MLiglon IGL Laboratory v1.3.0 — Advanced Walkthrough
9 November 2026 | 75 scenes

=== 01 — The IGL Laboratory · Advanced Technical Walkthrough ===

JUSTIN: Welcome to the advanced tour of the MLiglon Integer Gate Logic Laboratory, version one point three point zero. I am Justin, and throughout this extensive tutorial Richard, Patrick, Rachel, Sarah, and I will explore a different building block for neural computation. This is a guided investigation for website visitors as well as a scientific explanation. We will work with exact two-input Boolean gates, graded responses between the corners, activation scale, integer arithmetic, numerical traces, illustrative local candidate search, and the WebGL three-dimensional explorer. We will take care with the calculations and distinguish what the simulation demonstrates from what requires independent hardware benchmarks. The older nine-minute introduction remains useful as an entry point; here we will go deeper. No complicated training environment is needed: these demonstrations run in the browser.

RICHARD: Rachel, what should a technically curious viewer expect by the closing scene?

RACHEL: They should be able to explain how the two weights and bias place four input patterns on a five-region activation, reproduce all sixteen Boolean truth tables, interpret the continuous response, and distinguish an educational local optimizer from full IGL training.

PATRICK: We will also encourage viewers to challenge results. Inspect the actual numbers, move parameters independently, and remember that a visualization supports understanding but does not substitute for a benchmark.

=== 02 — Why the IGL Laboratory exists ===

RICHARD: 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.

RACHEL: Justin, why build an entire laboratory when there is already a research website?

JUSTIN: Because a sentence about non-monotonic logic cannot convey the geometry. Moving a slider while four marker positions and numerical results respond together makes the claim inspectable, not merely repeatable. We want visitors to find interesting questions, not simply accept a sales pitch.

RICHARD: Whenever a proposed advantage sounds surprising, the right next question is where to place the inputs and how to reproduce the output. This laboratory invites that scrutiny.

=== 03 — Open the laboratory in your browser ===

RACHEL: The demonstration is already available on the MLiglon website. Open the interactive laboratory using a modern desktop, laptop, or tablet browser. Open the page and begin experimenting immediately. Move the sliders and examine the response directly in the browser. A larger screen is helpful for reading the truth table and the three-dimensional surface, but the interface adjusts to smaller displays.

PATRICK: Sarah, does a visitor need a specialized AI accelerator or a high-end graphics card?

SARAH: No. The basic node calculations and panels run in ordinary browser JavaScript. For the optional three-dimensional explorer, the browser needs WebGL support. You can still use the original panel five surface and all the numerical displays when WebGL is unavailable.

RACHEL: A useful first check is to move one input slider and confirm that the weighted sum, active region, and normalized output change together. That gives us a reliable baseline before any detailed experiment.

=== 04 — Interface tour: from top toolbar to six panels ===

SARAH: At the top of the laboratory, the banner names the Integer Gate Logic demonstrator and offers color theme, font family, and overall text size. Below it, two rows of preset controls separate the sixteen exact Boolean functions from the four graded examples. The Getting Started expander provides a short operating guide. Panel one edits weights, bias, P, arithmetic mode, and fractional inputs. Panel two draws the activation curve. Panel three shows the binary truth table. Panel four reports the arithmetic trace. Panel five contains the original fixed-view response surface plus the new WebGL launch button. Panel six demonstrates a local candidate search. All six panels follow the same numerical core.

JUSTIN: Richard, which display should a first-time visitor watch while moving a weight?

RICHARD: Watch the activation graph and the truth table together. The graph explains where the four weighted sums fall; the table explains what those regions mean for the four Boolean inputs. Then consult the arithmetic trace for the precise selected value.

JUSTIN: For a first inspection, change only one weight, then watch the activation markers, truth table, and arithmetic trace respond together.

=== 05 — First session: a ten-minute route ===

JUSTIN: Begin with the XOR preset and P equal to sixteen in Exact real mode. The node has weights sixteen and sixteen and bias zero. Check the truth table in order: zero-zero produces zero, zero-one produces one, one-zero produces one, and one-one produces zero. Click the row for one-zero and notice that the input sliders follow the selected corner. Next select fractional inputs, such as X one equals point four five and X two equals point four. Watch the arithmetic trace and fixed-view 3D marker update. Then open the interactive 3D explorer, rotate the surface, hover to inspect a point, and return to the main interface. Finally compare Fixed INT, XNOR, and NAND before resetting to XOR.

SARAH: Rachel, what is the shortest meaningful first experiment?

RACHEL: Load XOR, choose input zero-one, then one-zero, and compare the table. Move the fractional sliders next. Open the 3D viewer only after you understand the output on those four corners. The exploration becomes much more intuitive in that order.

PATRICK: If one number surprises you, stop and reconstruct S before changing anything else. That is the fastest way to learn the controls.

=== 06 — The two-input IGL node ===

RICHARD: The node is built from a weighted sum and a non-monotonic output function. For two inputs, the sum S equals W one times X one plus W two times X two plus bias B. The sum is passed through the five-region activation function, producing a raw output Y. A normalized response is Y divided by P. A positive weight moves its input contribution to the right along the activation curve; a negative weight moves it to the left. Bias shifts all four Boolean corner sums at once. Because the activation is not monotonic, moving farther to the right can produce either a higher or a lower output depending on the active region. This is how the same elementary node can implement the exact XOR example.

RACHEL: Justin, is this just an ordinary perceptron with a new logo?

JUSTIN: The weighted sum is familiar, but the transfer function is different. It rises, falls, rises again, and saturates. That non-monotonicity permits a single node to distinguish input patterns that a conventional monotone threshold unit cannot separate by one affine decision boundary.

RICHARD: The practical habit is simple: compare the expected behavior with the visible numerical trace, then change one experimental choice at a time.

=== 07 — P: the scale parameter that shapes every node output ===

JUSTIN: Before discussing the sixteen logic gates, we need to understand a parameter that appears everywhere in this laboratory. It is the positive activation scale, P. P is not a probability, and it is not another input signal. It defines the spacing of the activation breakpoints and the raw output scale. Its practical influence depends on the coefficients and input representation, not on P alone. If P is sixteen, the important weighted sums are zero, sixteen, thirty-two, and forty-eight. The raw node output Y ranges from zero to sixteen. Dividing Y by P produces the normalized result Y over P, between zero and one.

RACHEL: Richard, why does one symbol control both where the curve turns and how high it rises?

RICHARD: Because this node uses the same scale for the horizontal breakpoint distances and the vertical output range. The curve first rises from zero to P, falls to zero at two P, rises again to P at three P, and then stays at P. That design makes ratios to P especially useful for comparing different parameter settings.

SARAH: Pause at this point in the application. Look for the P selector in panel one, the breakpoint labels in panel two, and Y divided by P in panel four. These are three views of one common scale.

=== 08 — The mathematics of P: normalized geometry ===

RICHARD: Let us put P into the complete node equation. First compute S equals W one times X one, plus W two times X two, plus B. Then apply the five-region activation f sub P to obtain Y. Finally compute the normalized result u equals Y divided by P. To compare two scales, rewrite S over P as W one over P times X one, plus W two over P times X two, plus B over P. These three parameter ratios determine the position of each input relative to the activation breakpoints.

PATRICK: Rachel, does changing P automatically change the logic represented by the node?

RACHEL: Not necessarily. If you multiply P, both weights, and bias by the same positive factor, the normalized real-mode surface remains the same for the same inputs. The raw sum and raw output scale up, but the ratios and u do not change. If instead you hold the weights fixed while changing P, the relative positions change and the resulting gate may change completely.

JUSTIN: In short, P supplies the unit of measurement for the activation. The ratios W one over P, W two over P, and B over P tell us where the node lives inside that geometry.

=== 09 — P and fixed-point arithmetic across a network ===

RACHEL: An integer neural network needs a consistent interpretation of numbers. This is another reason a scale such as P is useful. In this laboratory, Fixed INT mode rounds each input times P to an integer grid. The effective input levels are zero, one over P, two over P, and so on up to one. Integer weights and bias are also rounded. The weighted integer dot product is divided by P and rounded back to a scaled sum before the five-region activation is applied.

SARAH: Patrick, what happens when one node sends its output to another node?

PATRICK: The next node must know the representation of what it receives. The first node might provide normalized u, or a raw integer value Y that represents u times P. Both can be implemented, but the next set of weights and arithmetic must use a compatible scale. You cannot casually mix raw integers and normalized fractions and expect the same computation.

RICHARD: For example, at P sixteen an integer output of eight represents normalized one half. At P eight, that same normalized value is represented by a raw output of four. Those numbers differ, but the represented quantity is the same. Managing that relationship matters whenever nodes or layers exchange values.

JUSTIN: Do not confuse a useful numeric representation with a measured speedup. The browser is illustrating arithmetic, not benchmarking a physical processor.

=== 10 — Why powers of two are hardware-friendly ===

PATRICK: The selectable P values in this version are one, two, four, eight, sixteen, thirty-two, and sixty-four. They are powers of two. That is a practical choice for binary computer hardware, where shifts and fixed-point conventions can make scaling by two to the k comparatively straightforward. An appropriately designed integer operation may replace division by a power of two with a shift, but a shift by itself does not guarantee the same rounding behavior for negative values. Bit width and overflow must also be considered.

SARAH: Justin, is P restricted by mathematics to powers of two?

JUSTIN: No. The piecewise formula works for any positive P. A decimal-oriented system could use ten, one hundred, or a different application-specific scale. However, those are possibilities for other implementations, not choices offered by this version of the laboratory. On ordinary binary processors, dividing by ten is not the same simple bit-shift operation as dividing by sixteen.

RACHEL: There are tradeoffs. A larger scale can provide a finer fixed-point input grid if enough bits are available, but it also enlarges intermediate integer values and may require more headroom. Hardware cost depends on rounding, bit widths, instruction sets, memory traffic, and the complete data path.

RICHARD: The engineering rule is to choose P and the arithmetic representation together, then verify numerical accuracy and overflow margins rather than assuming that a larger P is always better.

=== 11 — Demonstration: changing P can preserve XOR—or create AND ===

SARAH: Now for a concrete experiment. Load XOR with P sixteen, W one sixteen, W two sixteen, and bias zero. At binary inputs zero-zero, zero-one, one-zero, and one-one, the normalized outputs are zero, one, one, zero. The sums are zero, sixteen, sixteen, and thirty-two.

JUSTIN: Rachel, what if we change P from sixteen to eight and let the laboratory rescale the weights?

RACHEL: The weights become eight and eight. The sums become zero, eight, eight, and sixteen. Relative to the new breakpoints, those are the same positions: zero, P, P, and two P. So the normalized XOR outputs remain zero, one, one, zero in real mode.

PATRICK: Now disable rescaling and set P to eight while keeping the two weights at sixteen and bias zero. Then the four sums are zero, sixteen, sixteen, and thirty-two. At this P, the single-high cases fall at two P, which produces zero; both-high is above three P, producing one. That is now AND, not XOR.

RICHARD: This is a powerful lesson. P is not an isolated cosmetic knob. It changes the relative geometry unless all the parameters are adjusted consistently. Later we will inspect how Fixed INT adds quantization effects even when ratios are preserved.

=== 12 — Using the P selector and the rescale option ===

RACHEL: In panel one, choose P from one, two, four, eight, sixteen, thirty-two, and sixty-four. The default is sixteen. The weight and bias sliders cover negative three P to positive three P. The interface also offers a checkbox called Scale W one, W two, B when changing P. Leave it checked to maintain normalized coefficient ratios, subject to the controls and rounding. Clear it to investigate what happens when the breakpoints move but the coefficient values do not.

RICHARD: Sarah, what do you recommend as a fair comparison between P values?

SARAH: First stay in Exact real arithmetic, keep the weights and bias proportional to P, and compare the normalized truth table and interior response. Next clear the rescale box and repeat. Then switch to Fixed INT to examine how rounding and the input grid change. Change only one of those experimental conditions at a time.

JUSTIN: The P selector is now worth practicing on its own. The distinction between preserving ratios and holding absolute coefficients fixed will recur throughout the

lesson.

=== 13 — The patented five-region activation shape ===

RACHEL: The activation has five successive regions. For nonpositive weighted sum S, output is zero. Between zero and P, the output rises as S. Between P and two P, it falls as two P minus S. Between two P and three P, it rises again as S minus two P. Above three P, it saturates at P. Divide raw Y by positive P to obtain a normalized output from zero to one. The turning points at P and two P are central to the single-node XOR example. Changes at segment boundaries are continuous here, though slopes change abruptly.

PATRICK: Sarah, why does a falling segment matter so much?

SARAH: Because it lets a larger weighted sum lower the output rather than always increase it. For XOR, both inputs high can land on a valley even though each single-high input reaches a peak. The exact region and scale determine the result.

RACHEL: Notice that these breakpoints describe a continuous piecewise-linear function; the corners change slope, rather than producing a discontinuity in real mode.

=== 14 — Boolean truth tables: why there are exactly 16 ===

PATRICK: A two-input Boolean truth table has exactly four rows: zero-zero, zero-one, one-zero, and one-one. Each row can independently be assigned zero or one. That creates two to the fourth power, or sixteen, possible Boolean functions. The preset library contains all sixteen, from constant FALSE and TRUE to AND, OR, XOR, XNOR, NAND, NOR, input pass-throughs, negations, and directional implications. These are exact at the binary corners using the five-region activation. Between the corners, the response may be graded and can take a range of values.

RACHEL: Patrick, does all sixteen mean every function of arbitrarily many inputs?

PATRICK: No. Exactly sixteen applies to the four input combinations of two Boolean variables, so there are two to the fourth possible truth tables. The one-node catalog demonstrates those sixteen cases. It says nothing by itself about a full network's capacity on a larger task.

SARAH: The reason we spend time on every one of the sixteen functions is not memorization. It is understanding how one fixed node architecture can take on a wide variety of behaviors.

=== 15 — Gate 01 · FALSE ===

RACHEL: The constant false gate returns zero for every binary input pair. Its two weights and bias are zero, so all four weighted sums sit in the zero-output region. The expected output sequence is 0, 0, 0, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

JUSTIN: Why bother giving constant functions a place among the sixteen?

RICHARD: They are valid Boolean functions and useful controls. Including them means the catalog is mathematically complete, not only a selection of familiar named gates.

JUSTIN: A constant-zero output is an important baseline. If it changes under a

Boolean input, we have changed the parameters, the mode, or the premise of the experiment.

=== 16 — Gate 02 · AND ===

JUSTIN: The AND gate activates only when both inputs are one. At P equal to sixteen, choose weights sixteen and sixteen and bias minus sixteen. The sums at zero-zero, zero-one, one-zero, one-one are minus sixteen, zero, zero, and sixteen. Only the last reaches output sixteen. The expected output sequence is 0, 0, 0, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

SARAH: What single change gives us an exact one only at one-one?

RACHEL: Here the negative bias holds the single-high cases at zero. The combined contribution moves the both-high case onto the rising peak at P.

PATRICK: Notice the difference between a correct truth table and a useful continuous response. The AND preset is exact at the corners; the surface between them is another question to investigate.

=== 17 — Gate 03 · A AND NOT B ===

PATRICK: This directed gate is true only when input one is high and input two is low. At P sixteen, W one is sixteen, W two is minus sixteen, bias zero. The four sums are zero, minus sixteen, sixteen, and zero. The negative weight for the second input suppresses the output in both cases where that input is one. The expected output sequence is 0, 0, 1, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RICHARD: How does the node express which input is being negated?

JUSTIN: The sign of the associated weight determines its direction in the weighted sum. Swapping the weights swaps which single-high condition is accepted.

RICHARD: Here the negative second weight makes one-zero the privileged corner. Reversing the inputs exposes the directionality immediately.

=== 18 — Gate 04 · A (X1) ===

SARAH: The first input is simply passed through at the two Boolean corners. At P sixteen, set W one to sixteen, W two to zero, and bias zero. Both cases with X one equal to one produce normalized output one, regardless of X two. The expected output sequence is 0, 0, 1, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

PATRICK: Is passing through an input really one of the sixteen logic functions?

SARAH: Yes. A Boolean function does not need to depend on both inputs. Ignoring one input is perfectly valid, as are the constant functions.

RACHEL: This projection function offers a useful calibration: changing X two should not affect the Boolean output while its weight remains zero.

=== 19 — Gate 05 · NOT A AND B ===

RICHARD: This is the mirror of A AND NOT B. A negative weight on X one and a positive weight on X two produces a one only at zero-one. At P sixteen the weights

are minus sixteen and plus sixteen with zero bias. The expected output sequence is 0, 1, 0, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RACHEL: Why compare this immediately with the previous directional gate?

PATRICK: Because it isolates the meaning of signed weights. The same node geometry, reversed between the two inputs, reverses the accepted direction.

SARAH: The response is the reverse directional case from A AND NOT B. Comparing both plots helps make the signs of the weights intuitive.

=== 20 — Gate 06 · B (X2) ===

RACHEL: This pass-through gate copies the second Boolean input. The first weight is zero, the second is P, and the bias is zero. The binary table reads zero, one, zero, one. The expected output sequence is 0, 1, 0, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

JUSTIN: What does this tell us about the role of either weight?

RICHARD: A zero weight removes that input from the calculation. This simple baseline helps explain why the more complex gates require interacting contributions.

JUSTIN: We have removed the first input entirely by giving it zero weight. Compare the current arithmetic trace before and after you change X one.

=== 21 — Gate 07 · XOR ===

JUSTIN: Exclusive OR is true when exactly one of the two inputs is one. With W one and W two both equal to P, bias zero, the sums are zero, P, P, and two P. The five-region activation outputs zero, P, P, and zero. A single node therefore implements the exact XOR table at the four binary corners. The expected output sequence is 0, 1, 1, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

SARAH: Does the output remain Boolean when the inputs are half values?

RACHEL: Not necessarily. Along the interior the response is graded, and depending on the sum it can fall on a rising or falling segment. The binary table is only the corners.

PATRICK: The four XOR corners are exact, even though the interior is not required to be Boolean. That separation is essential for understanding the surface.

=== 22 — Gate 08 · OR ===

PATRICK: OR is true for either single-high input and for both high. At P sixteen, use weights thirty-two and thirty-two and bias minus sixteen. The sums are minus sixteen, sixteen, sixteen, and forty-eight. The activation outputs zero, sixteen, sixteen, and sixteen. The expected output sequence is 0, 1, 1, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RICHARD: Why are the OR weights larger than the AND weights here?

JUSTIN: They place either single-high case on the first peak and both-high case on the high plateau, despite the negative bias applied to all four sums.

RICHARD: The OR example shows that the second rising region or plateau can be just as useful as the first peak when assigning truth-table outcomes.

=== 23 — Gate 09 · NOR ===

SARAH: NOR returns one only when both inputs are zero. Set both weights to minus P and bias to P. The sums at the four corners are P, zero, zero, and minus P, so the normalized vector is one, zero, zero, zero. The expected output sequence is 1, 0, 0, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

PATRICK: Does NOR require a special negative activation function?

SARAH: No. The activation is unchanged; negative weights and positive bias place the four sums on the desired segments.

RACHEL: NOR is not a new node type. The same activation combined with signed coefficients and an offset reproduces its complement-like behavior.

=== 24 — Gate 10 · XNOR ===

RICHARD: Exclusive NOR is true when the two binary inputs agree. Use W one and W two equal to P and bias equal to P. The four sums are P, two P, two P, and three P. The five-region function yields P, zero, zero, and P. The expected output sequence is 1, 0, 0, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RACHEL: Why does a positive bias invert the XOR truth pattern here?

PATRICK: It shifts the four sums onto a different sequence of the non-monotonic segments. It is not simple numerical negation of weights; the change uses the activation shape.

SARAH: XNOR and XOR are a particularly revealing pair. A bias shift changes where the four sums intersect the existing five activation regions.

=== 25 — Gate 11 · NOT B ===

RACHEL: Negating the second input takes weight one equal to zero, weight two equal to negative P, and bias equal to P. The vector is one, zero, one, zero. The expected output sequence is 1, 0, 1, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

JUSTIN: What lesson does this simple inversion offer before moving into implications?

RICHARD: Bias can establish a default one and a negative weight can subtract the second input from that default, while the other input remains irrelevant.

JUSTIN: The NOT B example isolates the bias and one negative coefficient. It is small enough to verify completely with mental arithmetic.

=== 26 — Gate 12 · A OR NOT B ===

JUSTIN: The gate A OR NOT B is false only when the first input is zero and the second is one. At P sixteen, the normalized parameters are two, one, and one; the binary output vector is one, zero, one, one. The expected output sequence is 1, 0, 1, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

SARAH: Why is this called a directional implication-like function?

RACHEL: It is true for almost every combination but excludes one particular directional case. Its complement or swapped-input version helps reveal the asymmetry.

PATRICK: The excluded zero-one corner is why this gate is directional. Reverse the inputs and inspect the truth table instead of relying on the name alone.

=== 27 — Gate 13 · NOT A ===

PATRICK: Negation of the first input uses W one equal to negative P, W two equal to zero, and bias equal to P. Its vector is one, one, zero, zero, because only X one matters. The expected output sequence is 1, 1, 0, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RICHARD: Could this be useful for checking the bias control?

JUSTIN: Yes. The baseline one at zero input comes from bias, while a negative contribution from X one removes it at the high state.

RICHARD: Inverting X one is another simple test for whether input order and signs are handled consistently throughout the interface.

=== 28 — Gate 14 · NOT A OR B ===

SARAH: The inverse directional implication, NOT A OR B, is false only for first input one and second input zero. The representative normalized coefficients are one, two, and one. The resulting binary vector is one, one, zero, one. The expected output sequence is 1, 1, 0, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

PATRICK: What comparison makes the difference between the implication functions obvious?

SARAH: Keep the truth-table order fixed at zero-zero, zero-one, one-zero, one-one, then compare which of the two single-high cases is uniquely false.

RACHEL: This is the complementary directional case. Compare the two asymmetric truth tables and the position of their active markers.

=== 29 — Gate 15 · NAND ===

RICHARD: NAND is one for every binary combination except both inputs high. With P sixteen, set both weights to minus thirty-two and bias to forty-eight. The sums fall at three P, P, P, and minus P, giving normalized outputs one, one, one, zero. The expected output sequence is 1, 1, 1, 0, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

RACHEL: Does NAND require combining two separate nodes?

PATRICK: Not in this example. A single node already reproduces the four NAND outputs. The negative weights and large positive bias exploit multiple activation regions.

SARAH: NAND has just one false corner. Its parameter placement is a useful reminder that larger or negative weights can still produce exact Boolean results.

=== 30 — Gate 16 · TRUE ===

RACHEL: The constant true function has both weights equal to zero and bias equal to P. Every binary combination maps to the same weighted sum P, so every normalized output is one. The expected output sequence is 1, 1, 1, 1, ordered by inputs zero-zero, zero-one, one-zero, and one-one.

JUSTIN: Why finish the catalog with TRUE?

RICHARD: Together with FALSE, it confirms the library includes the edge cases. The claim is about all sixteen possible functions of two Boolean variables.

JUSTIN: A constant-one gate completes the catalog. The sixteen rows represent every possible mapping from four binary input combinations to binary output.

=== 31 — The four graded example presets ===

JUSTIN: Four additional presets demonstrate non-Boolean graded behavior. Soft XOR uses normalized weight ratios three quarters and three quarters with zero bias. Graded AND uses one half and one half with zero bias. The asymmetric example uses ratios one and one half, again with zero bias. Graded NOR uses negative one-half weights for both inputs and bias equal to P. Each preset invites inspection of intermediate outputs, but those values are not calibrated probabilities. A single three-parameter node also does not represent every imaginable continuous two-input function. Its flexibility should be studied within the specified five-region mathematical form.

SARAH: Rachel, why include graded presets if the Boolean functions are the headline?

RACHEL: Because real measurements are not always zero or one. The application shows how intermediate outputs arise between Boolean corners, and how two nodes that agree at the corners can still respond differently in the interior. Those are engineering questions, not merely logic puzzles.

PATRICK: Try the graded presets at X one equals point four five and X two equals point four, then compare the intermediate output to the exact Boolean corner values.

=== 32 — Fixed INT: the implemented arithmetic path ===

RICHARD: Fixed INT is the demonstration's specified scaled-integer arithmetic, not a measured microcontroller implementation. The input is clamped to zero through one and mapped to an integer by rounding P times the input. Integer weights and bias are rounded. The dot product is W one times X one integer plus W two times X two integer. Then S is the rounded dot product divided by P, plus bias. Apply the same five-region activation at breakpoints P, two P, and three P. The browser uses JavaScript rounding, including its specific behavior for signed half-integer values; a hardware port must select and document a matching rounding convention. Real mode instead evaluates the original fractional inputs directly, without that intermediate integer quantization.

RACHEL: Justin, does Fixed INT mean the browser is actually running on an embedded INT8 microcontroller?

JUSTIN: No. It means the numerical evaluation follows a specified integer

quantization rule. It is useful for inspecting rounding and discontinuities. It cannot be used as evidence of processor instruction counts, latency, or energy consumption.

RICHARD: We should keep that implementation boundary visible throughout our examples. A pedagogical integer calculation is not a hardware performance measurement.

=== 33 — Activation scale study · P = 1 ===

JUSTIN: At P equals one the activation turns at sums one, two, and three. Exact Boolean presets can still be represented because the normalized coefficients are integers here. However, integer-mode input resolution is extremely coarse: a fractional input is rounded to either zero or one. A small change in the input may not change the integer representation at all.

RACHEL: Patrick, why can the exact corners be correct while interior behavior is so coarse?

PATRICK: At the corners, quantized inputs are exact zero or one; between them, a one-step grid has only two levels. One must not infer the quality of interior approximation from the four Boolean rows.

SARAH: With P equal to one, the rounded input values occupy only the two endpoint levels. The exact Boolean corners can still be correct even though the interior grid is extremely coarse.

=== 34 — Activation scale study · P = 2 ===

PATRICK: At P equals two, integer-mode input levels are zero, one-half, and one after normalization. That is still a coarse lattice, but it introduces a midpoint the previous scale could not represent. Compare the same nominal fractional input in real and fixed mode and watch its integer encoding and arithmetic trace.

JUSTIN: Richard, is a midpoint enough to make this equivalent to real arithmetic?

RICHARD: No. There are still only three representable levels per input. The stair-step effect remains, and parameter rounding can introduce additional differences.

JUSTIN: At P two, there are only three normalized input levels: zero, one half, and one. Watch what happens when an input crosses a rounding threshold.

=== 35 — Activation scale study · P = 4 ===

RACHEL: At P equals four, the input lattice has increments of one quarter. A requested input of point four five maps to the nearest quarter, which is point five. For asymmetric weights, the change may shift the weighted sum into a different activation region. The truth table corners can remain exact while the interior surface shows quantized steps.

SARAH: Rachel, what is the cleanest experimental comparison here?

RACHEL: Keep one preset and its normalized parameter ratios, hold the requested input values constant, then switch between exact and fixed modes. Note the encoded inputs as well as the output.

PATRICK: At P four, the normalized grid spacing is one quarter. Compare the

quantized values displayed in the arithmetic trace with the continuously selected inputs.

=== 36 — Activation scale study · P = 8 ===

SARAH: P equals eight offers eighth-step input representation in the demonstrated integer path. A fractional signal closer to a particular eighth rounds there. This can make the surface look finer than at P equals four, but display density is not the same as numerical accuracy. Compare exact weighted sums with integer-scaled sums and observe any region changes around P and two P.

RICHARD: Justin, why is a smooth-looking surface sometimes misleading?

JUSTIN: Because a viewer may interpolate between sampled triangles. The actual integer function can have discrete jumps. Always inspect the trace for the selected point.

RICHARD: At P eight, each integer input step represents one eighth. The truth table can remain identical while the interior fixed-point surface becomes more detailed.

=== 37 — Activation scale study · P = 16 ===

RICHARD: P equals sixteen is the default and the convenient reference for most examples. The XOR preset uses weights sixteen and sixteen with zero bias. The integer input grid advances in increments of one sixteenth. In real mode, X one point four five and X two point four produce a continuous weighted sum; in fixed mode each requested input first maps to a nearby integer level.

PATRICK: Sarah, should viewers make this their baseline?

SARAH: Yes. Start with P sixteen to reproduce the guide examples. Once the calculation makes sense, change one scale choice at a time and document whether proportional rescaling was enabled.

RACHEL: At P sixteen, inspect how the two fractional test inputs round to their stored integer values. The coefficients and arithmetic mode still determine the final result.

=== 38 — Activation scale study · P = 32 ===

JUSTIN: At P equals thirty-two, the integer input lattice is finer, but weight and bias ranges also grow proportionally. If the rescaling checkbox is selected, the preset coefficients scale with P and the normalized real-mode response often remains identical. If you leave the coefficients unchanged while reducing the activation breakpoints, the response shape can change substantially.

RACHEL: Patrick, what does a higher P not establish?

PATRICK: It does not establish better performance on hardware, faster execution, lower power, or guaranteed accuracy. Those require a separately implemented and measured system.

SARAH: At P thirty-two, the fixed-point input grid is finer, but the integer dot product can also grow. More grid levels are not an independent hardware benchmark.

=== 39 — Activation scale study · P = 64 ===

PATRICK: At P equals sixty-four, the input quantization increment is one sixty-fourth of the normalized range. The application still uses the same five-region activation and the same four Boolean corner definitions. Greater integer resolution may make the fixed surface more closely track the real-mode surface for some values, but not for every possible rounding configuration.

JUSTIN: Richard, why keep an eye on parameter bounds at this scale?

RICHARD: The UI constrains each coefficient between negative three P and positive three P. A valid experiment must respect those ranges, particularly when changing P without proportional rescaling.

JUSTIN: At P sixty-four, the fractional input grid is finer still. Compare Real and Fixed INT numerically, and check coefficient ranges before changing scale without rescaling.

=== 40 — Display toolbar, presets, and learning guide ===

JUSTIN: The upper toolbar has three groups of display preferences: color theme, font family, and overall text size. The default font is IBM Plex Sans. Larger text and high-contrast themes help when using a smaller monitor or presenting the laboratory to an audience. These choices change appearance, not the underlying parameters or calculations.

SARAH: Rachel, why include these options in a mathematical demonstration?

RACHEL: Because the values must be readable. The truth table, breakpoint labels, and arithmetic trace are only useful when viewers can inspect them comfortably. Choose a font and size that work for your display.

PATRICK: The numerical results remain the same when you change the theme or font. Treat those controls as accessibility tools, not experiment variables.

=== 41 — Panel 1: node parameters ===

RICHARD: The first panel contains the two weights W one and W two, the bias B, the activation scale P, arithmetic-mode controls, and the proportional-rescaling checkbox. Its fractional-input sliders X one and X two range from zero to one. The formula and explanatory line update as controls change. Each slider changes one aspect of the weighted sum. The weights determine the contribution from each input; the bias shifts the whole weighted-sum field; and P determines the activation breakpoints and scaling of normalized output.

RACHEL: Justin, which parameter changes the response most dramatically?

JUSTIN: There is no universal answer. Weights affect the two inputs differently; bias shifts every corner's weighted sum together. Start from a known exact preset, move one parameter at a time, and watch which points cross a region boundary.

RICHARD: Try a positive, a negative, and a zero value for each coefficient separately. The arithmetic trace will reveal the effect more clearly than the colors alone.

=== 42 — Panel 2: five-region activation plot ===

RACHEL: 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 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. Sometimes, 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. At a weighted sum of P, the raw output equals P. At a weighted sum of two P, the raw output is zero.

PATRICK: Sarah, what should we learn from the four colored markers?

SARAH: Each marker represents the input pair zero-zero, zero-one, one-zero, or one-one. Their horizontal positions correspond to weighted sums and their heights to the activation output. Watching the markers slide is a direct explanation for a changing truth table.

RACHEL: Every marker represents one of the four Boolean sums. The colored markers make changes to region assignments visible immediately.

=== 43 — Panel 3: live truth table and classification ===

PATRICK: The interactive truth table always evaluates the four binary input pairs independent of the two input sliders. Its columns show each input pair, weighted sum S, activation region, normalized output Y divided by P, and a response indicator. Exact Boolean classification is displayed only when all four outputs match a known truth table within numerical tolerance. When the response is graded, the application shows the nearest Boolean pattern and its mean absolute error instead of claiming an exact logical match. Clicking a table row transfers its binary coordinates to the selected inputs.

RACHEL: Patrick, why do the truth-table outputs remain fixed when the fractional input sliders move?

PATRICK: The table deliberately evaluates the four binary corners, independently of the selected point. The sliders move the point being inspected and therefore the trace and surface marker. This separation lets you distinguish a gate's binary classification from its behavior between binary inputs.

SARAH: The truth table is evaluated at the four Boolean corners, while the fractional-input sliders explore arbitrary points between those corners.

=== 44 — Panel 4: arithmetic trace ===

SARAH: The arithmetic trace is the most direct numerical debugging view. A single node has two weights, a bias, and two selected input coordinates. Those five values determine the weighted sum. The trace then identifies each input, its weighted contribution, the weighted sum including bias, the active activation region, raw output Y, and normalized output Y divided by P. In Fixed INT mode it also exposes the quantized arithmetic path. To investigate a particular calculation, select a truth-table row or move the input sliders. First compare the displayed weighted contribution against W one times X one plus W two times X two. Then confirm S after adding B. Match S to the active region in panel two, and calculate its raw Y and normalized Y divided by P. Finally, if Fixed INT is active, check which input values were quantized and where rounding occurred in the division by P. This

sequence distinguishes an arithmetic misunderstanding from a misleading interpretation of the plotted surface.

JUSTIN: Richard, can we verify a number without trusting the graph colors?

RICHARD: Absolutely. Inspect W one, W two, the input values, and the bias. Calculate the weighted sum, identify its activation region, evaluate that region's formula, and divide by P. The arithmetic trace exposes every one of those stages.

JUSTIN: To reproduce the trace, write down five values: W one, W two, B, X one, and X two. Compute the weighted sum, identify the activation region, and divide the raw output by P.

=== 45 — Panel 5: the original projected 3D response surface ===

JUSTIN: Version 1.3.0 retains the original 2D Canvas graph, which uses a fixed pseudo-3D projection of Y divided by 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 selected input location. A 2D canvas implementation does not require WebGL and is intentionally preserved as a widely compatible graphics 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. In the panel five heading, click Explore interactive three D 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.

SARAH: Rachel, why keep the older pseudo-three-dimensional image now that WebGL exists?

RACHEL: It loads quickly, remains easy to interpret, and does not require WebGL support. The separate explorer adds camera control and deeper inspection without making the regular laboratory dependent on GPU features. The two views complement each other.

PATRICK: The original panel five surface remains useful as a quick spatial overview and a fallback when advanced graphics are unavailable.

=== 46 — Panel 6: illustrative local candidate search ===

RICHARD: The sixth panel implements an illustrative, derivative-free coordinate search. Choose any of the sixteen Boolean truth tables as the target. At each step, the search evaluates up to six permitted neighboring candidates: plus or minus a step in W one, W two, and bias. It scores the four binary cases using mean squared error and accepts the best candidate only when it strictly improves the current score. Try One or Try Ten controls repeat that process. The local method can stall; it is not the patented full chain-isolation algorithm and must not be presented as proof of training performance.

RACHEL: Justin, does Try Ten mean it has trained a neural network?

JUSTIN: No. It performs repeated greedy, local candidate comparisons on this one node. It can illustrate derivative-free parameter updates and reductions in mean squared error, but it does not implement the full chain-isolation methodology or train a multi-layer IGL model.

RICHARD: Each Try button evaluates only local parameter alternatives for a selected target gate. It cannot guarantee a global optimum.

=== 47 — Opening, positioning and closing the explorer ===

RACHEL: The new WebGL explorer opens from the button at the top of panel five. It is an in-page window that can be resized or maximized on supported desktop browsers, not a separate website or a replacement for the original chart. Its toolbar includes orthographic or perspective projection, six color palettes, four mesh resolutions, a cross-section dropdown, and independent visibility toggles. The main viewing area supports camera movement and point inspection. A numerical readout below the surface follows the selected or hovered input. The explorer is synchronized with the laboratory's core calculation, and closing it returns the viewer to the regular six-panel interface.

PATRICK: Sarah, is the explorer a separate website?

SARAH: No, it is a resizable in-page dialog tied to the same model. You can dismiss it without losing the parameter state. That connection is valuable because every visual sample can be traced back to the exact laboratory calculation.

RACHEL: When the 3D explorer opens, the six-panel laboratory remains underneath it. You can return to the original view at any time.

=== 48 — Camera controls: orbit, pan, zoom and reset ===

PATRICK: The WebGL viewer's mouse and touch controls let the user manipulate the camera. Drag to orbit the response surface around the target. Hold Shift while dragging to pan. Use the mouse wheel or pinch gesture to zoom, or the plus and minus buttons to change magnification. Reset view restores a useful default camera position. The camera is independent of model parameters: rotating the scene never changes the IGL calculation, and zooming does not modify Y divided by P.

RACHEL: Patrick, what is the difference between orbiting and panning?

PATRICK: Orbiting changes the camera direction around the response surface; panning shifts the camera target. Zoom changes the scale of the presentation. Reset view is the fastest way back if a dragged perspective becomes confusing.

SARAH: Try orbiting first, then panning to change the framing. If you lose your bearings, reset the view before judging the geometry.

=== 49 — Projection, color palettes and surface resolution ===

SARAH: The projection selector offers orthographic and perspective displays. Orthographic projection keeps parallel edges parallel and is useful for comparing heights without foreshortening. Perspective projection provides depth cues and can make the surface feel more spatial. Six independent color palettes are available: Spectrum, Ocean Cyan, Thermal, Terrain, Graphite, and Cobalt Gold. These only color the normalized output and do not change any numeric values. Surface resolutions use thirty-two, sixty-four, ninety-six, or one-hundred-twenty-eight samples per direction. Increasing resolution can reveal more detail and can increase graphics cost; the numeric evaluator remains the same.

JUSTIN: Richard, is a thermal color palette showing temperature or measured power?

RICHARD: Not at all. Every palette maps normalized output height to colors for

visibility. Spectrum, ocean, thermal, terrain, graphite, and cobalt-gold are display choices. The numerical output is unchanged, and a chart color should never be mistaken for a physical energy measurement.

JUSTIN: A vivid palette does not make one point numerically larger unless the chosen legend defines that mapping. Numeric inspection remains authoritative.

=== 50 — WebGL visibility controls ===

JUSTIN: The WebGL visibility options are independent visual layers. Axes and ticks give spatial reference, while numeric labels annotate coordinates. The floor grid helps reveal slopes and relative height. Wireframe overlays the sampled triangles, making surface tessellation visible. Shading changes how lighting reveals facets; the selected-input marker identifies the current X one and X two. Automatic orbit moves the camera for a demonstration, but does not alter any numerical values.

SARAH: Rachel, what value do the numeric labels add when the shape is already visible?

RACHEL: They anchor interpretation. Without axes and ticks, a ridge might look impressive but its coordinates are ambiguous. With labels, the floor grid, a selected-input marker, and optional wireframe, engineers can inspect geometry and then cross-check values numerically.

PATRICK: Turn labels and axes on for precise discussion; turn visual overlays off when you want to examine the geometry with less clutter.

=== 51 — Interactive point inspection and pinning ===

RICHARD: The WebGL viewer determines the input coordinates by ray-picking a point on the visible surface triangles. It then sends those coordinates to the same numerical IGL core used by the laboratory. The inspector displays X one, X two, S, raw Y, normalized Y divided by P, and the active region. To try this, move the pointer over the colored surface rather than the dark background. Watch the hover-inspection readout as the coordinates change. Next click a point to pin it; the marker and numerical values remain in place while you move the pointer. Choose Use point in lab to transfer the selected inputs to the main laboratory sliders, and compare the updated arithmetic trace and response graph. Select Clear pin when you want to return to unrestricted hover inspection. The displayed result is calculated from the selected coordinates; it is not inferred from the pixel color.

RACHEL: Justin, how do I know a highlighted point is an actual evaluated result?

JUSTIN: The viewer ray-picks a point on the drawn triangles, but the displayed X values are passed back through the shared IGL core. The inspector reports calculated S, Y, Y divided by P, and region. Clicking pins that point; using it in the lab transfers its input coordinates.

RICHARD: A pinned point is an evaluated coordinate, not a pixel color guess. Compare X one, X two, S, Y, Y/P, and region.

=== 52 — Cross-section tools and spatial interpretation ===

RACHEL: The Cross-section selector offers Off, X one fixed, and X two fixed. It draws a line across the response surface while one input remains constant at the laboratory's currently selected value. This gives us a one-dimensional path through the full three-dimensional geometry. For a useful experiment, load XOR in Exact

real mode, set X one to zero point five, and open the WebGL explorer. Select X one fixed and rotate the camera until the slice is clearly visible. Along that path, as X two moves from zero to one, the normalized output first rises from one-half to one and then falls to one-half. Move the held input in the main laboratory, reopen the explorer, and compare the new slice. Then select X two fixed to explore the other direction. The cross-section is a visual overlay on the existing surface, not a separate plotted curve. Use the point inspector when you need exact numerical values.

PATRICK: Sarah, what question does a cross-section answer that orbiting cannot?

SARAH: A slice fixes one input and varies the other. It turns a complicated surface into a specific one-dimensional path through the non-monotonic activation. This makes rising, falling, and plateau regions easier to identify and discuss.

RACHEL: Cross-sections turn an attractive three-dimensional plot into a testable one-dimensional profile along a chosen input path.

=== 53 — Resizing, fullscreen, export and model synchronization ===

PATRICK: First, consider the window and its viewport. The explorer is a resizable dialog inside the webpage, rather than a separate browser window. On desktop systems that support CSS resizing, drag its lower-right corner. Maximize and Restore change the size of the page overlay. Fullscreen asks the browser for a fullscreen view and may be unavailable under particular browser policies. The surface redraws as the viewport changes. To save a useful image, choose the desired camera orientation, projection, palette, mesh density, and visibility settings. You can also pin an informative point before selecting Save PNG. The application combines the WebGL image with its numeric overlay into a local PNG file, normally named MLiglon underscore IGL underscore 3D underscore Surface dot PNG. The exported image represents the current view; it does not contain the sampled mesh as a 3D file or the complete numeric dataset as CSV or JSON. Keep a copied parameter configuration alongside the image for reproducibility.

RACHEL: Patrick, what exactly will Save PNG contain?

PATRICK: It captures the current rendered view and supporting numerical display as a static image, subject to the browser's capabilities. Keep a textual configuration alongside it, because an image alone cannot always document every parameter, arithmetic setting, or zoom choice.

SARAH: Use the PNG export when documenting the geometry, but accompany it with the parameter values and arithmetic mode for reproducibility.

=== 54 — Performance and rendering considerations ===

SARAH: 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 times 32 or 64 times 64, disable wireframe and auto orbit, exit 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.

JUSTIN: Richard, is the highest mesh resolution always the most scientifically correct?

RICHARD: No. It is visually finer, but the underlying Fixed INT function can jump between quantized levels. A finer sampled mesh may show those steps more clearly; it does not replace exact per-point arithmetic. On a slower device, lower resolution can improve navigation.

JUSTIN: For correctness, compare direct evaluation at exact coordinates. Mesh density improves rendering detail, not the underlying model.

=== 55 — Experiment 1: reproduce XOR with one node ===

JUSTIN: To reproduce XOR, set P to sixteen and select Exact real arithmetic. Click the XOR Boolean preset. The two weights should both display sixteen and the bias should read zero. In the truth table, verify the normalized outputs zero, one, one, zero in row order. Then select each row in turn and watch its colored marker on the activation curve. The four weighted sums are zero, sixteen, sixteen, and thirty-two. The one-input-high cases reach the first peak, but both inputs high reach the valley at two P. Open the WebGL surface to confirm that the interior is not simply a Boolean step function.

SARAH: Rachel, why is XOR the right place to begin the full experiment series?

RACHEL: It is a familiar linearly inseparable binary truth table. With both weights equal to P and bias zero, the four sums land at zero, P, P, and two P. The activation converts them to zero, P, P, and zero. That is the central illustration of a non-monotonic node.

PATRICK: Repeat the four Boolean inputs in row order. The sequence zero, one, one, zero is the simplest complete check of XOR.

=== 56 — Experiment 2: exact XNOR and the effect of bias ===

RICHARD: For XNOR, retain P equal to sixteen and Exact real arithmetic. Select the XNOR preset, and verify both weights equal sixteen while the bias is also sixteen. The weighted sums at binary corners are sixteen, thirty-two, thirty-two, and forty-eight. This places the first corner on the first peak, the two single-high corners at the valley, and both-high on the second rise's endpoint. The normalized truth table is therefore one, zero, zero, one. Compare this screen with XOR using the same camera orientation; the surfaces move because the bias shifts every weighted sum to a different position.

RACHEL: Justin, how can adding bias transform the same positive weights into XNOR?

JUSTIN: The bias moves all four weighted sums to the right. With W one and W two equal to P and bias equal to P, the sums occupy P, two P, two P, and three P. Peaks at the ends and a valley in the middle yield one, zero, zero, one after normalization.

RICHARD: The geometry matters more than the label. The XNOR bias shifts where the same four input combinations lie along the activation.

=== 57 — Experiment 3: AND, NAND, NOR and OR ===

RACHEL: Compare four familiar functions in sequence. Choose AND and verify that only one-one produces a one. Choose NAND and observe that the result is one everywhere except one-one. Choose NOR and observe that the only positive corner is zero-zero. Finally choose OR, where every input pair except zero-zero produces one. Pay attention to the parameter display each time. The node never changes architecture; signed weights and bias move the four input sums onto different regions of the same activation curve. Use the truth table as the formal test of these four functions rather than identifying them from the surface shape alone.

PATRICK: Sarah, is one logic gate fundamentally better than the others?

SARAH: No. They are different target functions. The educational value is that a single structural node can be parameterized into all of them. Our job is to examine why the active regions and parameter signs produce each table—not rank the Boolean functions.

RACHEL: The same non-monotonic function is reused throughout. The preset parameters determine which Boolean mapping you obtain.

=== 58 — Experiment 4: asymmetry versus symmetry ===

PATRICK: Choose the Asymmetric graded example at P sixteen in Exact real mode. Its weights are sixteen and eight and bias is zero. Compare the zero-one and one-zero rows. The lower second weight gives a smaller contribution for a second-input-only event, so the response differs from a first-input-only event. Move the input sliders to fractional inputs and watch the weighted trace. In WebGL, orbit around the surface and enable the X one fixed cross-section. Then swap the two weights manually and see how the surface's directional preference reverses. This illustrates how a node can respond differently to its two sources of information.

RACHEL: Patrick, why does asymmetry matter if I can test only two input combinations?

PATRICK: When W one and W two differ, the zero-one and one-zero cases can land in different positions along the activation curve. The 3D surface is no longer symmetric across the diagonal. That helps reveal how individual input contributions combine.

SARAH: A non-symmetric surface is a useful sign to compare input order. Swap X one and X two and check the weighted sum directly.

=== 59 — Experiment 5: inspect graded logic and continuous inputs ===

SARAH: Choose Soft XOR and leave arithmetic mode set to Exact real. Inspect each of the four normalized truth-table outputs and notice the intermediate values at the single-high corners. These are graded responses rather than exact Boolean zeros and ones. Now select X one at point two five and X two at point seven five; note S, activation region, raw Y, and normalized output. Reverse the two inputs and compare. In the 3D explorer, hover across the surface to inspect the same response continuum. A value between zero and one is an output level, not automatically a calibrated probability.

JUSTIN: Richard, can a value of one-half be called half true?

RICHARD: For this demonstration it is best described as a normalized graded response, not a calibrated probability. Unless a model is trained and calibrated for a particular application, the Y divided by P value should not be interpreted as

a probability of an event.

JUSTIN: A graded output should not be presented as a calibrated probability. It is a normalized activation response at that coordinate.

=== 60 — Experiment 6: Real versus Fixed INT ===

JUSTIN: Choose the Asymmetric preset, P sixteen, Exact real mode, with X one equal to point four five and X two equal to point four. The weights are sixteen and eight, and bias zero. The continuous contributions are seven point two and three point two, so S equals ten point four. That value lies in rising region two; raw Y is ten point four, and normalized Y divided by P equals point six five. Now switch to Fixed INT. X one becomes the integer seven, because sixteen times point four five rounds to seven. X two becomes six, because sixteen times point four rounds to six. The integer dot product is sixteen times seven, plus eight times six, which equals one hundred sixty. Divide by sixteen and round to obtain S equals ten. Thus Y divided by P is ten over sixteen, or point six two five. The difference from real mode is point zero two five.

SARAH: Rachel, show us the important difference in an arithmetic trace.

RACHEL: In real mode with asymmetric weights sixteen and eight, fractional inputs point four five and point four produce a weighted sum of ten point four. In Fixed INT, the quantized inputs seven and six produce a sum of ten and normalized output point six two five. In Exact real mode, the sum is ten point four and the normalized output is point six five. Both remain in rising region two, but the output values differ because of the specified rounding steps.

PATRICK: In other examples, quantization can even move the sum into a different activation region. This example stays within rising region two. Always inspect the actual rounding rule and the trace, rather than assuming quantization necessarily changes the active region.

=== 61 — Experiment 7: study activation scale and parameter rescaling ===

RICHARD: Start with XOR at P sixteen and keep the Scale W one, W two, B option selected. Increase P to thirty-two. The two weights become thirty-two while their ratios to P remain one; the exact Boolean truth-table outputs stay the same. This is a proportional rescaling demonstration, not new training. Next disable the rescaling option and change P to eight. Observe how the absolute weights and bias no longer track the shifted activation boundaries, possibly changing the truth table and graded surface. Restore a known preset before concluding which effect came from the scale itself and which came from unscaled parameters.

RACHEL: Justin, if I double P and weights, have I automatically improved precision?

JUSTIN: No. Rescaling can preserve the mathematical normalized response in real mode. In Fixed INT, grid spacing and rounding may change, but the demonstration does not measure effective hardware accuracy. Increasing P is not automatically equivalent to reducing model error on a real dataset.

RICHARD: Scaling P and scaling all coefficients proportionally can preserve normalized geometry; simply increasing P proves no accuracy or energy advantage.

=== 62 — Experiment 8: greedy search toward a different gate ===

RACHEL: Start with exact XOR at P sixteen in real mode. In panel six choose AND as

the target function. Note the starting mean squared error between the current four outputs and the four desired AND bits. Click Try One and inspect the accepted parameter change, if any, and the number of candidates evaluated. Then click Try Ten and observe how the score evolves. The search evaluates up to six permitted nearby parameter alternatives per step and keeps only a strict improvement. It can stop at a local solution. Finally load the exact AND preset and compare the results; that preset is not proof that the local search would necessarily discover it.

PATRICK: Sarah, why might the local search stop short of an exact AND preset?

SARAH: It accepts only strict improvements among nearby, permitted coordinate changes. A better solution might require a move that temporarily makes the score worse, or several changes together. The exact preset provides a target reference, not a guarantee that this local routine will discover it.

RACHEL: If search stalls, inspect which one-step neighbors were eligible. A local optimum is not necessarily the globally best set of parameters.

=== 63 — Experiment 9: use cross-sections to find non-monotonic behavior ===

PATRICK: For a cross-section experiment, load XOR in Exact real mode, use P sixteen, and fix X one at one half. Open WebGL and select X one fixed. As X two moves from zero to one half to one, the weighted sum passes through eight, sixteen, and twenty-four. The normalized response is one half, one, and one half. This particular slice rises to one peak and then falls; it does not climb again. Rotate the camera to inspect the ridge, compare orthographic and perspective projections, and inspect the exact coordinates at pinned points. Other slices and parameter choices can cross additional activation regions.

RACHEL: Patrick, what does the slice reveal about a local peak?

PATRICK: For this path the peak occurs at X two equal to one half, where S equals P and the normalized output reaches one. Both ends of the slice have normalized output one half. The arithmetic trace establishes those values independent of the camera angle.

SARAH: A different slice may reach other regions, but this one covers only the first rise and subsequent fall. The pinned numerical readout is authoritative; camera motion and color do not modify the node calculation.

=== 64 — Experiment 10: prepare a reproducible technical observation ===

SARAH: To document an observation, note laboratory version one point three point zero, the chosen preset or custom coefficients, P, arithmetic mode, and the parameter rescaling setting. Note X one and X two for the selected point, then copy the exact weighted sum, region, raw Y, and normalized Y divided by P. Export the textual configuration and truth table from panel six. In WebGL, choose a projection, palette, resolution, and visibility settings, pin an interesting point if appropriate, and save a PNG. When comparing experiments, distinguish directly evaluated numbers from triangles interpolated in a sampled surface.

JUSTIN: Richard, what should be saved if two researchers want to compare their findings?

RICHARD: Document the preset or coefficients, P, arithmetic mode, rescaling policy, selected input coordinates, exact output trace, and any selected WebGL settings.

Save the copied configuration text and the static PNG separately so others can reproduce the observation.

JUSTIN: An observation becomes useful to another engineer when the settings, arithmetic mode, selected inputs, and numerical outputs are recorded.

=== 65 — Accessibility and presentation tips ===

JUSTIN: For accessible use, choose one of the ten interface themes and six font families, including IBM Plex Sans, IBM Plex Sans Condensed, IBM Plex Serif, IBM Plex Mono, a system sans, and Arial or Helvetica fallback. The text size selector offers small, medium, large, and extra large. For a projector demonstration, use a high-contrast theme, readable numeric labels, and enough zoom. Tooltip help can be opened from the question-mark icons by pointer or keyboard focus. The ordinary panels remain usable when WebGL is unavailable. A static screenshot may be visually attractive, but an accessible accompanying textual description still helps technical readers.

SARAH: Rachel, what if the application is being demonstrated on a projector?

RACHEL: Increase the interface text size, choose a high-contrast theme and a legible font, and consider zooming the page before starting. Use the graphical explorer for visual intuition but rely on its numeric labels for precision. Check accessibility before presenting to an audience.

PATRICK: A clear demonstration favors legibility over decoration. Increase font size and contrast before zooming into a complex surface.

=== 66 — Troubleshooting your browser session ===

RICHARD: If the laboratory appears unusually small, increase its font-size option or your browser zoom. If a diagram seems stale, reload the page, return to the XOR preset, and check the truth table before changing multiple settings. Unexpected values are often caused by a non-default P setting, by Fixed INT rounding, or by a parameter adjustment made earlier in the session.

RACHEL: Justin, what should a visitor check first if the numbers do not match this video?

JUSTIN: Confirm that the page says version one point three point zero, reload the exact preset, set P to sixteen, select Exact real arithmetic, and set the same X one and X two inputs as our example. Check whether automatic parameter rescaling is enabled; the same raw weights with a different P can behave differently.

RICHARD: Use the arithmetic trace as the reference point. Write down the five values used in that calculation: W one, W two, bias, and the two input coordinates. If the numerical trace is correct but the image is unclear, the issue is likely presentation rather than the underlying node calculation.

=== 67 — Troubleshooting: the WebGL explorer ===

RACHEL: If the WebGL explorer reports that WebGL is unavailable, update the browser, check graphics drivers or enterprise browser policies, and try another current browser. The original panel-five chart should still work. If the surface looks black or unexpectedly blank, try Reset View and a lower mesh resolution. If labels or tick values are hidden, check the visibility toggles; if colors are confusing, choose Spectrum or Ocean Cyan. For difficult touch interactions, use the

available zoom buttons and try a desktop mouse. This is a graphics troubleshooting issue, not automatically a failure of the underlying IGL calculation.

PATRICK: Sarah, if WebGL fails, does the demonstration become unusable?

SARAH: No. The ordinary six panels and the original canvas-based surface still provide the key numerical education. Verify browser settings and graphics support for the enhanced explorer, but do not mistake a GPU availability issue for a defect in the IGL arithmetic core.

RACHEL: The WebGL failure message does not invalidate the arithmetic. Return to the original panel five surface and use the trace and truth table.

=== 68 — Browser processing, privacy and responsible use ===

PATRICK: This laboratory computes the weighted sum, activation value, truth tables, candidate search, and displayed surfaces in the visitor’s browser. It does not need a remote AI inference service for these calculations. Ordinary web-page hosting, font loading, analytics, and browser policies are separate considerations; we should not infer a blanket privacy guarantee from the fact that the arithmetic is local.

RACHEL: Patrick, does moving the sliders require transmitting the IGL arithmetic to a remote model?

PATRICK: No for the demonstrator’s model calculations. The browser evaluates the current parameters locally. The optional WebGL view similarly renders a locally sampled surface. This is a transparency feature of the research demonstration, not evidence of particular performance, security, or privacy properties in a deployed AI product.

SARAH: For a reproducible session, you can copy the configuration and truth table, note the browser and laboratory version, and compare the exact numerical trace. Avoid including private information in shared screenshots of an unrelated browser window.

=== 69 — What a single-node demonstration can and cannot show ===

SARAH: The six panels share one two-input node and one set of parameters. The truth table proves the four Boolean corner responses for that node, while the fractional-input controls and three-dimensional surface show additional graded responses. The local candidate search demonstrates a limited search step. These observations are powerful because they make the arithmetic visible, but their scope is deliberately narrow.

JUSTIN: Richard, how should a technically sophisticated viewer interpret those results?

RICHARD: Treat the node, the arithmetic trace, and the measured outputs as the evidence in this laboratory. Do not interpret a small on-screen search as proof of full neural-network training, or a colorful surface as a measurement of inference speed or energy consumption. Those require separate experiments and representative workloads.

JUSTIN: This distinction strengthens the demonstration: showing exactly what is calculated makes both successful results and limitations open to inspection, replication, and criticism.

=== 70 — Verification results and known gaps ===

JUSTIN: The shipped test report documents more than twenty-five thousand numerical assertions covering gate tables, all seven P settings, both arithmetic modes, and randomized evaluations. Additional browser checks cover responsive layout and the WebGL controls. Those checks support confidence that the demonstration behaves as intended under tested conditions. They do not constitute a formal proof over every numerical input, nor verification of the complete patented training architecture, and they do not establish latency, power, or accuracy results for embedded hardware.

SARAH: Rachel, do thousands of test assertions prove the entire architecture is correct?

RACHEL: They are useful confidence in this educational implementation, not a universal proof. Tests cover the catalog, scale settings, arithmetic and visualization behavior. The overall architecture and any claims about trained networks require independent experiments beyond the single-node demonstration.

PATRICK: Large test counts increase confidence within their stated scope. They are not substitutes for independently measured hardware or full-network results.

=== 71 — Frequently asked questions ===

RICHARD: A common question is whether all the sixteen gates are trained from scratch. The presets are exact precomputed parameter examples; the local optimizer is a separate demonstration. Another question is whether choosing a higher P automatically improves accuracy. It does not: it changes arithmetic scale and quantization behavior under specified conditions. A third question concerns the 3D image: its surface is a sampled representation of a numerical model. The exact trace and truth table are the authority for a selected input. The application is a visual educational lab, not a full trainer or a hardware benchmark.

RACHEL: Justin, what is the single most important distinction to leave with the viewer?

JUSTIN: This is a transparent, interactive explanation of one IGL node. It demonstrates exact Boolean corners, graded behavior, and a specific quantized arithmetic model. It does not demonstrate complete multi-layer training or measured hardware performance. Knowing the boundary makes the tool more credible.

RICHARD: The difference between a loaded exact preset and a learned parameter set matters. Keep those two experiments separate.

=== 72 — What the full IGL architecture adds ===

RACHEL: The application we have studied is intentionally limited to one two-input node. A practical IGL network contains many connected nodes, with information propagated through its learned structure. The broader research explores discrete parameter optimization through chain isolation, separating portions of a network so candidate parameter changes can be evaluated more effectively. Those procedures are beyond what the little Try One and Try Ten buttons do. The educational connection is conceptual: the laboratory exposes how a discrete parameter change can alter a node output, sometimes substantially, without differentiating through the five-region curve. It is a starting point for discussing learning and information flow, not a substitute for a multi-node benchmark.

PATRICK: Sarah, how should viewers interpret the word training in the title of the broader research?

SARAH: Training refers to searching for parameters that minimize an objective across examples. Here we only illustrate local neighbor evaluation. The production research architectures are more complex and must be judged with proper training, testing, baseline comparisons, and reproducibility.

RACHEL: A strong demonstration leaves room for scrutiny. The fact that every Boolean truth table has an exact preset is a statement about representational capability; discovering useful parameters on a novel task is a separate research question.

=== 73 — A worked model-checking protocol ===

PATRICK: When conducting a reproducible session, start by noting laboratory version, preset name, P, arithmetic mode, W one, W two, bias, and selected input coordinates. Write down the four binary outputs, the selected arithmetic trace, and the activation region. In the WebGL popup note projection, palette, mesh density, enabled axes and labels, camera orientation, and cross-section mode. Save both the textual configuration from the copy button and a static PNG of the chosen view. If discussing performance, make clear that a browser rendering demonstration has not measured microcontroller latency, watt-hours, working memory, or neural network accuracy. For educational replication, the exact Boolean table is the strongest first check.

RACHEL: Patrick, what could a reviewer challenge in a screenshot alone?

PATRICK: A screenshot does not tell us whether it represents real arithmetic or quantized arithmetic, whether coefficients were rescaled, or which points were selected. The accompanying numerical record resolves that ambiguity.

SARAH: If another person can reproduce the same S, region, and Y/P, you have a much stronger observation than a screenshot alone.

=== 74 — Key lessons: from gates to graded responses ===

RICHARD: We began with a weighted sum S, followed by the five-region activation and normalization by P. We learned why P matters: it sets the activation breakpoints, the raw output scale, and the integer input grid in Fixed INT mode. Proportional scaling of weights and bias can preserve normalized behavior in real mode, while changing P alone can transform the represented gate. We then reproduced all sixteen two-input Boolean truth tables with a single node and explored graded responses between the binary corners. Finally, the six linked panels and WebGL explorer let us inspect the arithmetic and geometry in detail.

JUSTIN: Richard, what is the central message for an engineer seeing IGL for the first time?

RICHARD: The representation is more flexible than the single conventional threshold gate that might come to mind. But representational flexibility should invite training experiments, error analysis, and careful measurement rather than an unsupported conclusion about complete neural networks.

JUSTIN: For students, the lab is a way to experiment with a nonlinear transfer function. For researchers, it is an accessible source of hypotheses and

reproducible questions.

=== 75 — Keep exploring · Final invitation ===

JUSTIN: I am Justin, and on behalf of Richard, Patrick, Rachel, and Sarah, thank you for joining this technical tour of the MLiglon IGL Laboratory, version one point three point zero. You now know how a two-input IGL node can match all sixteen Boolean truth tables and still have rich, graded behavior between the corners. You also know the role of P: it defines activation scale, supports consistent integer representation, and changes the arithmetic grid. Powers of two are convenient in binary implementations, while other positive scales can be useful in other systems. Return to the interactive laboratory at M Liglon dot com slash demo, and use the downloadable user guide when you want to reproduce a particular experiment. Above all, change one variable at a time, examine the numerical trace, and let evidence guide your interpretation.

SARAH: Rachel, one last sentence before we conclude?

RACHEL: Curiosity is the beginning. A reproducible experiment is how curiosity becomes useful engineering knowledge.

PATRICK: Thanks for watching. The most valuable next demonstration might be the one you design for yourself.

JUSTIN: Until next time, keep questioning established assumptions, and let the numerical evidence guide the result.