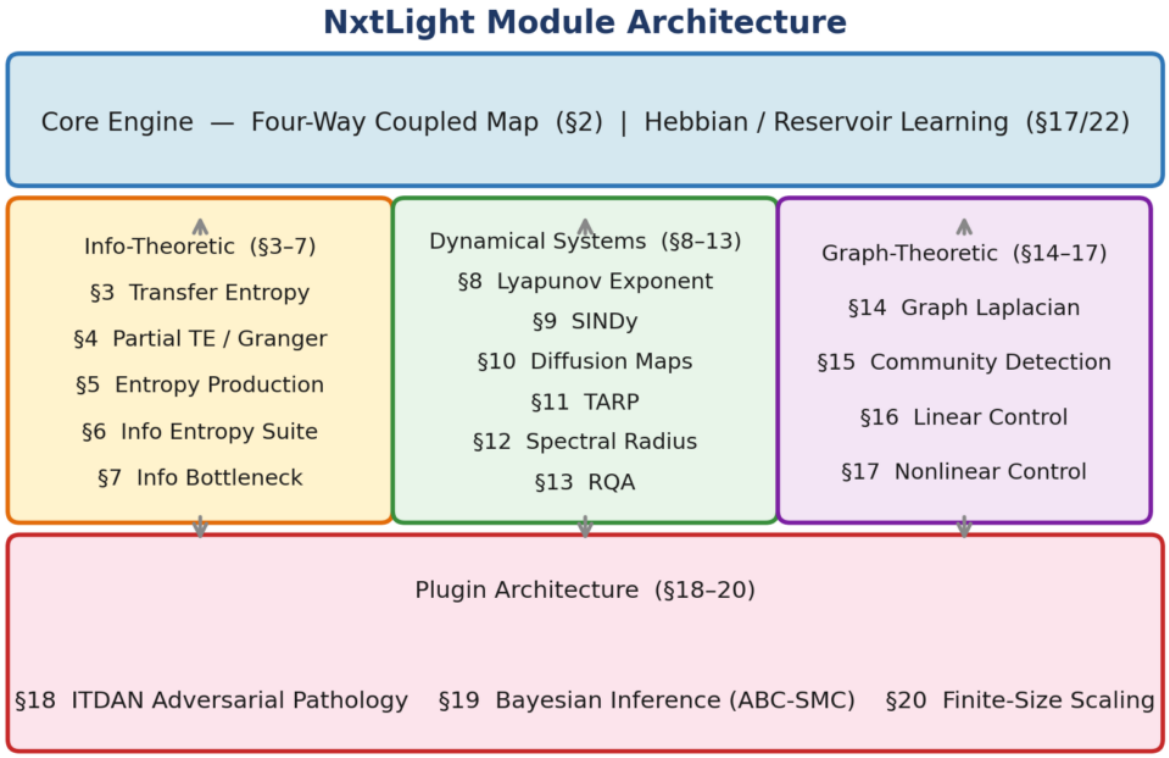
The NxtLight Framework addresses a gap in scientific software: the absence of a unified, domain-agnostic platform for simulating dynamical processes on networks whose topology, node state, and edge connectivity can evolve in real time. Existing tools either target a specific domain (epidemiological compartment models, power grid simulators, financial contagion packages) or require deep software engineering expertise to adapt.

Figure 2.3 NxtLight module map organized by analytic tradition.
All modules receive co-evolved (G, A, w) state from the core engine each timestep.



The NxtLight Framework provides a ***configurable middle layer:*** domain experts define their system in a configuration file; the framework handles graph management, time integration, analytical module dispatch, visualization, and output. use Python for modeling and generating graphs for analyses. This page provides an introduction to that process and some example algorithms. Please consider all information tentative and for instructional purposes only. All final presentations are in PDF form and available by request.

The Computing Package Exists in the Following Submodules

Submodule	**Responsibility**	**Key Dependencies**
engine/	Graph state management, time stepping, rewiring logic, event dispatch	NetworkX, SciPy, CuPy (optional)

config/	TOML parsing, schema validation, default injection, error reporting	tomllib, pydantic
modules/	Built-in analytical plugins: TE, SINDy, diffusion maps, homology, Lyapunov, TARP, hub ID	NumPy, SciPy, pysindy, ripser, nolds
gui/	Tabbed window, graph canvas, chart panels, status bar, color mapping	matplotlib, Plotly/Dash, PyQt6
outputs/	Spreadsheet writer, chart export, run metadata logging	openpyxl, pandas
plugins/	Drop-in directory scanned at startup; also entry-point discovery	importlib.metadata
utils/	GPU detection, logging, progress tracking, unit conversion	CuPy, psutil, loguru

The following modules are built into the framework as reference implementations. Each is also a fully compliant plugin and can be replaced or extended by a community module of the same name. All modules are documented with NumPy-style docstrings and tested in the pytest suite.

Module	**Method**	**Primary Open-Source Dependency**
transfer_entropy	Kraskov k-NN estimator (default), Gaussian estimator, binned estimator. Measures directed information flow between node time series.	IDTxl, pyinform
sindy	Sparse Identification of Nonlinear Dynamics. Recovers governing equations from node state time series using the STLSQ optimizer.	pysindy
diffusion_maps	Spectral embedding of the graph state manifold. Identifies slow collective variables and intrinsic low-dimensional structure.	PyDiffMap
persistent_homology	Topological persistence of graph connectivity at multiple spatial/weight scales. Produces persistence diagrams and Betti number time series.	ripser, persim

lyapunov	Lyapunov exponent estimation from node state divergence trajectories. Characterizes attractor stability and chaos onset.	nolds
tarp	Tests of Accuracy with Random Points. Validates simulation outputs against observational or synthetic reference data.	tarp (custom implementation)
hub_identification	Betweenness centrality, degree entropy, eigenvector centrality, and TE-based hub scoring. Updated at each analysis window.	NetworkX, NumPy