NxtLight Framework

What the 17 Analytical Modules Do — In Plain English

NxtLight Workshop · nxtlight.com · © 2026 William Reed

The NxtLight Framework is a research platform that simulates networks — systems of interconnected nodes, like social networks, power grids, or neural circuits — and watches them evolve over time. The document is organized into two parts. Sections 1 through 4 describe the foundation of the framework: what it is, how it is built, what rules govern its behavior each moment in time, and what broad categories of behavior it can produce. Sections 5 through 21 are the 17 analytical modules — the scientific instruments NxtLight uses to observe and interpret what is happening inside a running simulation. Each module comes from a well-established scientific tradition and contributes a unique perspective. Together, all 21 sections provide a comprehensive picture of the framework from the ground up.

Part I — Foundation (Sections 1–4)

Module 1: What the NxtLight Framework Is

The NxtLight Framework is, at its heart, a sophisticated computer laboratory for studying networks. A network is any system made up of individual parts — called nodes — that are connected to each other by links — called edges. Think of a city's road system: intersections are nodes, and roads are edges. Or think of the human brain: neurons are nodes, and synapses are edges. What makes NxtLight unusual is that it simulates networks where two things happen at the same time: the signals traveling across the network influence how the wiring changes, and the changing wiring in turn shapes where future signals can travel. Most research tools treat structure and behavior as separate problems; NxtLight studies them together, as nature actually presents them (Watts & Strogatz, 1998; Barabási & Albert, 1999).

Module 2: How the Framework Is Built

Under the hood, the NxtLight Framework is a Python computer program. The network itself is managed by a well-tested software library called NetworkX, which keeps track of all the nodes, edges, and their properties. The researcher configures every aspect of a simulation — how many nodes, how densely connected they are, how strongly signals decay, and dozens of other settings — by editing two plain-text configuration files. This design means that any experiment NxtLight has ever run can be perfectly reproduced by anyone who has those same two files, which is a cornerstone of good science. When

a modern graphics card is available, the framework automatically uses it to perform calculations much faster, just as video games use a GPU to render images quickly.

Module 3: The Four Rules That Govern Each Moment

Every tick of the simulation clock, the framework applies four rules in strict order — and that order matters. First, the wiring of the network is updated: new connections may form and old ones may weaken or disappear, based on recent activity. Second, signals propagate across the freshly updated wiring, so the new structure immediately shapes where information can go. Third, the background noise level across each connection is recalculated based on how strongly those connections have been used. Fourth, a signal-to-noise ratio is computed for the whole network, measuring how clearly meaningful signals are standing out above the noise. These four steps repeat thousands of times per simulation, creating the rich, evolving behavior that NxtLight then analyzes with the 17 modules in Part II.

Module 4: The Three Phases a Network Can Be In

One of the most important insights the NxtLight Framework is built around is that networks — like water — can exist in fundamentally different phases depending on conditions. In the Crystallized phase, the network is so densely connected that signals flood everywhere at once; the system is highly active but disorganized, like a room where everyone is shouting at the same time. In the Dissolved phase, the network has too few connections; signals fizzle out and the system goes quiet, like a party where no one is talking to anyone. In between lies the Dissipative phase — the sweet spot where the network is structured enough for meaningful information to flow along specific pathways, but dynamic enough to adapt and reorganize. NxtLight's research focuses heavily on this middle zone, because it is where the most interesting and useful network behavior lives (Prigogine & Stengers, 1984; Langton, 1990).

Part II — Analytical Modules (Sections 5–21)

Module 5: Transfer Entropy

Think of Transfer Entropy as a "rumor tracker." In any network — whether it is a group of friends, a power grid, or a cluster of brain cells — information flows from one node (person, station, or neuron) to another. This module measures how much knowing what Node A did in the past actually helps you predict what Node B will do next. A high score means real influence is flowing; a low score means the two are basically independent. NxtLight uses this as its main early-warning signal: when the average flow of influence across the whole network starts to drop, the system is heading toward collapse.

Module 6: Partial Transfer Entropy / Granger Causality

Module 5 is great at spotting influence, but it can be fooled: if A influences B and B influences C, it looks like A influences C directly — even if A and C have nothing to do with each other. Partial Transfer Entropy fixes that by asking, "After we account for everyone else in the room, does A still tell us anything new about C?" If yes, the influence is real and direct. This is the same idea as Granger Causality, a well-known technique in economics for testing whether one time-series truly drives another (Granger, 1969). The module gives NxtLight a cleaner, cheat-proof map of who is really in charge.

Module 7: Lyapunov Exponent

Imagine two identical snowflakes dropped side-by-side at the top of a mountain. In calm terrain they stay close together; on a chaotic, rocky slope they diverge wildly. The Lyapunov Exponent measures exactly that divergence rate for the network's behavior over time. A positive value means the system is chaotic — tiny differences in starting conditions balloon into completely different outcomes. A negative value means the system is stable and predictable. NxtLight tracks this number to determine whether the network is in a safe, orderly zone or has tipped into unpredictable chaos (Wolf et al., 1985; Rosenstein et al., 1993).

Module 8: SINDy (Sparse Identification of Nonlinear Dynamics)

Most real systems — weather, economics, biology — follow hidden equations that nobody wrote down. SINDy is a machine-learning technique that watches what the network actually does over time and then reverse-engineers the simplest possible equation that explains it (Brunton et al., 2016). "Sparse" means the method prefers short, tidy equations over complicated ones, following the scientific principle of Occam's Razor. NxtLight uses SINDy to turn a simulation's behavior into readable math, which makes it much easier to explain to other scientists what is actually driving the network.

Module 9: Diffusion Maps

Diffusion Maps is a map-making tool for very complicated, high-dimensional data (Coifman & Lafon, 2006). Imagine you have thousands of measurements about the network at every moment in time. Plotting all of them at once would be like trying to read a map that has a million overlapping roads. Diffusion Maps finds the two or three most important "directions" in that data — the hidden axes that explain most of what is going on — and draws a clean, low-dimensional picture. NxtLight uses this to visualize how the network's overall state drifts, clusters, and shifts over a simulation run.

Module 10: TARP (Tests of Accuracy with Random Points)

When NxtLight builds a model and fits it to data, it is important to know whether the model's uncertainty estimates are honest. TARP answers a simple question: "If I pick a random true answer and draw a circle of confidence around my prediction, does the true answer actually fall inside that circle as often as it should?" (Lemos et al., 2023). If the model claims 90% confidence, the true answer should land inside the confidence region 90% of the time — not 50% and not 100%. TARP is essentially a lie-detector test for how well the model knows what it does not know.

Module 11: Persistent Homology

Persistent Homology comes from a branch of mathematics called Topological Data Analysis (TDA). The basic idea is to look for "holes" — loops, voids, and cavities — in the pattern of connections across the network (Edelsbrunner & Harer, 2010). A small cluster of nodes that form a ring, for example, creates a one-dimensional hole. "Persistent" means the module tracks how long those holes last as the network evolves; a hole that appears briefly and disappears is probably noise, while one that persists for a long time is a real structural feature. NxtLight uses this to detect subtle shapes in the network's architecture that ordinary statistics would completely miss.

Module 12: Entropy Production Rate

In physics, entropy is a measure of disorder — the more mixed up and random a system is, the higher its entropy. Entropy Production Rate measures how fast disorder is being created inside the network at any given moment (Prigogine, 1967; Tomé & de Oliveira, 2012). A healthy, organized network creates entropy at a moderate, steady rate — it is doing useful work. A network that is about to fall apart suddenly produces entropy very rapidly, like a spinning top just before it tips over. NxtLight uses this module as a thermodynamic health monitor, watching for those dangerous spikes that signal an impending breakdown.

Module 13: Graph Laplacian / Fiedler Value

The Graph Laplacian is a mathematical matrix that encodes the entire wiring diagram of a network. Its second-smallest eigenvalue — a single number called the Fiedler value after the mathematician Miroslav Fiedler — tells you how well-connected the network is overall (Fiedler, 1973). Think of it as a measure of how many bridges there are between different parts of the network. When the Fiedler value drifts toward zero, the network is about to split into isolated islands. NxtLight monitors it continuously as an early-warning alarm: a falling Fiedler value is the first sign that the network is fracturing.

Module 14: Community Detection

Real-world networks naturally organize into communities — tight-knit groups of nodes that talk to each other a lot but communicate less with the outside world, just like cliques in a school cafeteria. Community Detection algorithms (specifically the Louvain and Leiden methods; Blondel et al., 2008; Traag et al., 2019) automatically find these groups and measure how sharply defined they are using a score called modularity (Q). A high modularity means the network has clear, distinct communities; a low one means everything is blended together. NxtLight uses this to track whether specialized sub-groups are forming, dissolving, or merging as the simulation runs.

Module 15: Network Controllability (Linear)

Network Controllability asks: "How many nodes would I need to directly control in order to steer the entire network to any state I want?" (Liu et al., 2011). Fewer required control points means the network is easier to manage; more means it is stubborn and hard to influence. The linear version of this module uses an elegant result from control theory: the minimum number of "driver nodes" needed can be found by computing a maximum matching on the network — essentially figuring out which nodes have no redundant backup (Hopcroft & Karp, 1973). NxtLight uses this to identify the most strategically critical nodes — the levers of the system.

Module 16: Nonlinear Controllability

Module 15 assumes the network behaves like a simple, linear system — that pushing twice as hard produces twice the result. Real networks rarely work that way. Nonlinear Controllability relaxes that assumption and checks whether the actual, nonlinear dynamics of the network make it harder or easier to control than the linear estimate predicts (Brunton & Kutz, 2019). The module computes a ratio: if that ratio is greater than one, the nonlinearities are making the network harder to steer than it looked. NxtLight uses this as a reality check on the simpler estimate — a safety audit before any control strategy is recommended.

Module 17: Hebbian Learning / Reservoir Computing

This module is the engine underneath everything else — it governs how the network learns. Hebbian learning follows a rule famously summarized as "neurons that fire together, wire together" (Hebb, 1949): if two nodes are active at the same time, the connection between them gets stronger. Over time, the network spontaneously builds information highways along its most-used routes. Reservoir Computing (Jaeger, 2001) adds a complementary idea: the network itself, with all its recurrent connections, acts like a kind of memory — a "reservoir" that stores recent history in its collective state.

Together, these two mechanisms allow the NxtLight network to adapt, learn, and remember without any explicit programming of what to learn.

Module 18: Spectral Radius Tracking

The spectral radius of a network is the largest eigenvalue of its adjacency (connection) matrix — a single number that captures the network's overall connectivity and signal-amplification power (Strogatz, 2001). When this number sits close to 1.0, the network is in what physicists call the "edge of chaos": it is just stable enough to not explode, but dynamic enough to be maximally useful for storing and processing information. If it climbs above 1, signals cascade out of control; if it drops well below 1, the network goes silent. NxtLight tracks this in real time because it is one of the clearest indicators of whether the network is in its most productive operating zone.

Module 19: Information Entropy Suite

Rather than a single measurement, this module is a small suite of four related tools, each asking a different question about disorder and structure. Degree Entropy (H_deg) measures how evenly connections are spread across nodes — a high value means no single node dominates. State Entropy (H_state) measures how varied the activity levels are across the network. Sample Entropy (SampEn) checks whether the network's behavior is predictably repetitive or genuinely irregular (Richman & Moorman, 2000). Excess Entropy (E) measures how much organizational memory the network has built up from its own history (Crutchfield & Feldman, 2003). Together they give NxtLight a multi-angle portrait of the network's informational health.

Module 20: Recurrence Quantification Analysis (RQA)

RQA is built on a beautifully simple idea: if you plot every moment in time against every other moment and ask "was the system doing roughly the same thing at both times?", you get a picture called a Recurrence Plot (Eckmann et al., 1987). Diagonal lines in that picture mean the system repeated similar sequences of behavior. Two key statistics — Determinism (DET, how much of the plot consists of diagonal lines) and Laminarity (LAM, how much consists of vertical lines) — tell you how regular and predictable the network is. NxtLight monitors LAM in particular because sudden spikes in laminarity are reliable early warnings of an impending phase transition — the network is about to change its character.

Module 21: Information Bottleneck

The Information Bottleneck principle asks a fundamental question: when information flows from one part of a network to another, how much of the original signal's meaning is preserved, and how much is lost or compressed away? (Tishby et al., 2000). Think of it like a telephone game: by the time the message reaches the far end of a long chain,

how much of the original meaning survives? The module computes an efficiency matrix across all pairs of communities in the network, where a value near 1.0 means information crosses that bridge almost perfectly and a value near 0 means the communities are informationally isolated. NxtLight uses this to pinpoint exactly where the network's communication is breaking down.

References

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.

Brunton, S. L., & Kutz, J. N. (2019). Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control. Cambridge University Press.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 113(15), 3932–3937.

Coifman, R. R., & Lafon, S. (2006). Diffusion maps. Applied and Computational Harmonic Analysis, 21(1), 5–30.

Crutchfield, J. P., & Feldman, D. P. (2003). Regularities unseen, randomness observed: Levels of entropy convergence. Chaos, 13(1), 25–54.

Eckmann, J.-P., Kamphorst, S. O., & Ruelle, D. (1987). Recurrence plots of dynamical systems. Europhysics Letters, 4(9), 973–977.

Edelsbrunner, H., & Harer, J. (2010). Computational Topology: An Introduction. American Mathematical Society.

Fiedler, M. (1973). Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 23(2), 298–305.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. Econometrica, 37(3), 424–438.

Hebb, D. O. (1949). The Organization of Behavior: A Neuropsychological Theory. Wiley.

Hopcroft, J. E., & Karp, R. M. (1973). An n^{5/2} algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, 2(4), 225–231.

Jaeger, H. (2001). The 'echo state' approach to analysing and training recurrent neural networks (GMD Report 148). German National Research Center for Information Technology.

Lemos, P., Jeffrey, N., Cranmer, K., Ho, S., & Perreault-Levasseur, L. (2023). Sampling-based accuracy testing of posterior estimators for general inference. Astrophysical Journal Letters, 949(1), L14.

Liu, Y.-Y., Slotine, J.-J., & Barabási, A.-L. (2011). Controllability of complex networks. Nature, 473(7346), 167–173.

Prigogine, I. (1967). Introduction to Thermodynamics of Irreversible Processes (3rd ed.). Wiley-Interscience.

Richman, J. S., & Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. American Journal of Physiology — Heart and Circulatory Physiology, 278(6), H2039–H2049.

Rosenstein, M. T., Collins, J. J., & De Luca, C. J. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. Physica D, 65(1–2), 117–134.

Strogatz, S. H. (2001). Exploring complex networks. Nature, 410(6825), 268–276.

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. Proceedings of the 37th Allerton Conference on Communication, Control, and Computation, 368–377.

Tomé, T., & de Oliveira, M. J. (2012). Entropy production in nonequilibrium systems at stationary states. Physical Review Letters, 108(2), 020601.

Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports, 9, 5233.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. Nature, 393(6684), 440–442.

Wolf, A., Swift, J. B., Swinney, H. L., & Vastano, J. A. (1985). Determining Lyapunov exponents from a time series. Physica D, 16(3), 285–317.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439), 509–512.

Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. Physica D, 42(1–3), 12–37.

Prigogine, I., & Stengers, I. (1984). Order Out of Chaos: Man's New Dialogue with Nature. Bantam Books.