BCQM Stage–3 Working Note: Logging Specification for Causal and Metric Graph Construction

Peter M. Ferguson
Independent Researcher

26 February 2026

Abstract

Stage–2 BCQM simulations established a stable spatial cloth (community super–graph G_1) and demonstrated an empirical decoupling between the spatial crossover contour $N_k(n)$ and the temporal lock contour $N_\ell(n)$. Exploratory Stage–3 work revealed that extracting meaningful causal and metric graph objects (G_t and G_2) from existing Stage–2 logs is problematic: the available data were designed for cloth diagnostics and lack the resolution, directional coverage, and temporal granularity required for genuine causal and metric analysis. This note steps back from the data–driven approach and defines, from first principles, what G_t and G_2 should be as physical objects, what observables are required to construct them, and what the simulation kernel must log to support those constructions. This specification has been reviewed and incorporates engineering refinements from that review. It forms the basis for Stage–3 run design and the implementation brief for the updated logging module.

1 Background and motivation

The Stage–2 programme produced two empirically distinct finite–size crossover contours at fixed coherence n:

- the *spatial knee* $N_k(n)$, defined by the cloth core fraction $\Phi = N_{\text{core}}/N_{\text{total}}$ crossing 0.5; and
- the *temporal lock contour* $N_\ell(n)$, defined by the low–Q_{clock} tail probability falling below 0.05.

These contours do not coincide: temporal coherence survives well into regimes of spatial fragmentation. This decoupling is the central empirical result motivating Stage–3: space and time emerge as parallel, not subordinate, structures.

Attempts to extract causal and metric graph objects from existing Stage–2 logs encountered three recurring obstacles:

1. **Logging granularity.** Existing logs record only end–of–bin events. Intra–bin causal structure is integrated out, making directed flow between events unresolvable below the bin scale.
2. **Sparse coverage.** Ledger–derived adjacency is a forest under natural thresholds: time–tagged edges cover only 0.4–0.5% of the cyclic cloth core graph. Loop–based diagnostics (holonomy) are therefore undefined.
3. **Spatial contamination of temporal objects.** The natural node set for G_t (the community partition from the spatial cloth) imports spatial structure into what should be an independent temporal construction. A genuinely independent G_t requires a clock–native node definition.

This specification resolves all three issues by defining the required observables from the physics outward and deriving the logging requirements from those definitions.

2 Physical definitions

2.1 The causal graph Gt: a clock–native construction

G_t is intended to represent the macroscopic causal structure of the system *without reference to the spatial cloth.* Its definition therefore rests on the clock and thread–transition structure alone.

Nodes. A node of G_t is a *clock state*: a group of threads sharing a common clock–phase bin at a given measurement step. At each measurement epoch τ the N active threads are partitioned into groups $\{T_a(\tau)\}$ according to their current v_{glue} clock phase, binned post–hoc into M_ϕ equal–width phase bins covering $[0, 2\pi)$. A node is the pair (a, τ), representing phase bin a at epoch τ.

This definition is independent of the spatial community partition. The spatial communities enter only if one wishes to study how clock–state nodes *relate* to spatial communities, which is a diagnostic question, not a definitional one.

Phase binning and circular boundaries. Because clock phase is a circular ($U(1)$) quantity, bin boundaries must be handled with care: a thread at phase 0.01 and a thread at $2\pi - 0.01$ are physically nearly synchronised but will fall in opposite linear bins. Bin assignment should use circular statistics; in practice the analysis scripts should allow for dynamically centring bins around the mean order parameter of the core to avoid spurious boundary artefacts. The simulation logs raw continuous phases (Section 3.1); the choice of M_ϕ and any centring convention are therefore deferred entirely to the analysis stage and do not appear in the simulation config.

Directed edges. An edge $(a, \tau) \to (b, \tau + \delta)$ exists if at least one thread transitions from phase bin a at epoch τ to phase bin b at epoch $\tau + \delta$, for a chosen temporal resolution δ. The natural edge weight is the transition count. For a first implementation $\delta = 1$ epoch is sufficient; coarser choices (δ = bin width) reproduce a mesoscopic version comparable to the bin–endpoint G_t explored in Stage–3 VII_d.

DAG property. If the clock advances monotonically (guaranteed by the v_{glue} construction), choosing $\delta > 0$ ensures G_t is a DAG by construction.

Physical interpretation. G_t encodes how clock coherence propagates: a dense, narrow G_t (threads mostly staying in the same phase bin) corresponds to high temporal coherence; a diffuse, broad G_t (threads spreading across phase bins) corresponds to clock metastability. G_t is therefore the natural object for studying the temporal lock transition.

2.2 The metric graph G2: a distance–native construction

G_2 is the metric counterpart to the topological cloth G_1. Where G_1 encodes connectivity, G_2 encodes distance in an operationally meaningful sense.

Nodes. The nodes of G_2 are mesoscopic communities defined by the Louvain partition on the cloth core graph, plus a single macroscopic **Halo node** (assigned ID -1). Any event present in `cloth_trace` but absent from the Louvain partition of the cloth core (i.e. a singleton or disconnected event) is assigned to the Halo node. This prevents the community count K from exploding at large N (where outside–only singleton transitions dominate), keeps the transition matrix compact, and provides a clean single node for Core→Halo and Halo→Core causal–feedback diagnostics.

Note: the Halo node (-1) is deliberately excluded from geometric diagnostics (diffusion distance, Ricci curvature, triangle–inequality tests) since it is a catch–all bookkeeping node, not a spatial community.

Distance definition. The edge weight between communities C_A and C_B in G_2 is a derived *distance*, not a raw transition count. Two natural choices are:

1. **Diffusion distance.** Define the random–walk transition matrix P_{ij} on G_1 (row–normalised flow weights, including self–loops; see Section 3.2). The diffusion distance between C_A and C_B at diffusion time s is

$$d_s(C_A, C_B) = \left(\sum_k \frac{(P^s_{Ak} - P^s_{Bk})^2}{\phi_k} \right)^{1/2},$$

   where ϕ_k is the stationary distribution of the random walk. This distance respects cluster structure and is robust to small perturbations.

2. **Spectral embedding distance.** Embed communities into the eigenspace of the G_1 graph Laplacian using the leading d non–trivial eigenvectors, then measure Euclidean distance in that embedding. The effective dimension d_{eff} has a natural interpretation as the number of significant eigenvectors.

Both should be computed and compared in Stage–3.

Physical interpretation. G_2 is the object on which geometric diagnostics are meaningful: Ricci curvature estimates, triangle–inequality tests, and effective–dimension scaling. It is also the natural spatial background against which G_t causal propagation can be measured (e.g. the operational speed of causality defined as maximum spatial hop per clock step).

3 What the logs must provide

3.1 For Gt: per–epoch clock–phase transitions

To construct G_t with clock–native nodes the simulation must log, at each measurement epoch τ:

L1. Per–thread clock phase $\phi_p(\tau) \in [0, 2\pi)$ for each thread $p = 1, \ldots, N$, stored as `float32`. This is the raw v_{glue} phase, not a binned or scalar summary; `float32` precision is sufficient for mesoscopic clock synchronisation and halves storage compared with `float64`. If phase–difference precision near lock becomes a concern, `float64` may be selected via a config option. The choice of M_ϕ phase bins is deferred entirely to analysis; the simulation kernel does not need to know about phase bins at all.

L2. Per–thread event ID $u_p(\tau)$: the event currently occupied by thread p at epoch τ. (Already present in `cloth_trace.event_at_end` at bin resolution; this extends it to epoch resolution or retains it at the chosen δ.)

L3. Per–thread core mask $m_p(\tau) \in \{0, 1\}$: whether thread p is on the cloth core at epoch τ. (Already present at bin resolution; same extension applies.)

These three fields, stored at resolution δ, are sufficient to construct G_t with any desired phase binning and any node definition that is a function of clock phase.

3.2 For G2: per–bin directed community flows

To construct G_2 with well–defined diffusion distances the simulation must log at each bin boundary k:

L4. Directed community flow counts $F_{ij}(k)$: the number of thread transitions from community C_i at bin $k-1$ to community C_j at bin k, stored in sparse form as arrays of `(src, dst, count)` triples. *Self–loops ($i = j$) must be included.* If a thread remains in the same community between bins, that transition must be tallied; omitting self–loops silently corrupts the row–normalisation of P_{ij} and therefore the stationary distribution ϕ_k used in the diffusion distance.

L5. Directed community flow counts (core–only) $F^c_{ij}(k)$: same as L4, restricted to threads with core mask $= 1$ at both endpoints.

L6. Community occupancy $n_i(k)$: number of threads in community C_i at bin k, required to normalise $F_{ij}(k)$ into a transition matrix $P(k)$.

L7. Community membership map π: the mapping from event IDs to community IDs (including Halo assignment for singletons), stored once per run. This ensures reproducibility if community detection parameters change across versions.

3.3 For the joint Gt/G2 diagnostic

L8. Spatial distance matrix on G_1: pairwise community distances computed from G_1 (either graph–hop distance or diffusion distance at a chosen s), stored once per run as a symmetric matrix or sparse dictionary. The Halo node (-1) is excluded from this matrix.

4 Logging container: stage3 trace

All new fields are stored under a dedicated top–level container `stage3_trace` in `RUN_METRICS`, activated by a feature flag `stage3_trace_enabled: true/false` in the run config. The flag must control only memory allocation and file writing; it must never touch the logic that updates edge weights or thread states, so that the Stage–2 baseline remains uncompromised.

The container schema is:

```
"stage3_trace": {
  "meta": {
    "stage3_trace_version": "1.2",
    "code_commit":  "<git hash>",
    "config_file":  "<filename>",
    "run_id":       "n{n}_N{N}_seed{seed}",
    "delta_epochs": <int>,
    "phase_dtype":  "float32"
  },
  "community_partition": {
    "method":          "louvain",
    "resolution":      <float>,
    "num_communities": <int K>,
    "halo_id":         -1,
    "node_to_comm":    [c_0, c_1, ..., c_{V-1}]
  },
  "clock_phase_by_epoch": [
    {"epoch": tau,
     "phase": [phi_0, phi_1, ..., phi_{N-1}]},
    ...
  ],
  "core_mask_by_epoch": [
    {"epoch": tau,
     "mask": [m_0, m_1, ..., m_{N-1}]},
    ...
  ],
  "community_flow_by_bin_all": [
    {"bin": k, "src": [...],
               "dst": [...], "count": [...]},
    ...
  ],
  "community_flow_by_bin_core": [
    {"bin": k, "src": [...],
               "dst": [...], "count": [...]},
    ...
  ],
  "community_occupancy_by_bin_all": [
    {"bin": k, "comm": [...], "count": [...]},
    ...
  ],
  "community_occupancy_by_bin_core": [
    {"bin": k, "comm": [...], "count": [...]},
    ...
  ],
  "community_turnover_by_bin_all": [
    {"bin": k, "comm": [...],
               "entry": [...], "exit": [...], "stay": [...]},
    ...
  ],
  "community_turnover_by_bin_core": [
    {"bin": k, "comm": [...],
               "entry": [...], "exit": [...], "stay": [...]},
    ...
  ],
  "island_trace": {
    "island_version": "1.0",
    "definition": {
      "occ_threshold_frac": <float>,
      "occ_threshold_count": <int>,
      "radius_hops": <int>,
      "matching": "max_overlap"
    },
    "comm_to_island_by_bin_all": [
      {"bin": k, "island_id_by_comm": [i_0, i_1, ..., i_K, i_halo]},
      ...
    ],
    "island_stats_by_bin_all": [
      {"bin": k, "island": [...],
                 "num_comm": [...], "occ": [...], "occ_core": [...]},
      ...
    ]
  },
  "g1_distance_matrix": {
    "method":    "graph_hop",
    "comm_ids":  [...],
    "distances": [[...], ...]
  }
}
```

5 Islands and turnover diagnostics

Stage–3 aims to distinguish a persistent cloth geometry object from transient "islands" (localised, time–varying concentrations of activity or coherence). The logging fields above are sufficient to construct islands and turnover metrics in post–processing; however, for convenience and to reduce repeated recomputation, the specification includes two derived outputs (required): `community_turnover_by_bin_*` and `island_trace`.

5.1 Community–level turnover (derived, required)

Given the per–bin flow matrices (which must include self–loops), community turnover can be derived per bin for each community ID (including the Halo node):

$$\text{entry}_i(k) = \sum_j F_{j\to i}(k), \tag{1}$$

$$\text{exit}_i(k) = \sum_j F_{i\to j}(k), \tag{2}$$

$$\text{stay}_i(k) = F_{i\to i}(k). \tag{3}$$

These quantities support direct "dynamic turnover" diagnostics: entry/exit asymmetry, persistence within communities, and the relationship between turnover and Q_{clock} filtering (core–only versus all).

5.2 Islands on the cloth (derived, required)

Threshold policy. The island occupancy threshold must be specified explicitly using one of:

- `occ_threshold_frac`: a fixed fraction of threads (e.g. 0.05), or
- `occ_threshold_count`: a fixed count of threads (e.g. 5), with implied `occ_threshold_frac` $=$ `count`$/N_{\text{threads}}$.

The chosen policy is part of the run definition and must be recorded in `island_trace.definition`.

An "island" is defined operationally as a connected cluster of communities on the fixed cloth core graph whose occupancy exceeds a chosen threshold in a given bin. A minimal construction is:

(a) choose an occupancy threshold (e.g. fraction of threads),

(b) select the set of communities above threshold at bin k,

(c) connect two selected communities if their G_1 hop distance is at most a small radius (e.g. 1 or 2),

(d) take connected components as islands.

Island turnover is then quantified by tracking these components across bins via maximum overlap of member communities (stable island IDs), and reporting per–island occupancy, size, and entry/exit rates. This diagnostic explicitly separates:

- *persistent cloth*: the underlying G_1 geometry and its distance matrix; from
- *dynamic islands*: time–varying occupancy clusters living on that cloth.

The optional `island_trace` block records the per–bin community–to–island mapping and island summary statistics. It is intended as a compact, audit–friendly intermediate object for Stage–3 analysis scripts.

6 Run design implications

6.1 Storage estimates

For $N = 32$, $B = 20$ bins, $K \approx 30$ Louvain communities plus the Halo node, storing phases at bin resolution (δ = bin width):

- `clock_phase_by_epoch`: $20 \times 32 \times 4$ bytes (`float32`) ≈ 2.5 kB per run.
- `community_flow_by_bin_all`: at most $(K+1)^2 \times B \approx 31^2 \times 20 = 19{,}220$ triples; in sparse form well under 100 kB per run.
- `g1_distance_matrix`: $K^2 = 900$ floats ≈ 7 kB.

Total `stage3_trace` overhead is well under 200 kB per run, compared with the ~ 23 MB existing `RUN_METRICS` files. Even at $N = 1024$ (future scaling), per–epoch phase logging at `float32` would add only ~ 80 MB per seed: easily manageable.

6.2 Recommended initial run set

The first Stage–3 runs with `stage3_trace` enabled should use the $n = 0.800$ knee bracket established in VII_c and VII_d:

- $n = 0.800$, $N \in \{22, 24, 28, 32\}$ (below–knee, near–knee, post–knee, far–post).
- 10 seeds per (N, n) point.
- Bins = 20; same epoch schedule as VII_b/VII_c.
- `stage3_trace_enabled: true`.

This gives a seed–robust, cross–regime dataset from which G_t, G_2, Core/Halo causal sectors, and the joint metric–causal diagnostic can all be computed. Extension to $n = 0.850$ and $n = 0.900$ follows once the pipeline is validated at $n = 0.800$.

7 What this specification does not do

This specification deliberately avoids:

- Changes to the simulation primitives (G_0): the dynamics are unchanged; only the logging is extended.
- Claims about upstream modifications (non–commutative amplitudes, plaquette sensitivity, dynamical exclusion): these remain speculative forward directions, contingent on what G_t and G_2 actually show.

- Holonomy diagnostics at event level: the Stage–3 holonomy lab note established that these are not supportable from existing logs; this specification addresses that by designing the right logs from the outset, not by patching analysis scripts.

8 Summary of required logging fields

Field	Purpose	Resolution	New?
`clock_phase_by_epoch`	G_t node definition (`float32`)	Per bin (min)	Yes
`core_mask_by_epoch`	G_t core filtering	Per bin	Yes
`community_flow_by_bin_all`	G_2 transition matrix (incl. self–loops)	Per bin	Yes
`community_flow_by_bin_core`	G_2 core–only matrix	Per bin	Yes
`community_occupancy_by_bin_all`	G_2 normalisation	Per bin	Yes
`community_occupancy_by_bin_core`	G_2 normalisation (core)	Per bin	Yes
`community_turnover_by_bin_all`	Turnover (entry/exit/stay) per community	Per bin	Yes
`community_turnover_by_bin_core`	Turnover (core endpoints)	Per bin	Yes
`island_trace`	Islands on the cloth (IDs and per–bin summaries; includes threshold policy)	Per bin	Yes
`community_partition`	Reproducibility; Halo node (ID -1) defined here	Once per run	Yes
`g1_distance_matrix`	Joint diagnostic (Halo excluded)	Once per run	Yes

Table 1: Summary of `stage3_trace` logging fields and their roles. Self–loops must be included in all flow fields. Turnover and island summaries are required for Stage–3 analyses. The Halo node (ID -1) is excluded from metric diagnostics.