark background.
2. Read the status label “HOVER INSPECTION” and the numeric fields X1, X2, Y/P, Y, S and region.
3. Click the surface. The label changes to “PINNED SAMPLE” and a point marker identifies the selection.
4. Choose “Use point in lab” to transfer X1 and X2 into the main laboratory input controls.
5. Observe panel 4 and panel 5 update to the new live point.
6. Click “Clear pin” to discard the selection and resume hover inspection.

> **Input-synchronization precision**
>
> “Use point in lab” rounds the selected X1 and X2 to the nearest 0.05 before applying them to the main laboratory. The point inspected within WebGL may therefore differ slightly from the subsequent live slider point. This is deliberate, matching the main real-mode input control granularity.

The readout is obtained by calling the evaluation core with the current model state, not by taking the interpolated height of the triangle. This matters when Fixed INT produces discontinuous outputs.

24. Cross-section tools and spatial interpretation

The Cross-section dropdown provides Off, X1 fixed, and X2 fixed modes. It draws a line along a selected input coordinate using the laboratory’s current live value for the held-constant axis. Use these slices to investigate one-dimensional behavior while retaining the 3D context.

1. Load XOR in Exact/real mode and set X1 to a moderate fractional value such as 0.50.
2. Open WebGL and select Cross-section → X1 fixed.
3. Orbit to reveal the slice line across changing X2.
4. Move X1 in the underlying laboratory when the viewer is closed, reopen it, and compare the new slice position; the view stays linked to the model.
5. Switch to X2 fixed and repeat. Use numerical hover readouts to understand region changes along the chosen line.

Cross-sections are visual overlays on the 3D surface, not a separate 2D response-curve chart. They show how the output changes while one input remains fixed. At high contrast boundaries, increasing mesh detail may produce a cleaner visual slice without changing the underlying exact evaluation.

25. Resizing, fullscreen, export and model synchronization

Window and viewport

The explorer is a resizable in-page dialog, not a separate browser window. In desktop environments supporting CSS resizing, drag the lower-right corner. Use Maximize/Restore for a large page overlay; Fullscreen requests the browser's Fullscreen API and may be denied by browser policy. The view re-renders when dimensions change.

Saving a PNG

1. Choose a camera angle and desired projection.
2. Select a color palette, mesh resolution, cross-section and visibility options.
3. Optionally pin an interesting surface point so its numeric values remain visible.
4. Press Save PNG. The browser combines the WebGL canvas and numeric-overlay canvas into a local PNG.
5. Save or open the downloaded file, normally named MLiglon_IGL_3D_Surface.png.

The export is an image of the current display. It does not export the sampled mesh or the numerical values as CSV, JSON, or 3D mesh data. Browsers may require download permission; if the browser cannot create the image, the viewer reports an error.

Synchronization rules

W1, W2, B, P, arithmetic mode and live X1/X2 values are shared with the underlying laboratory. Changes that affect model output cause the surface to be regenerated. Saved pins are reevaluated when the model changes. Camera angle, color palette, resolution and on/off drawing controls do not change the node parameters.

26. Performance and rendering considerations

The WebGL explorer creates its graphics context lazily when first opened, which keeps the regular page usable without a GPU. Four mesh-density selections give visitors a tradeoff between detail and graphics workload. WebGL 1 shaders, camera matrices, ray/triangle picking and color mapping are implemented in the supplied JavaScript rather than a remotely hosted 3D framework.

A graphics-intensive viewer can perform differently across machines. If orbiting is sluggish, use 32 × 32 or 64 × 64, disable wireframe and auto orbit, close other resource-intensive applications, and try a current browser with functioning graphics drivers. The laboratory's CPU-side numerical core remains the source of truth for the results.

The renderer supports context loss and restoration messages. The presence of such handlers does not guarantee that every device/driver combination has been validated. The development report documents Chromium with software WebGL, desktop/tablet/mobile viewport checks, and explicit limits on Safari, mobile GPUs, physical multitouch, and forced context-loss testing.

PART V | Guided experiments and lesson plans

27. Experiment 1: reproduce XOR with one node

> **Goal**
>
> Understand how a non-monotonic activation gives the exact XOR truth table with W1=P, W2=P, and B=0.

1. Set P=16 and Exact/real mode.
2. Choose XOR from the Boolean preset row.
3. Verify W1=16, W2=16, B=0.
4. Read all four truth-table rows and their activation regions.
5. Open the WebGL viewer and inspect the response surface. Orbit to see the rising and falling sections.
6. Select each row from the truth table to move the white live marker. Record the raw Y and normalized Y/P in panel 4.

Input	S	Region	Y	Y/P
00	0	I	0	0
01	16	II	16	1
10	16	II	16	1
11	32	III	0	0

Interpretation: the 11 input yields S=2P, a larger sum than either 01 or 10, but the falling region maps that larger sum back to zero. This is the distinctive non-monotonic mechanism behind the simple single-node XOR construction.

28. Experiment 2: exact XNOR and the effect of bias

1. Click XNOR and use P=16, Exact/real.
2. Verify normalized coefficients W1/P=1, W2/P=1, B/P=1, so W1=16, W2=16, B=16.
3. Read the four normalized outputs: 1, 0, 0, 1.
4. Observe how adding bias shifts all four input sums relative to the same fixed activation regions.
5. Compare the XNOR and XOR surfaces; preserve the same projection for a fair comparison.

Input	S/P	Y/P
00	1	1
01	2	0
10	2	0
11	3	1

Teaching point: XOR versus XNOR does not require a new activation function. Changing a single bias shifts the four sums to different intervals of one transfer function.

29. Experiment 3: AND, NAND, NOR and OR

1. Select AND and verify a single “1” at input 11.
2. Select NAND and verify a single “0” at input 11.
3. Select NOR and verify a single “1” at input 00.
4. Select OR and verify zeros only at input 00.
5. Look at the normalized coefficient triples in the reference table, paying attention to signed weights and bias.
6. For each case, compare the four marker positions in panel 2 and the 3D response surface in panel 5.

These four examples show that exact binary-corner behavior is not restricted to symmetric positive-weight choices. Negative weights and nonzero bias can reposition the corner sums on the activation curve. Do not assume the continuous response away from those corners is the same as a conventional fuzzy-logic operator.

30. Experiment 4: asymmetry versus symmetry

1. Choose the Asymmetric graded example.
2. Set P=16, Exact/real.
3. Compare W1=16 and W2=8. The two single-high input pairs need not have the same output.
4. Click rows 01 and 10 and compare the raw values S and Y.
5. Open WebGL, orbit, and enable the X1 fixed slice.
6. Swap the roles of W1 and W2 manually and compare what changes.

Teaching point: the equality W1=W2 produces an input-exchange symmetry in the weighted sum. When the weights differ, S(X1,X2) generally does not equal S(X2,X1); the resulting output may also differ, except when two sums happen to fall on equal-output sections of the activation function.

31. Experiment 5: inspect graded logic and continuous inputs

1. Choose Soft XOR and retain Exact/real.
2. Read the four truth-table entries and identify outputs that lie strictly between zero and one.
3. Set live X1=0.25 and X2=0.75, then compare with X1=0.75 and X2=0.25.
4. Open panel 5 and WebGL; hover the surface to inspect values between the Boolean corners.
5. Discuss why a continuous output cannot be inferred solely from the truth table.

Teaching point: a four-row Boolean truth table is a sparse sample of the continuous two-input response. Many parameter choices can match the same Boolean corner outputs yet produce different graded behavior inside the input square. Evaluating those intermediate inputs is one reason the 3D viewer is useful.

32. Experiment 6: Real versus Fixed INT

1. Load the Asymmetric example with P=16 and Exact/real mode.
2. Set X1=0.45 and X2=0.40.
3. Record S, raw Y, normalized Y/P, and active region: 10.4, 10.4, 0.65, II.
4. Switch to Fixed INT. The inputs are represented at integer precision 1/P and the model uses rounded integer operations.
5. Compare the resulting arithmetic trace; for the same requested input pair, the theoretical integer calculation gives X1p=7, X2p=6, S=10, Y/P=0.625.
6. In WebGL enable wireframe, observe any step-like transitions, and note that the mesh approximates discontinuities.

> **Caution about control snapping**
>
> Changing arithmetic mode can snap live slider values. Use panel 4 to report the actual represented X1/X2 after the switch, rather than assuming it retains the previous decimal requests exactly.

33. Experiment 7: study activation scale and parameter rescaling

1. Load XOR with P=16 and select “Scale W1, W2, B when changing P”.
2. Change P to 32. Confirm that W1 and W2 become 32, preserving their ratios to P.
3. Check the four Boolean truth-table normalized outputs; they remain XOR for this exact preset.
4. Disable parameter rescaling and change P to 8. Compare the resulting normalized ratios and watch for slider-bound clamping.
5. Restore XOR at P=16 to reset a clean baseline.

The distinction here is between preserving normalized coefficients and preserving absolute parameter magnitudes. P changes both activation thresholds and the integer quantization grid. Consequently, “same visible gate” and “identical continuous numerical behavior” are not synonymous in every mode or scale.

34. Experiment 8: greedy search toward a different gate

1. Start with XOR at P=16 in Exact/real mode.
2. In panel 6 choose AND as the target.
3. Observe the initial MSE against AND.
4. Click Try 1 step and read the candidate count, accepted parameter change and new MSE.
5. Click Try 10 and observe whether the algorithm improves further or stops on a plateau.
6. Compare the resulting truth-table vector to the exact AND preset.
7. Reset to XOR, then load exact AND to see what the target solution looks like.

The experiment is an illustration of scoring and accepting parameter neighbors. Results depend on the starting point, step size, current mode and allowed parameter range. A lack of further improvement does not imply the desired function is impossible to represent; the preset catalog independently demonstrates that all 16 exact functions are representable.

35. Experiment 9: use cross-sections to find non-monotonic behavior

1. Load XOR, set X1=0.50 and X2=0.50.
2. Open WebGL and enable X1 fixed.
3. Rotate to a shallow oblique view and observe the slice path as X2 changes.
4. Compare with X2 fixed, then switch to perspective projection.
5. Hover along the slice and note where S enters region III and Y decreases as S increases.
6. Save a PNG documenting the camera and numerical state.

A fixed-input slice can make the rise–fall–rise mechanism easier to see than a full top-down surface. The exact values, not the apparent camera height, should be used to explain quantitative properties.

36. Experiment 10: prepare a reproducible technical observation

1. Choose a preset, mode and activation scale, and record them in the experiment notes.
2. Set both live inputs and write down W1, W2, B, P, X1, X2, S, Y, Y/P and activation region from panel 4.
3. Use Copy configuration + table in panel 6 and paste the text into a local research notebook.
4. Open WebGL and choose a projection, palette, mesh density and viewing angle.
5. Pin a point if needed and capture a PNG.
6. Record whether values are exact core outputs or sampled visual approximations. Include the browser and plugin version with any report.

This workflow supports demonstrations, issue reports and independent reproduction. A screenshot alone is not an adequate source for numerical validation; always accompany it with the evaluated parameters and the four-row truth table.

PART VI | Accessibility, troubleshooting and administration

37. Accessibility and presentation tips

- Use Large or Extra Large when the default text is too small for projector demonstrations or smaller screens.
- Try IBM Plex Sans Condensed when wider labels wrap excessively; preserve legibility rather than forcing dense screens.
- For printed screenshots, choose a palette with sufficient contrast and keep numeric labels enabled.
- The main truth-table rows can be selected by mouse and keyboard (Enter/Space); many help controls are keyboard-focusable.
- WebGL keyboard orbit and zoom work when its canvas has focus; the dialog traps Tab focus at its ends and Escape closes it.
- The graph itself is visual. For accessible numerical interpretation, use the text-based arithmetic trace and truth table.
- Auto orbit defaults off, supporting users who prefer reduced motion and careful inspection.

Font preferences, screen magnification, pointer size, and assistive technology support vary by operating system and browser. While the plugin includes accessible labels, readouts and keyboard shortcuts, full accessibility conformance has not been established by the supplied test report. Production deployment should include independent keyboard and screen-reader evaluations.

38. Troubleshooting: the main laboratory

Symptom	Likely cause / recommended action
Nothing appears where shortcode is placed	Confirm plugin active, shortcode spelled [igl_demo], page builder evaluates shortcodes.
Only a warning about one instance	Remove duplicate [igl_demo] instances on the same page.
Styling missing or compressed	Check stylesheet requests, theme CSS conflict, CSP and CDN cache.
Preset buttons do not change results	Inspect browser JS errors and verify igl-core.js loads before igl-demo.js.
Font remains a system font	Google Fonts may be blocked; expected fallback behavior.
Preference resets on revisit	Local storage unavailable, blocked, or in private browsing.
Slider and displayed value differ	Hard-refresh, check current mode/scale; Fixed INT snaps input and parameters.
After changing P, parameters changed	Rescale checkbox is enabled or a value was clamped to ±3P.
Clipboard copy failed	Browser permission, insecure context or clipboard policy; copy manually or use HTTPS.
Truth table disagrees with live point	Truth table always uses binary corner inputs; live point may be fractional.

39. Troubleshooting: the WebGL explorer

Symptom	Recommended check
Explorer says WebGL unavailable	Try current Chrome, Firefox or Edge, verify GPU and driver support, review policies or extensions.
Dark viewport with no geometry	Allow a moment for shader compilation; check browser console and graphics support.
Surface is jagged	Increase mesh density; know that finite meshes approximate discontinuities.
Low frame rate	Use 32×32/64×64, turn off wireframe and auto orbit, close GPU-heavy tabs.
Labels overlap	Rotate or zoom, use a larger popup, disable numeric labels temporarily.
Mouse drag seems not to pan	Use Shift+drag for pan; ordinary drag rotates.
Click appears to miss intended location	Click on visible triangle mesh; camera angle and finite sampling affect ray-picked coordinates.
Apply point changed X1/X2 slightly	Transferred inputs snap to 0.05 increments in main laboratory.
Fullscreen fails	Browser may block request; Maximize can be used instead.
PNG not downloaded	Check browser download permissions and pop-up/download policies.
Context lost or reset	Follow message; close/reopen viewer, refresh or change browser/graphics configuration.

40. Security, privacy and network considerations

The laboratory’s model calculations, truth-table evaluation, local candidate search, and 3D rendering happen in the visitor’s browser. The supplied JavaScript does not require a remote inference service. Visual preferences may be stored in that browser’s localStorage. A saved image is generated locally and downloaded through the browser. The package requests IBM Plex font stylesheets from Google Fonts when remote font loading is allowed, subject to the visitor’s network and browser policy.

No special WordPress database table, server-side model endpoint, or credentialed user input interface is defined by this plugin. However, general WordPress operation, hosting logs, analytics, caching extensions and external font delivery still have their own privacy considerations and should be assessed separately. The plugin does not collect or save research sessions to an application database.

Security policy and content delivery

A strict Content-Security-Policy can block styles, font files, dynamically constructed display elements, or WebGL behavior depending on the browser and configuration. If deploying on a highly restricted site, inspect the browser console and explicitly allow only required sources. Do not disable security protections blindly to force WebGL or remote fonts to load.

41. WordPress developer and file-structure reference

File	Responsibility
mliglon-igl-demo.php	Plugin metadata, script/style registration, enqueue logic, shortcode and template loading.
templates/laboratory.html	All fixed interface markup, controls, panels and text labels.
assets/igl-core.js	Shared deterministic model, gates, activation, quantization, truth rows, error and search.
assets/igl-demo.js	Browser UI state, interactions, SVG, legacy Canvas rendering, themes and tooltips.
assets/igl-webgl.js	Independent 3D popup, shader renderer, camera, picking and exports.
assets/igl-demo.css	Responsive layout, themes, typography, panel and popup styles.
README.txt	Installation, features and known interpretation limits.
TEST_REPORT.txt	Recorded model, browser and PHP verification results.
tests/*.js / *.php	Model regression, WebGL math and WordPress stub checks.

Asset load order

WordPress registers the model core first. The WebGL explorer depends on the core, and the browser controller depends on the explorer. This ensures the numerical API is available before the UI tries to instantiate it. Scripts are registered as footer scripts, and styling depends on the optional font stylesheet.

HTML integration and instance limits

The shortcode callback returns a static HTML template. A static guard prevents two simultaneous laboratory instances from being emitted on one page. The script controller also checks for an already-initialized root. These are deliberate implementation constraints; developers adding multi-instance support would need to namespace IDs and rework the root/state assumptions.

Upgrading or customizing safely

Preserve a working archive before editing. If you change the activation, arithmetic or preset coefficients, update numerical tests and the public explanatory notes together. If you change CSS or add themes, test readable contrast, mobile overflow and the extra-large font setting. If you change WebGL geometry or labels, verify both projections and point picking against the shared evaluation core.

42. Verification results and known gaps

The distributed TEST_REPORT.txt describes 25,838 model assertions across gate tables, seven P values, both arithmetic modes, randomized node evaluations and search properties. It also reports 3,290 camera/geometry/color assertions and 174 WebGL browser interaction checks at three viewport sizes. PHP syntax, stub WordPress bootstrap and JavaScript parsing checks passed in the development environment.

The reported browser exercising used Chromium with software WebGL under a virtual display. This is stronger evidence than a simulated graphics API, but it is not a substitute for validation in production

WordPress, mobile Safari, Android GPUs, every driver configuration, or the final caching/security environment. The report explicitly notes that physical pinch gestures, forced context loss, OS-level downloaded-file audits, and some fullscreen cases were not comprehensively tested.

> **Testing language to use in publications**
>
> Appropriate: “The v1.3.0 demonstrator passed numerical unit checks and automated Chromium WebGL interaction tests.” Avoid: “Exhaustively tested on all browsers and hardware” or “benchmark-proven IGL hardware speedup.”

43. Frequently asked questions

Is this a complete neural network training tool?

No. It demonstrates one node and an illustrative local coordinate search; it is not the complete IGL training system.

Why are there 16 gates?

Four binary input pairs have 2^4 possible truth-table output vectors.

Why can XOR be produced by one node?

The five-region response rises, falls, rises again and saturates; the 11 weighted sum can map back to zero.

Does exact XOR mean a perfect continuous XOR surface?

It means exact agreement with the four binary corner outputs. The intermediate continuous surface is defined by the node formula and may be graded.

Why is the truth table unchanged when I move a live slider?

It always tests 00, 01, 10, 11. Live X1/X2 select a separate point for the trace and response surface.

Why does a weight show “1P”?

The interface also prints the normalized parameter value relative to activation scale P.

Why did selecting a preset change P?

In Fixed INT, some fractional example coefficients require a larger P to be represented exactly as integer parameters.

What does the nearest-gate error mean?

The mean absolute distance of the four normalized outputs from the closest ideal Boolean truth-table vector.

Why are Fixed INT surfaces stepped or jagged?

Inputs and intermediate arithmetic are quantized. Finite WebGL triangles can further smooth/disguise abrupt jumps visually.

Is the WebGL renderer needed for numerical correctness?

No. All calculations use the same shared core. WebGL is an optional advanced visualization.

Can I export a 3D mesh?

Not in v1.3.0. Save PNG creates an image only.

Can I run two laboratories on one page?

Not without modifying the plugin. The supplied shortcode intentionally allows one instance.

Are font files packaged in the plugin?

No. Google Fonts is used where permitted, otherwise system fallbacks are applied.

Are my experiments uploaded to a server?

The plugin itself performs calculations and rendering locally; it does not define a research-data upload endpoint.

Why did the local search stop with nonzero MSE?

It is a greedy bounded neighbor search and can be trapped in local minima or plateaus.

Does this tool measure energy or inference latency?

No. It reports mathematical outputs and visualizations, not empirical hardware benchmarks.

Appendices | Quick reference and reproducibility

Appendix A. Complete gate matrix

The 16 bit strings are listed in the exact ordering used by the source: 00, 01, 10, 11. Coefficients are relative to P. Choose P=16 to obtain the actual raw W1, W2, bias values by multiplication.

Function	00	01	10	11	W1/P	W2/P	B/P
FALSE	0	0	0	0	0	0	0
AND	0	0	0	1	1	1	-1
A AND NOT B	0	0	1	0	1	-1	0
A (X_1)	0	0	1	1	1	0	0
NOT A AND B	0	1	0	0	-1	1	0
B (X_2)	0	1	0	1	0	1	0
XOR	0	1	1	0	1	1	0
OR	0	1	1	1	2	2	-1
NOR	1	0	0	0	-1	-1	1
XNOR	1	0	0	1	1	1	1
NOT B	1	0	1	0	0	-1	1
A OR NOT B	1	0	1	1	2	1	1

Function	00	01	10	11	W1/P	W2/P	B/P
NOT A	1	1	0	0	-1	0	1
NOT A OR B	1	1	0	1	1	2	1
NAND	1	1	1	0	-2	-2	3
TRUE	1	1	1	1	0	0	1

Appendix B. Default settings and saved preferences

Item	Default
Start preset	XOR
Initial parameters	W1=16, W2=16, bias=0, P=16
Initial mode	Exact / real
Initial inputs	X1=1, X2=0
Search target	XOR
Rescale on P change	Enabled
Interface theme	Ocean Blue Light
Font and size	IBM Plex Sans / Medium
WebGL projection	Orthographic
WebGL palette	Spectrum
WebGL mesh	64 × 64
WebGL slice	Off
WebGL toggles on	Axes, numeric labels, floor grid, shading, live marker
WebGL toggles off	Wireframe, auto orbit

Only the visual preference selections are persistently stored by the main controller. Users should not assume that their custom model parameters, local-search trajectory, camera angle, or pinned points are automatically saved between page sessions.

Appendix C. Symbol and term glossary

Term	Definition
Activation	Piecewise function Y=f(S;P).
Bias	Constant B added to weighted sum.
Boolean truth table	Four normalized outputs for two binary inputs.
Candidate search	One-step greedy comparison of nearby parameter configurations.
Core evaluator	Shared JavaScript numerical implementation used throughout the application.
Exact gate	A four-row response that agrees with an ideal Boolean function to numerical tolerance.
Fixed INT	Input, weight, bias and intermediate integer rounding model.
Graded response	A response with one or more intermediate normalized values.
Local minimum	Configuration where nearby allowed candidate steps fail to improve the objective.
Mesh	Set of sampled triangles used to draw the WebGL response surface.
Normalized output	Y divided by activation scale P.
Orthographic	Camera projection without perspective foreshortening.
Perspective	Camera projection that provides depth cues through foreshortening.
Pinned point	A mesh position selected for persistent inspection.
Region	One of the five activation intervals.
Scale P	Positive parameter setting activation breakpoints.
Slice	Surface profile while one input is held constant.
Truth-table order	00, 01, 10, 11.
WebGL	Browser graphics interface used for GPU-backed rendering of the 3D viewer.

Appendix D. Numerical verification checklist

1. For selected P, evaluate activation at S=0, P, 2P, 3P and at one point inside each region.
2. For each of 16 Boolean presets, compare the four normalized output values against the reference bit string.
3. Repeat selected tests in Exact/real and Fixed INT at multiple P values.
4. Check fractional-input arithmetic with hand-calculated weighted sums.
5. Verify that rescaling P preserves normalized ratios where feasible and identifies clamping.
6. Confirm that search MSE decreases only on accepted improving steps.

7. Compare browser trace values against values obtained from the core evaluation code.
8. Inspect WebGL orthographic and perspective modes; verify hover readout values against direct core evaluation.
9. Test panel functionality on both a wide desktop and a narrow mobile viewport.
10. Document the browser, OS, plugin version, theme and error messages when reporting a problem.

Appendix E. Source and scope of this manual

Primary reference: supplied MLiglon Interactive IGL Laboratory v1.3.0 archive (mliglon-igl-demo-1.3.0.zip). The guide was cross-checked against templates/laboratory.html; assets/igl-core.js; assets/igl-demo.js; assets/igl-webgl.js; assets/igl-demo.css; README.txt; TEST_REPORT.txt; and mliglon-igl-demo.php. Mathematical values and coefficient tables reflect the source implementation rather than general assumptions about related IGL research.

Figures are explanatory drawings or screenshots of the v1.3.0 application. The formulas use conventional mathematical notation in place of code-level variable names where this increases clarity. The document intentionally separates source-verified behavior from recommended operating practices and instructional examples. It does not replace the patent texts, a software API specification, or a controlled scientific benchmark report.

MLiglon Interactive IGL Laboratory v1.3.0 · Comprehensive User Guide · Edition 1 · November 2026. For application updates and related information, visit https://mliglon.com/demo/.