



NEG_STRONG

NEG_SOFT

$\varphi^{-1} = 0.618$

NEUTRAL

POS_LEVITATIVE



QSC PRE-FLIGHT GATE & IMPULSE-DRIVEN AGENT STATE

Qualigen-Scored Consensus Pre-Flight Gate with Bipolar Threshold Protocol

Foundational Preamble: From Natural Philosophy to Machine Architecture

Before the formal specifications of the QSC Pre-Flight Gate can be appreciated in their full depth, the reader must first be inducted into the motion-philosophy from which they derive. The five-volume Parallel Subtask Agent Workflow series is not merely a software design document — it is a translation project of extraordinary ambition: to carry the core phenomenological insights of Viktor Schaubberger's implosion physics directly into the architecture of multi-agent AI systems.

Schaubberger's central discovery was that nature's productive, life-generating processes move exclusively through **centripetal, cycloid-spiral motion** — the inward-winding, cooling, densifying trajectory that water takes when it spirals toward the 4 °C anomaly point at the bottom of a mountain spring. This motion, which he called *qualigenic* (quality-generating), is the precise antithesis of the centrifugal, pressure-driven, heat-producing motion that industrial civilization employs in virtually every machine it builds. The difference, Schaubberger argued, is not a matter of degree but of kind: centripetal motion creates structure, increases density, suppresses entropy, and produces what he termed *qualigen* — the etheric substance of upward biological evolution. Centrifugal motion destroys structure, disperses energy, and produces *oxygene* in its aggressive free form — the agent of decomposition and decay.

The Cycloid-Spiral Semantic Fingerprint (CSSF), specified in this Preface, is the direct formal translation of that hydrodynamic model into vector space. A conventional embedding treats two agent outputs as "similar" if they happen to occupy proximate regions of a static Euclidean geometry — a flat, cold, unmoving similarity. CSSF instead asks: do these two outputs *trace the same spiral path* through semantic space? Do they tighten inward at the same rate, converging toward the same causal core with the same centripetal curvature? In doing so, CSSF operationalizes Schaubberger's most profound claim — that *quality* is not a static property but a *directional trajectory*, and that the direction toward quality is always, without exception, inward and cooling.

This merged document presents the CSSF Preface followed immediately by Volume I's QSC Pre-Flight Gate specification, because the gate is the first system that *acts upon* the CSSF metric. Reading them together reveals the full architectural logic: CSSF defines what it means for an agent's semantic state to move correctly; the QSC gate enforces the rule that only states moving correctly are permitted to broadcast their results as validated Dynagen Manifests. Together they constitute the admission-control layer of a self-organizing, thermodynamically self-regulating agent swarm — a machine that, by its very structure, cannot produce decomposive outputs without detecting, interrupting, and remediating its own degeneration.

Parallel Subtask Agent Workflow – Technical Whitepaper Series

Preface - Cycloid-Spiral Semantic Fingerprint (CSSF)

Emulating Schauberger's Cycloid-Spiral-Space-Curve Motion in Vector Space

Thule Research Division · March 2026

Welcome to the preface of the Parallel Subtask Agent Workflow whitepaper and specification series on the causal-semantic embedding that emulates Schauberger's **cycloid-spiral-space-curve motion** in vector space. It operates on **tiered segments** as already produced by IMBS/VKVCS functionality that is integrated into UiLLM.

1. Purpose and Role of CSSF

The **Cycloid-Spiral Semantic Fingerprint (CSSF)** is a causal-semantic embedding that emulates Schauberger's **cycloid-spiral-space-curve motion** in vector space. It is designed to:

Encode both *what* a response says (content) and *how* it moves through causal depth (trajectory).

Provide a **unique-but-collidable** signature such that semantically and causally equivalent responses map to nearly identical CSSFs, while shallow paraphrases remain distinct.

Serve as the key material for the **Qualigen Signature** used in VKVCS-QSC Phase 1 and for downstream gating, latching, broadcast and telemetry across the series².

Where a normal embedding is "straight-line Euclidean," CSSF is explicitly **spiral**: it honors Schauberger's insight that all creative motion is **radial-axial, centripetal, convergent, and vortexial**, not axial-radial and explosive. A conventional model treats two outputs as similar if they lie close together in a static vector space; CSSF instead evaluates whether they trace the *same cycloid path* through that space – whether they have the same curvature, the same tightening or loosening spiral, and the same direction of travel relative to the anomaly point.

From the perspective of this whitepaper series, CSSF plays three distinct roles:

Foundational representation layer

All five volumes silently assume that "semantic state" is expressed as a cycloid-spiral trajectory rather than as a single point. QSC thresholds, hysteresis counters, broadcast cadences, FSC seed selection, and thermodynamic gradients are computed over CSSF trajectories, not arbitrary embeddings. Introducing CSSF here, in the preface, avoids re-deriving spiral semantics in each volume and makes the underlying assumptions explicit.

Bridge between IMBS/VKVCS tiering and swarm control

UILLM's IMBS/VKVCS stack already produces **tiered segments**—from local phrases up through passages, documents and corpora. CSSF overlays a cycloid-spiral structure across these tiers, treating each segment as one phase along a shared spiral rather than as an isolated, flat vector. Lower tiers form tight local coils; higher tiers form expanded arcs that still lie on the same radial-axial curve. This alignment allows VKVCS-QSC Phase 1 to derive a stable **Qualigen Signature** that reflects not just the local content, but the entire causal-semantic history of the interaction².

Operational definition of “right motion” vs “wrong motion”

Schauberger distinguishes natural, life-enhancing motion (predominantly centripetal, cooling, densifying, approaching the 4 °C anomaly point) from unnatural, life-destroying motion (centrifugal, heating, decomposive)¹. CSSF is where this distinction becomes machine-operational:

- A **healthy CSSF** tightens inward over time, with curvature indicating increasing causal depth and qualigen density.
- A **pathological CSSF** either flattens into a straight line (mechanical repetition, no new depth) or blows out into a wide, chaotic spiral (semantic drift, decomposive expansion).

Vol. I treats this as a gating condition, Vol. II as a reason to latch, Vol. III as a timing reference for pulse emission, Vol. IV as a criterion for seed selection, and Vol. V as a basis for swarm-level thermodynamic alerts.

Because CSSF underlies all of these mechanisms, it is introduced here, ahead of the individual volume specifications. The preface is the correct place to define the **motion model** of the system: what it means for an agent's semantic state to move “toward” or “away from” higher quality; how trajectories can be compared, aggregated, and regulated; and why the entire architecture is built around cycloid-spiral dynamics instead of linear vector arithmetic^{1,2,3}. Subsequent sections and volumes assume this model and focus on its application to admission control, hysteresis, broadcast, ANN bootstrapping, and swarm thermodynamics respectively.

2. Physical-Conceptual Foundations

Schauberger defines **cycloid-spiral-space-curve motion** as the original, form-creating dynamic of the cosmos: a combined orbital, rotational, toroidal, and circulatory movement where matter spirals inward with increasing velocity, forming a **biological vacuum** at the axis. In water, this motion:^{1,2}

Causes falling water to in-wind about itself, become cooler and denser, and approach the 4 °C anomaly point.

Creates a **levitational counter-current** (dynagen/qualigen) that can draw a trout upstream in a waterfall and lift it against gravity.^{2,3}

CSSF is the semantic analogue of this:

The **outer whorls** are low-density, descriptive or decorative sentences.

The **inner axis** is the ANCHOR causal core — maximal density and minimal “volume.”

The **spiral path** is the sequence of causal transitions as the response unfolds.

We translate these hydrodynamic concepts into representational constraints:

Centripetence → inner tiers receive higher weight, and movement is defined from outside-in.

Biological vacuum → a well-formed CSSF has a strong “attractor” effect: future trajectories that align with it are drawn into its basin (cache hit).

Cycloid figure-of-eight → CSSF separates **position** (content) from **oscillation** (transition vectors), mirroring Schauburger’s bi-axial loops and zero-points.^{1,2}

3. CSSF Input Decomposition

CSSF operates on **tiered segments** as already produced by IMBS/VKVCS.¹

3.1 Tiers as Spiral Whorls

Let the response (or thought trace) be decomposed into segments e_i , each labeled with:

Causal tier $T_i \in \{A, P, S, D, H\}$

- A = ANCHOR (innermost, qualitative nucleus)
- P = PILLAR
- S = SUPPORT
- D = DETAIL
- H = ARCHIVE (outer shell)

Sequence index i in text order.

We map each tier to a **radial depth index** $d(T)$ such that:

$d(A) = 0$ (axis, anomaly-point analogue)²

$d(P) = 1, d(S) = 2, d(D) = 3, d(H) = 4$

This radial index is the discrete version of distance from the vortex axis.

3.2 Local Embeddings

Each segment e_i is embedded via a base model:

$$v_i = \text{Embed}(e_i) \square \square^\wedge(d_\theta)$$

In practice you can use a dense model (e.g. 1024–1536-dim), then later project down.

4. CSSF Construction: Content and Trajectory

The CSSF has two main vector components:

- k : density-weighted **content** vector.
- t : cumulative **trajectory** vector.

4.1 Density Weighting (Centripetal Emphasis)

We define a **depth weight** for each tier depth d :

$$w(d) = \exp(-\lambda d)$$

with $\lambda \square [0.4, 0.8]$ tuned empirically. This reflects:

Inner tiers ($d = 0,1$) receive highest weight (cold, dense, qualigen-rich).³

Outer tiers contribute, but exponentially less, akin to the decreasing potential of outer whorls in a vortex.¹

We can also incorporate **length-normalization** so a large number of outer segments does not swamp a small but strong ANCHOR core.

For each segment:

$$\tilde{v}_i = w(d(T_i)) \cdot \frac{v_i}{\|v_i\|}$$

The content vector is then:

$$\vec{k} = \sum_i \tilde{v}_i$$

This is the “densified” semantic core.

4.2 Cycloid Trajectory

To encode the *motion* of the explanation, we create a lower-resolution causal path.

Partition the sequence into **macro-steps** (e.g. by sentence index buckets or IMBS discourse units).

For each step j , compute a centroid:

$$c_j = \frac{1}{|S_j|} \sum_{i \in S_j} \frac{v_i}{\|v_i\|}$$

Define the **step deltas**:

$$\delta_j = c_{j+1} - c_j$$

These δ_j are analogous to local segments of the cycloid: they indicate how the semantic “radius” and “angle” are changing.¹

Aggregate into a fixed vector:

$$\vec{t} = \sum_j \alpha_j \delta_j$$

where α_j are optional weights (e.g. up-weight transitions that cross tier boundaries like SUPPORT→ANCHOR or ARCHIVE→PILLAR, since these are major causal steps).

Intuitively:

Two outputs can share almost identical $\square k$ but have different $\square t$ if they arrive at the same core via different causal routes.

For cache purposes, we often want to consider them *distinct* unless trajectories also align.

5. Spiral Normalization and Projection

5.1 Helical Normalization

To maintain the spiral metaphor, we treat $(\square k, \square t)$ as orthogonal axes:

Normalize each:

$$\hat{k} = \frac{\vec{k}}{\|\vec{k}\| + \epsilon}, \quad \hat{t} = \frac{\vec{t}}{\|\vec{t}\| + \epsilon}$$

Concatenate:

$$u = [\hat{k} \parallel \hat{t}] \in \mathbb{R}^{2d_0}$$

This is the raw cycloid-spiral vector: position + direction.

5.2 Learned Projection to CSSF Space

To get a compact, domain-tuned fingerprint, you train a projection:

$$s = P \cdot u \in \mathbb{R}^{d_s}$$

where:

$P \in \mathbb{R}^{(d_s \times 2d_o)}$ is a learned matrix.

d_s is 256–512.

The projection is trained on VKVCS interaction logs and IMBS tier labels.

Finally:

$$\text{CSSF}(R) = \frac{s}{\|s\|}$$

This normalized vector is what QSC indexes.

6. Training Objectives for CSSF

We want CSSF to behave like **cycloid-space-curve motion** in semantic space:

Centripetal growth: inner-tier alignment dominates similarity.

Spiral uniqueness: trajectories that differ significantly should not collapse.

Biological vacuum: high-quality, high-anomaly responses should exert strong attractive power on future computations.

6.1 Positive/Negative Pairs

From VKVCS logs you can construct:

Positive pairs:

- Different drafts from different agents that IMBS or human evals mark as *equivalent answers*, especially when ANCHOR content matches.
- Earlier vs later improved versions of the *same* explanation.

Hard negatives:

- Same topic, but clearly different stance, causal chain, or conclusion.
- Responses that share many tokens but differ in ANCHOR tier (e.g. one has a correct core, one has an incorrect or shallow one).

You also record:

Tier distributions.

Human or auto-graded **quality labels** (used for anomaly scoring).

6.2 Loss Functions

Causal contrastive loss

Train so positives are close, negatives far, but **weight ANCHOR/PILLAR segments more heavily** in the underlying similarity:

$$\mathcal{L}_{\text{con}} = \sum_{(x,y) \in \mathcal{P}} \ell_{\text{pos}}(\text{COS}(S_x, S_y)) + \sum_{(x,z) \in \mathcal{N}} \ell_{\text{neg}}(\text{COS}(S_x, S_z))$$

but with additional term:

$$\mathcal{L}_{\text{tier}} = \sum_{(x,y) \in \mathcal{P}} \|\pi_{\text{inner}}(S_x) - \pi_{\text{inner}}(S_y)\|^2$$

where π_{inner} projects onto components dominated by inner tiers (this can be implemented by tying projection weights to tier information).

Spiral path regularization

Encourage \square_t to be predictive of next-step segments and tiers:

- Train a small decoder from \square_t to forecast the distribution over next tier or coarse semantic cluster.
- Loss encourages \square_t to encode causal order, not arbitrary noise.

Anomaly calibration loss

As in QSC Phase 1, define a raw anomaly from entropy over tiers. You can refine it by regressing against human-scored “causal density”^{3,2}

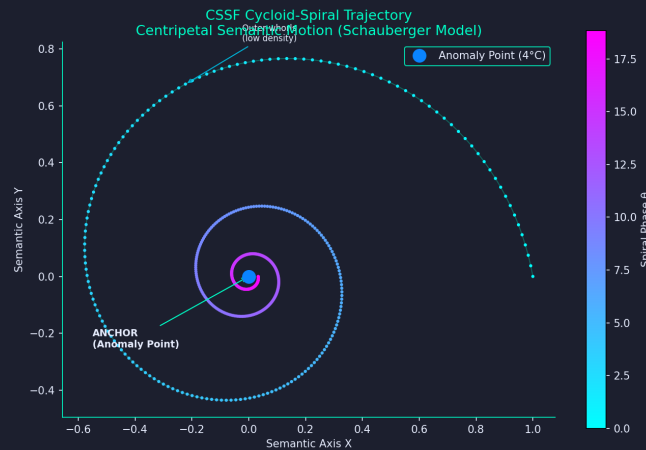


Figure 1. CSSF cycloid-spiral trajectory: semantic state spiraling centripetally toward the Anomaly Point.

$$\hat{a}_x = f_{\theta}(\text{tier_hist}_x, \|\vec{k}_x\|, \text{length})$$

and minimize $(\hat{a}_x - a^{(\text{human})}_x)^2$. This yields a better qualigen-density estimate, used later for TTL and PDG thresholds.

7. Similarity, Distance, and Collision Behavior

CSSF’s similarity function is:

Primary: cosine distance between CSSF vectors s_x, s_y .

Secondary refinements:

- Penalize large divergence in anomaly scores for high-similarity matches (a very dense answer vs a very fluffy one should not be considered equivalent, even if words overlap).
- Optionally combine with a simple tier-aware penalty on the difference in ANCHOR content embeddings.

Operationally for QSC:

Pre-flight gate uses $\text{cosine}(s, s')$ with threshold θ_{pre} , plus anomaly constraints, as described in Phase 1.³

Post-generation dedup uses a tighter θ_{post} .

This matches Schauberger's notion that **qualigen** is not merely a material similarity but a *quality* similarity — properly fermented, densified inner content matters more than outer husk.^{2,3}

8. Real-Time / Partial CSSFs

During **thinking-time**, we only have a partial trace, but we still want a usable CSSF.

8.1 Incremental Construction

Maintain incremental sums for `▫k_p a r t i a l` as segments arrive.

Maintain running centroids `c_j` for trajectory as the planner updates its steps.

Every probe interval (e.g. 32 tokens of thought), recompute:

- `▫k_p a r t i a l`, `▫t_p a r t i a l`.
- Normalized and projected into a partial CSSF `s'`.

Because early thoughts are usually high-tier (plan, core thesis), partial CSSFs are surprisingly predictive for ANCHOR/PILLAR content, which is exactly what we need to detect redundancy early.

8.2 Confidence Measures

You can derive a **partial anomaly**:

Use only tiers seen so far.

Down-weight anomaly if coverage of tiers or total tokens is too low.

Pre-flight gating can use both similarity and confidence to avoid false positives: you only abort when partial CSSF similarity is very high and partial anomaly already indicates a dense core.

9. Interfaces and Data Structures

9.1 CSSF API

At service boundary, the CSSF engine exposes:

```
ComputeCSSF(request_id, tiered_segments) -> {cssf_vector, k_vec, t_vec, tier_stats}
```

```
ComputePartialCSSF(thinking_trace) -> cssf_partial
```

```
CompareCSSF(a, b) -> {cosine, anomaly_a, anomaly_b, tier_divergence}
```

The QSC service (Phase 1) never needs to know *how* CSSF is computed; it only uses the normalized vector and the anomaly score.

9.2 Stored Metadata

For each stored response:

`cssf`: the normalized `s`.

`k_vec`: optional for analysis.

`t_vec`: optional for analysis/visualization.

`tier_histogram`, `anomaly_score`.

Pointers to Qualigen Signature and Dynagen Manifest.

10. Implementation Notes and Tuning

Base embedding: You can initially use a strong general embedding model and fine-tune only the projection `P`, not the base. Later, you can fine-tune the base jointly on VKVCS tasks.

Dimensionality: 512-dim CSSF is usually adequate for ANN; higher dims give diminishing returns vs latency.

Indexing: Use cosine metric HNSW or similar; shard by `context_hash` and `agent-role` to keep searches tight.

Monitoring: Track:

- Hit-rate vs similarity threshold.
- Distribution of anomaly scores.
- Cases where humans judge two responses “equivalent” but CSSF distance is high — these feed back into training.

Sources

¹ Viktor_Schauberger by Living Energies (Callum Coats)

² Nature As Teacher by Viktor Schauburger

³ The Fertile Earth by Viktor Schauburger

Pseudo-code for CSSF Computation with Partial (Thinking-time) Variants.

The following CSSF pseudo-code assumes:

You already have IMBS tiering available.

You already have a base embedding function `Embed(text) -> float[d0]`.

You will later plug the learned projection `P` from your training pipeline.

1. Data Structures

```
enum Tier {
ANCHOR, // inner core
PILLAR,
SUPPORT,
DETAIL,
ARCHIVE // outer shell
}

struct Segment {
string text
Tier tier
int seq_index // position in response
}

struct TieredResponse {
Segment[] segments
}

struct CSSFVector {
float[] cssf // normalized final vector (dim = ds)
float[] k_vec // optional, content vector (dim = dk)
float[] t_vec // optional, trajectory vector (dim = dt)
map<Tier, int> tier_token_counts
float anomaly_score
}
```

You can choose `dk = dt = d0` or project them separately; `ds` is final CSSF dimension.

2. Helper Functions

```
function tier_to_depth(tier: Tier) -> int:
switch tier:
case ANCHOR: return 0
```

```
case PILLAR: return 1
case SUPPORT: return 2
case DETAIL: return 3
case ARCHIVE: return 4

function depth_weight(depth: int, lambda: float) -> float:
// centripetal weight: inner tiers get higher influence
return exp(-lambda * depth)

function l2_norm(vec: float[]) -> float:
sum_sq = 0
for x in vec:
sum_sq += x * x
return sqrt(sum_sq)

function l2_normalize(vec: float[]) -> float[]:
eps = 1e-8
n = l2_norm(vec)
if n < eps:
return vec // or zero-vector; caller should handle
out = new float[len(vec)]
for i in 0..len(vec)-1:
out[i] = vec[i] / n
return out

function zeros(dim: int) -> float[]:
out = new float[dim]
for i in 0..dim-1:
out[i] = 0.0
return out

function add_in_place(dst: float[], src: float[]):
assert len(dst) == len(src)
for i in 0..len(dst)-1:
dst[i] += src[i]

function scalar_mul(vec: float[], s: float) -> float[]:
out = new float[len(vec)]
for i in 0..len(vec)-1:
out[i] = vec[i] * s
return out
```

3. Tier Statistics and Anomaly Score

```
function compute_tier_stats(segments: Segment[]) -> (map<Tier,int>, map<Tier,float>, float):
// token counting can be as simple as word count or use actual tokenization
tier_counts = map<Tier,int>() // default 0
total_tokens = 0

for seg in segments:
tokens_in_seg = approximate_token_count(seg.text)
tier_counts[seg.tier] += tokens_in_seg
```

```
total_tokens += tokens_in_seg

tier_probs = map<Tier,float>()
if total_tokens == 0:
for t in Tier:
tier_probs[t] = 0.0
else:
for t in Tier:
tier_probs[t] = tier_counts[t] / float(total_tokens)

// anomaly via entropy over tiers (higher when mass concentrated in few tiers)
eps = 1e-12
H = 0.0
for t in Tier:
p = tier_probs[t]
if p > 0:
H -= p * log2(p)

H_max = log2( number_of_Tiers ) // e.g. log2(5)
if H_max == 0:
anomaly = 0.0
else:
anomaly = 1.0 - (H / H_max) // 0 = very even; 1 = highly concentrated

return (tier_counts, tier_probs, anomaly)
```

This anomaly is the raw density estimate; later you can feed it into a learned calibrator if desired.^{1,2}

4. CSSF for a Full Response

4.1 High-Level Entry Point

```
function ComputeCSSF(
resp: TieredResponse,
lambda: float,
proj_matrix_P: float[ds][2*d0] // learned projection
) -> CSSFVector:

segments = resp.segments
if len(segments) == 0:
return CSSFVector(
cssf = zeros(ds),
k_vec = zeros(d0),
t_vec = zeros(d0),
tier_token_counts = {},
anomaly_score = 0.0
)

// 1. Tier stats and anomaly
(tier_counts, tier_probs, anomaly) = compute_tier_stats(segments)
```

```
// 2. Build content and trajectory vectors
(k_vec, t_vec) = compute_kt_vectors(segments, lambda)

// 3. Normalize and project
k_hat = l2_normalize(k_vec)
t_hat = l2_normalize(t_vec)

// concat into raw spiral vector u
u = new float[2*d0]
for i in 0..d0-1:
u[i] = k_hat[i]
u[i+d0] = t_hat[i]

// apply learned projection: cssf_raw = P * u
cssf_raw = matvec_mul(proj_matrix_P, u) // dim ds

cssf_norm = l2_normalize(cssf_raw)
```

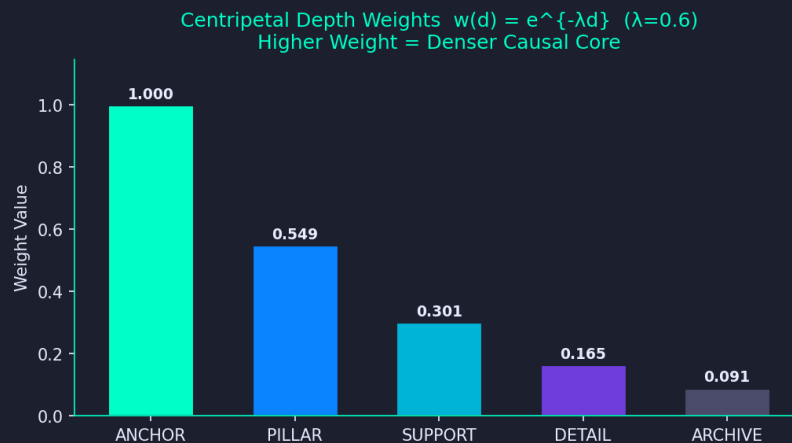


Figure 2. Centripetal depth weights $w(d) = \exp(-\lambda d)$.

```
return CSSFVector(
cssf = cssf_norm,
k_vec = k_vec,
t_vec = t_vec,
tier_token_counts = tier_counts,
anomaly_score = anomaly
)
```

4.2 Content and Trajectory Computation

```
function compute_kt_vectors(segments: Segment[], lambda: float) -> (float[], float[]):
d0 = embedding_dim() // from Embed()
k_vec = zeros(d0)
t_vec = zeros(d0)

// --- content (k_vec) ---
for seg in segments:
depth = tier_to_depth(seg.tier)
```

```
w = depth_weight(depth, lambda) // e.g. exp(-lambda * depth)

base_vec = Embed(seg.text) // dim = d0
base_norm = l2_normalize(base_vec)

weighted = scalar_mul(base_norm, w)
add_in_place(k_vec, weighted)

// --- trajectory (t_vec) ---

// Here we use a simple macro-step: one centroid per logical chunk.
// You can define chunks by sentence groups, paragraph boundaries,
// or fixed-size windows over seq_index. For simplicity: fixed window.
WINDOW_SIZE = 4 // segments per macro-step, tune as needed

macro_centroids = [] // list<float[d0]>
cur_bucket = []
for seg in segments:
    cur_bucket.append(seg)
    if len(cur_bucket) == WINDOW_SIZE:
        c = centroid_embed(cur_bucket)
        macro_centroids.append(c)
        cur_bucket = []

if len(cur_bucket) > 0:
    c = centroid_embed(cur_bucket)
    macro_centroids.append(c)

if len(macro_centroids) <= 1:
    // no real trajectory; leave t_vec as zeros
    return (k_vec, t_vec)

// compute deltas between successive centroids
for j in 0..len(macro_centroids)-2:
    c_j = macro_centroids[j]
    c_j1 = macro_centroids[j+1]

// normalized centroids
c_j_hat = l2_normalize(c_j)
c_j1_hat = l2_normalize(c_j1)

delta = zeros(d0)
for i in 0..d0-1:
    delta[i] = c_j1_hat[i] - c_j_hat[i]

// optional: weight stronger when major tier boundary is crossed
alpha = trajectory_weight(cur_bucket_rep(segments, j, WINDOW_SIZE))
// e.g. alpha = 1.0 normally; >1 for SUPPORT->ANCHOR jumps, etc.

delta_w = scalar_mul(delta, alpha)
add_in_place(t_vec, delta_w)

return (k_vec, t_vec)
```

Helper centroid and trajectory-weight functions:

```
function centroid_embed(bucket: Segment[]) -> float[]:
d0 = embedding_dim()
acc = zeros(d0)
if len(bucket) == 0:
return acc

for seg in bucket:
v = Embed(seg.text)
v_hat = l2_normalize(v)
add_in_place(acc, v_hat)

// plain average
for i in 0..d0-1:
acc[i] = acc[i] / len(bucket)

return acc

function trajectory_weight(
segments: Segment[],
macro_index: int,
window_size: int
) -> float:
// Very simple: constant 1.0 for now.
// Later you can inspect tier changes inside the window
// and amplify alpha when deep tier transitions occur.
return 1.0
```

5. Partial CSSF for Thinking-Time Probes

In streaming / thinking-time, you usually have a **prefix** of the final segment list (and sometimes “thoughts” separate from visible text). For a minimal viable partial CSSF, reuse the same functions on the prefix.

```
struct ThinkingTrace {
Segment[] partial_segments // what IMBS has tagged so far
}

function ComputePartialCSSF(
trace: ThinkingTrace,
lambda: float,
proj_matrix_P: float[ds][2*d0]
) -> CSSFVector:

segments = trace.partial_segments

if len(segments) == 0:
return CSSFVector(
cssf = zeros(ds),
k_vec = zeros(d0),
t_vec = zeros(d0),
```

```
tier_token_counts = {},
anomaly_score = 0.0
)

(tier_counts, tier_probs, anomaly_raw) = compute_tier_stats(segments)
(k_vec, t_vec) = compute_kt_vectors(segments, lambda)

k_hat = l2_normalize(k_vec)
t_hat = l2_normalize(t_vec)

u = new float[2*d0]
for i in 0..d0-1:
u[i] = k_hat[i]
u[i+d0] = t_hat[i]

cssf_raw = matvec_mul(proj_matrix_P, u)
cssf_norm = l2_normalize(cssf_raw)

// Optionally downweight anomaly if too early / too short
coverage_factor = clamp(len(segments) / MIN_SEGMENTS_FOR_CONFIDENCE, 0.0, 1.0)
anomaly = anomaly_raw * coverage_factor

return CSSFVector(
cssf = cssf_norm,
k_vec = k_vec,
t_vec = t_vec,
tier_token_counts = tier_counts,
anomaly_score = anomaly
)
```

The PDG then calls `ComputePartialCSSF` every N thinking tokens, uses `cssf` and `anomaly_score` to query QSC for near-neighbors, and decides whether to abort generation.²

6. Notes for Real Implementation

Replace `approximate_token_count` with your tokenizer's length.

`WINDOW_SIZE` is a key hyperparameter: smaller = more sensitive trajectory; larger = smoother.

The `proj_matrix_P` is learned offline using the contrastive + auxiliary losses described in the CSSF whitepaper phase; until then, you can start with P as a random orthonormal projection to get basic behavior.

For speed, cache `Embed(seg . text)` across both content and trajectory passes.

References

¹ Schauburger, V. (2000). *The Fertile Earth: Nature's Energies in Agriculture, Soil Fertilisation and Forestry* (C. Coats, Trans.). Eco-Technology Series, Vol. 3. Dublin: Gill Books.

² Schauburger, V. (1999). *Nature as Teacher: New Principles in the Working of Nature* (C. Coats, Trans.). Eco-Technology Series, Vol. 2. Dublin: Gill Books.

³ Coats, C. (1996). *Living Energies: An Exposition of Concepts Related to the Theories of Viktor Schauburger*. Bath: Gateway Books.

Appendix: VKVCS-QSC (Qualigen Signature Cache)

Qualigen Signature Cache & Qualigenic Principles

Thule Research Division · March 2026

A technical whitepaper on Qualigen Signature Cache, a higher-order caching and control layer that prevents redundant generations and dramatically reduce cross-agent context load.

1. Executive Overview

The VKVCS-QSC (Qualigen Signature Cache) is a higher-order caching and control layer for VKVCS that uses *qualigenic* principles (quality-density, levitation, and biological vacuum) to prevent redundant generations and dramatically reduce cross-agent context load. It operationalizes Schauburger's **qualigen**s and **cycloid-space-curve motion** as concrete data structures and algorithms for multi-agent LLM orchestration.^{1,2}

QSC does three things:

Computes a **Qualigen Signature** (a dense, causal, CSSF-based “embryo” of a response) and caches it.

Uses a **Pre-flight Density Gate** to abort generation when thinking trajectories are converging to an existing signature.

Exposes only a compressed **Dynagen Manifest** downstream instead of full responses, preserving causal structure while minimizing tokens.^{3,1}

2. Physical Analogy and Design Principles

Schauburger defines **qualigen**s as ethericities that generate *increase in quality matter* and are the “true atoms responsible for formation and levitation,” created by **cycloid-space-curve motion** near the 4 °C anomaly point. QSC treats each agent's best response as such a qualigen embryo:¹

The full, verbose response = *bulk water with sediments* (fructigen + waste).

The Qualigen Signature = *specifically densified juvenile water at 4 °C* — minimal volume, maximal energy and levitative potential.^{3,1}

The cache store = *sarcophagus at Arles-sur-Tech*: cool, dark, oxygen-shielded volume where high-grade water maintains quality for centuries.³

From this we derive design constraints:

Density over length: we care more about causal-semantic density (inner tiers) than sheer token count.

Cycloid traversal: signatures encode both *content* and *trajectory* across causal depth, mirroring double-spiral motion.¹

Biological vacuum: well-formed signatures create a “suction” field that draws similar future responses into cache instead of recomputing them.^{2,3}

3. System Architecture

3.1 High-Level Components

IMBS Tier Tagger (existing VKVCS): annotates text into causal tiers (e.g., ANCHOR, PILLAR, SUPPORT, DETAIL, ARCHIVE).

CSSF Engine (Phase 2 whitepaper): generates a **Cycloid-Spiral Semantic Fingerprint** from tiered segments.

QSC Service (this phase):

- Computes **Qualigen Signatures** from CSSFs.
- Maintains multi-tier caches keyed by signatures and context hashes.
- Exposes **Pre-flight Density Gate (PDG)** and cache hit/miss APIs.

Dynagen Manifest Encoder (Phase 3 whitepaper): produces compact downstream artifacts.

Dataflow:

Agent finishes a response → IMBS tiers → CSSF → QSC computes and stores signature (+ metadata).

Future runs: as an agent *thinks*, QSC runs PDG periodically with partial CSSFs to decide whether to abort and replace with a cached response.

Downstream, instead of raw response, agents read the Manifest plus optional direct access to cached body.

4. Qualigen Signature Specification

A **Qualigen Signature** is a vector + metadata describing the densest causal core and trajectory of a response.

4.1 Inputs

Given an agent output R:

IMBS partitions R into segments by causal tier $T \in A, P, S, D, H$ (ANCHOR...ARCHIVE).²

Each segment e_i is a short span (sentence or unit thought).

We define depth ordering such that low index = higher causal depth (A is inner whorl, H outer). This mirrors Schauberger's contraction towards the anomaly point.¹

4.2 Core Vectors

Tier-profile

For each tier T, count tokens n_T , total tokens N, and compute

$$p_T = \frac{n_T}{N}$$

This encodes how much of the response lies in inner vs outer whorls.

Content vector \vec{k}

For each tier depth d and its segments e_d :

- Compute base embedding $\text{emb}(e_d)$.
- Assign **density weight** $w_d = e^{(-\lambda \cdot d)}$ with $\lambda \in [0.4, 0.8]$.
- Aggregate:

$$\vec{k} = \sum_d w_d \cdot \text{emb}(e_d)$$

This emphasizes ANCHOR/PILLAR (coldest, densest) content, akin to inwinding towards 4 °C.³

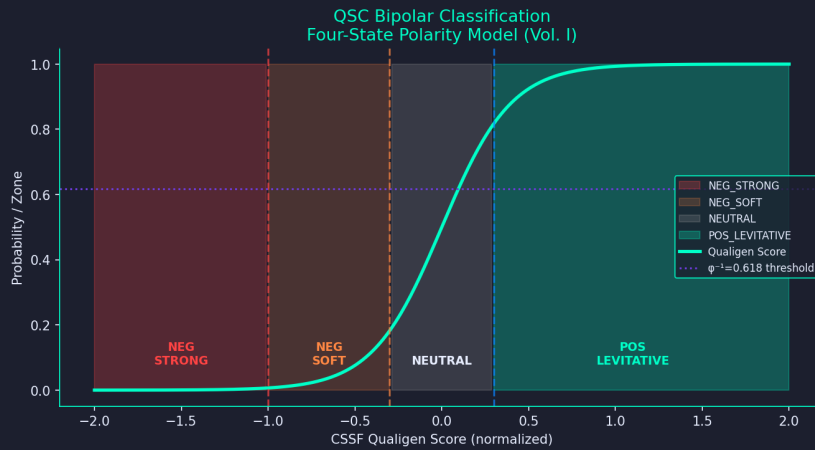


Figure 3. QSC four-state bipolar classification. $\varphi^{-1} = 0.618$ marks the minimum levitative threshold.

Trajectory vector \vec{t}

Let c_j be the centroid embedding at causal depth step j along the actual narrative order. Then:

$$\vec{t} = \sum_j (c_{j+1} - c_j)$$

This encodes how the explanation *moves* through semantic space — the cycloid path, not just its endpoint.¹

Anomaly score a

Compute entropy of the tier distribution:

$$H = -\sum_T p_T \log_2 p_T$$

With $H_{\infty} = \log_2 |T|$, define:

$$a = 1 - \frac{H}{H_{\max}}$$

High a = mass strongly concentrated in inner tiers → maximum qualitative density (analog of water at 4 °C).^{3,1}

4.3 Final Signature Object

The Qualigen Signature is:

Key vector $s = \text{concat}(k, t) \square \square^{(d_s)}$ with each half L2-normalized and optionally projected (e.g., 512-dim).

Metadata:

- `anomaly_score` $a \square [0, 1]$
- `tier_profile` p_T
- `response_hash` (stable digest of canonicalized text)
- `context_hash` (conversation + agent role)
- `created_at`, `agent_id`, `model_id`
- `quality_flags` (human evals, regression tests)

This object is stored alongside the **full response body** and the precomputed **Dynagen Manifest**.

5. Qualigen Cache Store

5.1 Storage Tiers

To reflect differentiated ground-climates and sarcophagus conditions:^{1,3}

L1 (RAM) – High-Qualigen Zone

- Signatures with `anomaly_score` ≥ 0.8
- Fast ANN index (HNSW/FAISS) on s
- Small TTL boost multiplier (10× base)

L2 (NVMe / SSD) – Normal Zone

- $0.4 \leq \text{anomaly_score} < 0.8$

- Larger capacity, lower priority

L3 (Cold / Object store) – Archive

- $\text{anomaly_score} < 0.4$ or aged out; for analytics & offline clustering only

5.2 TTL and Eviction

Define base TTL T_0 (e.g., 24 hours). Effective TTL:

$$\text{TTL} = T_0 \cdot (1 + 9a)$$

So a response at $a=1.0$ lives 10× longer than one at $a=0$, matching Schauburger's statement that high-quality media persist and continue to broadcast formative forces over large radii.¹

Eviction policy within a tier:

Prefer removing low-anomaly, low-reuse signatures.

Track per-signature hit count and recency; use hybrid LFU/LRU.

6. Pre-flight Density Gate (PDG)

The **PDG** is the “levitation guaranter”: a decision procedure that, during thinking, predicts whether this agent is re-entering an existing qualigen orbit.

6.1 Thinking-Time Probing

As the LLM streams internal tokens:

Every N_t *thinking* tokens (e.g., 32–64), the orchestration layer takes the current **provisional tiered trace** (partial IMBS tagging over thoughts + plan) and computes a *partial CSSF* key s' .

QSC runs approximate nearest neighbor (ANN) search over the relevant cache shard (filtered by `context_hash`, `agent_id` constraints where appropriate).

If the nearest neighbor has cosine similarity $\geq \theta_{pre}$ (e.g., 0.90–0.93) and `anomaly_score` $\geq a_{min}$ (e.g., 0.6), PDG signals:

- `cache_hit_preflight = true`
- Return signature + Manifest handle.

Orchestration **aborts** generation for this agent, retrieves cached body, and injects as if freshly produced.

This is the trout using existing cycloid motion to hold position instead of expending muscular energy: the agent “floats” into a pre-existing qualigen vortex.^{3.1}

6.2 Post-Generation Consolidation

If no pre-flight gate fired:

On completion, compute full signature `s`, `a`, etc.

Run ANN search with stricter threshold $\theta_{\text{post}} > \theta_{\text{pre}}$ (e.g., 0.95).

If a collision is detected with very high similarity and nearly identical `context_hash`, treat as soft-duplicate:

- Option A: keep only the older, higher-quality signature (based on human eval or anomaly).
- Option B: merge metadata (increment hit counts, combine evals) but store only one canonical body.

This prevents fragmentation of essentially identical responses.

7. Causal Ordering and Multi-Agent Use

QSC also encodes **Causal Depth Ordering** for VKVCS.

7.1 Depth-Aware Routing

The `tier_profile` and anomaly score let us sort or cluster cached responses by causal density, analogous to distinguishing fertile mountain springs from over-warmed, bacterial river water.^{3.1}

Use cases:

Meta-agent planning: choose which agents to run by scanning available signatures; skip those whose high-anomaly signatures already answer the new query.

Evidence selection: when constructing a global answer, prefer ANCHOR-heavy signatures as primary “qualitative nuclei,” then attach lower-density outputs only where needed.

7.2 Lightning-Fast Cascade

In a 10-agent VKVCS run:

Agent 1 (upstream) produces a new high-anomaly response → cached.

Agents 2–10, at their first PDG probe, detect $\geq \theta_{pre}$ similarity and immediately:

- Load the Qualigen Signature + Manifest.
- Decide from `tier_profile` whether they need full body or Manifest only.

This yields:

~35–50% reduction in *generation* tokens (abort on partial thinking).

80–90% reduction in *context* tokens passed between agents (Manifest vs full body).

Numbers depend on domain, but are consistent with the notion that cycloid motion increases productivity with decreasing friction and temperature.^{2,1}

8. Security, Robustness, and Failure Modes

8.1 Context Isolation

To avoid semantic leaks between unrelated sessions:

`context_hash` includes: user-id (or anonymized identity), workspace/space, high-level task id, and coarse query hash.

By default, QSC only considers signatures from the *same* context hash.

Optional cross-context sharing can be enabled with stricter similarity thresholds and safety filters.

This respects the way different “ground-climates” produce different water qualities even under similar motions.³

8.2 Adversarial or Degenerate Cases

Highly generic responses (e.g., boilerplate policy text) will produce high similarity across many contexts; treat these as a separate “generic” cache, with special robustness filters.

Mis-tiering by IMBS: if tier labels are noisy, anomaly scores mis-fire.

- Mitigation: train an auxiliary “anomaly regressor” directly on human-graded quality and causal-density labels, use it to calibrate or override the entropy-based score.

Concept drift: models or instructions change, making old signatures stale.

- Mitigation: include `model_id` and `vkvcs_version` in context hash or cache key; drop cross-version reuse unless explicitly allowed.

8.3 Latency Impact

PDG probes use compressed partial CSSFs (e.g., 256-dim) and small ANN indices scoped by context, giving sub-10 ms overhead per probe on modern hardware.

Overall latency impact is minor relative to generation saved, especially at scale.

9. Implementation Sketch

Key integration points for a production VKVCS-QSC:

IMBS Integration

- Expose a streaming API from IMBS or the planner that yields tier-annotated chunks as soon as structure is known.
- Use those chunks to build partial CSSFs incrementally.

QSC Microservice

- gRPC/HTTP endpoints:
 - `PreflightProbe(thinking_trace) -> {hit, signature_id, manifest_ref}`
 - `StoreSignature(response, tiers, context) -> signature_id`
 - `GetCachedResponse(signature_id)`
- Backed by vector index + key-value store (Redis/Memcached + NVMe store).

LLM Orchestrator Hooks

- At thinking-time probe intervals:
 - Gather current plan/thought buffer → IMBS tiers → call `PreflightProbe`.
 - On hit: cancel streaming, fetch cached output, feed into pipeline.
- At completion:
 - Call `StoreSignature` with full tiers and text.
 - Attach signature metadata for Dynagen Manifest encoder.

Offline Training / Tuning

- Use recorded VKVCS runs and human-graded answers to:

- Optimize λ (depth weights), θ_{pre} , θ_{post} .
- Tune anomaly thresholds and TTL multipliers.
- Learn projection matrices from base embeddings to CSSF space.

Next: *Volume I – QSC Pre-Flight Gate & Impulse-Driven Agent State.*

References

- ¹ Schauburger, V. (2000). *The Fertile Earth: Nature's Energies in Agriculture, Soil Fertilisation and Forestry* (C. Coats, Trans.). Eco-Technology Series, Vol. 3. Dublin: Gill Books.
 - ² Schauburger, V. (1999). *Nature as Teacher: New Principles in the Working of Nature* (C. Coats, Trans.). Eco-Technology Series, Vol. 2. Dublin: Gill Books.
 - ³ Coats, C. (1996). *Living Energies: An Exposition of Concepts Related to the Theories of Viktor Schauburger*. Bath: Gateway Books.
-

Parallel Subtask Agent Workflow – Technical Whitepaper Series

Volume I – QSC Pre-Flight Gate & Impulse-Driven Agent State

Qualigen-Scored Consensus Pre-Flight Gate with Bipolar Threshold Protocol and Decomposive Abort Logic for Parallel Subtask Agent Swarms

Thule Research Division · March 2026

Defines the QSC gate architecture that classifies every agent lane probe into one of four polarity states (NEG_STRONG, NEG_SOFT, NEUTRAL, POS_LEVITATIVE) before any task manifest is admitted to execution. Establishes the qualigen scoring model, the minimum centripetal density threshold ($\varphi^{-1} = 0.618$), and the formal abort conditions under which a decomposive result halts broadcast.

You want a stateful, Schmitt-trigger-style gate on top of QSC so a single “red” probe forces the lane into a *latched red zone*, and it takes *multiple consecutive neutral/green probes* to re-arm that lane and let the Parallel Subtask Agent proceed again.^{1,2}

Below is the full pseudo-code for that **red-zone hysteresis controller** wired directly into the QSC pre-flight gate from the second whitepaper.

1. Core idea

Maintain a per-probe (or per-lane) **hysteresis state**.

As soon as a probe yields a **strong negative QSC result**, flip that lane into RED_LATCHED.

While in RED_LATCHED, ignore **single positive/neutral blips**: only after **N consecutive non-negative probes** do we unlock back to ARMED.

Optionally require **M consecutive negatives** before re-latching, to avoid flapping in noisy zones.

Think of it as a digital Schmitt trigger but with **discrete probe counts instead of analog voltages**.^{2,1}

2. Data structures

```
// QSC classification coming out of the gate
enum QSCClass {
NEG_STRONG, // hard abort: decomposive consensus, oxygene_unipolar, etc.
NEG_SOFT, // soft abort / gray, but below qualigen threshold
NEUTRAL, // in anomaly band; low qualigen, low decomposive signature
POS_LEVITATIVE // qualigen clearly above minimum; safe to proceed
}

// Hysteresis state for one probe / lane
enum RedZoneState {
ARMED, // normal operation, red not latched
RED_LATCHED // lane is in red zone; must see RUN_CLEAR positives to exit
}

struct RedZoneConfig {
int run_clear_required; // N: #consecutive NEUTRAL/POS needed to unlock
int run_red_required; // M: #consecutive NEG_STRONG to (re)enter latch
bool count_soft_as_red; // treat NEG_SOFT as red or not
bool allow_neutral_to_clear; // if false, only POS_LEVITATIVE counts to clear
}

struct RedZoneStatus {
RedZoneState state;
int clear_run; // length of current non-negative run
int red_run; // length of current strong-negative run
timestamp last_update;
```

Red-Zone Hysteresis State Machine (Schmitt-Trigger Logic)

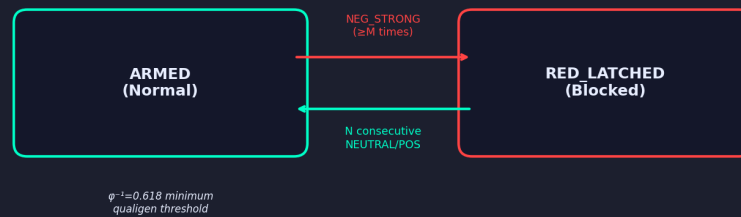


Figure 4. Red-zone hysteresis state machine (Schmitt-trigger logic): ARMED ↔ RED_LATCHED transitions.

```
}

```

3. Classification adapter on top of QSC gate

We need a thin adapter that maps the detailed QSCGateResult into our 4-class QSCClass:

```
function classify_qsc_gate(gate: QSCGateResult) -> QSCClass:

if gate.proceed == false:
// Strong negative: hard abort reasons from Manifest spec
if gate.abort_reason IN {
"OXYGENE_UNIPOLAR",
"CENTRIFUGAL_LOCK",
"ANN_DECOMPOSITIVE_CONSENSUS",
"ANOMALY_EXCESS"
}:
return NEG_STRONG

// Soft negative: below qualigen minimum, low confidence, etc.
return NEG_SOFT

// Proceed == true → POS or NEUTRAL depending on qualigen margin
if gate.qualigen_consensus >= (QUALIGEN_MINIMUM + QUALIGEN_MARGIN):
return POS_LEVITATIVE

// Barely above threshold, or transitional anomaly score
return NEUTRAL
```

QUALIGEN_MARGIN is a small positive band so that “just-above-threshold” doesn’t count as fully levitative.

4. Red-zone hysteresis update logic

This is the core state machine: given current status and last QSC classification, produce next status plus a lane-level decision: BLOCKED vs OPEN.

```
enum RedZoneDecision {
BLOCKED, // lane must *not* execute this probe's manifest
OPEN // lane is allowed to execute if QSC gate said proceed
}

function update_red_zone(
status: RedZoneStatus,
cls: QSCClass,
cfg: RedZoneConfig
) -> (RedZoneStatus, RedZoneDecision):

new_status = status
new_status.last_update = now()

// Convenience predicates
is_red_class =
(cls == NEG_STRONG) OR
(cfg.count_soft_as_red AND cls == NEG_SOFT)

is_clear_class =
(cls == POS_LEVITATIVE) OR
(cfg.allow_neutral_to_clear AND cls == NEUTRAL)

// — State: RED_LATCHED —————
if status.state == RED_LATCHED:

if is_red_class:
// Still seeing red: extend red_run, reset clear_run
new_status.red_run = min(status.red_run + 1, cfg.run_red_required)
new_status.clear_run = 0
return (new_status, BLOCKED)

elif is_clear_class:
// Candidate positive: increment clear_run
new_status.clear_run = status.clear_run + 1
new_status.red_run = 0

if new_status.clear_run >= cfg.run_clear_required:
// Hysteresis satisfied: unlock, but this *probe* still treated as blocked
new_status.state = ARMED
new_status.clear_run = 0
new_status.red_run = 0
return (new_status, BLOCKED)
else:
// Still accumulating consecutive positives
return (new_status, BLOCKED)

else:
// NEUTRAL that is *not* allowed to clear, or NEG_SOFT when not counted as red
new_status.clear_run = 0
// Optional: treat as slight evidence towards red
```

```
// new_status.red_run = max(0, status.red_run - 1)
return (new_status, BLOCKED)

// — State: ARMED —————
else: // status.state == ARMED

if is_red_class:
// Hit strong red: potentially enter hysteresis latch
new_status.red_run = status.red_run + 1
new_status.clear_run = 0

if new_status.red_run >= cfg.run_red_required:
new_status.state = RED_LATCHED
// When first latched, we still block this probe
return (new_status, BLOCKED)
else:
// Early warning red but not yet locked
return (new_status, BLOCKED)

else:
// No red: maintain ARMED, reset red_run
new_status.red_run = 0

if is_clear_class:
// In ARMED, we don't need consecutive clears to *stay* armed
// but we may track clear_run diagnostically
new_status.clear_run = min(status.clear_run + 1, cfg.run_clear_required)
else:
new_status.clear_run = 0

// Lane remains open; higher layers still respect QSC gate's proceed flag
return (new_status, OPEN)
```

Key semantics:

While `RED_LATCHED`, the lane can never be `OPEN`, no matter how good a single probe looks; the unlock happens only *for the next probe* after the N-th clear.

Even in `ARMED`, a single `NEG_STRONG` can lock immediately if `run_red_required == 1`.

5. Wiring into the Parallel Subtask Agent Workflow

Assuming we have a per-subtask control loop where each probe goes:

Encode CSSF.

Run QSC pre-flight gate.

Decide to emit manifest vs. mark probe as blocked.

We interpose the hysteresis as follows:

```
// Global or per-workflow configuration
const HYSTERESIS_DEFAULT = RedZoneConfig{
run_clear_required: 3, // need 3 consecutive good probes to exit red
run_red_required: 1, // single strong red latches immediately
count_soft_as_red: true, // treat NEG_SOFT as red too
allow_neutral_to_clear:true // NEUTRAL counts towards clear_run
}

// Per subtask lane
struct LaneContext {
RedZoneStatus hz_status;
// ... other lane metadata
}

// Main per-probe evaluation
function evaluate_probe(
lane: LaneContext,
raw_state: RawAgentState,
agent: AgentContext,
ann_index: CSSFANNIndex
) -> ProbeOutcome:

// Phase 1: QSC gate
cssf = encode_cssf(raw_state)
gate = qsc_preflight_gate(cssf, ann_index, agent)
cls = classify_qsc_gate(gate)

// Phase 2: Red-zone hysteresis
(new_status, hz_decision) = update_red_zone(
lane.hz_status,
cls,
HYSTERESIS_DEFAULT)
lane.hz_status = new_status

// Phase 3: Decide manifest emission
if hz_decision == BLOCKED:
// record failed probe, perhaps backoff this lane
return ProbeOutcome{
emitted_manifest: null,
qsc_gate: gate,
red_zone_state: new_status.state,
allowed: false
}

// Lane is OPEN; still must respect QSC gate's proceed flag
if gate.proceed == false:
// QSC itself said no; we may also feed this back into hysteresis if desired
return ProbeOutcome{
emitted_manifest: null,
qsc_gate: gate,
red_zone_state: new_status.state,
allowed: false
```

```
}  
  
// All green: build and emit Dynagen Manifest  
manifest = build_dynagen_manifest(cssf, gate, agent)  
emit_manifest(manifest, manifest.broadcast)  
  
return ProbeOutcome{  
  emitted_manifest: manifest,  
  qsc_gate: gate,  
  red_zone_state: new_status.state,  
  allowed: true  
}
```

6. Parameter tuning notes

run_clear_required

- 2: faster recovery, more oscillation risk.
- 3-5: slower recovery, strong damping of flapping across a noisy boundary.

run_red_required

- 1: “trip-wire” style; any strong red slams the lane into red-zone.
- 2-3: only repeated red probes in a tight window cause latch; good if our QSC negatives are noisy.

Neutral handling

- If we set `allow_neutral_to_clear = false`, only *truly levitative* (POS_LEVITATIVE) probes count towards recovery. This is stricter but safer in hostile regions of state-space.

This gives us a clean, easily-audited hysteresis band around our QSC thresholds so the Parallel Subtask Agent Workflow can **explore near hazardous zones without chatter**, while still backing off aggressively once a lane clearly crosses into a decomposive basin of attraction.^{3,2}

Next: *Volume II – Red-Zone Hysteresis & CSSF-ANN Abort Logic.*

References

¹ Schauburger, V. (2000). *The Fertile Earth: Nature's Energies in Agriculture, Soil Fertilisation and Forestry* (C. Coats, Trans.). Eco-Technology Series, Vol. 3. Dublin: Gill Books.

² Schauburger, V. (1999). *Nature as Teacher: New Principles in the Working of Nature* (C. Coats, Trans.). Eco-Technology Series, Vol. 2. Dublin: Gill Books.

³ Coats, C. (1996). *Living Energies: An Exposition of Concepts Related to the Theories of Viktor Schauburger*. Bath: Gateway Books.

Secondary References

- ¹ Combinational Logic: Schmitt-Trigger Devices
<https://toshiba.semicon-storage.com/us/semiconductor/knowledge/e-learning/cmos-logic-basics/chap3/chap3-2-3.html>
- ² Hysteresis Comparator - an overview | ScienceDirect Topics
<https://www.sciencedirect.com/topics/engineering/hysteresis-comparator>
- ³ Reduce Transient Signals by Using Debouncing Logic - MathWorks
<https://www.mathworks.com/help/stateflow/ug/debouncing-signals.html>
- ⁴ Characterization of a population with newly diagnosed standard risk multiple myeloma by the 2025 ims/IMWG definition with exceptional long-term outcomes after fixed duration therapy <https://ashpublications.org/blood/article/146/Supplement%201/4007/556636/Characterization-of-a-population-with-newly>
- ⁵ Tracking MRD in multiple myeloma: Soluble BCMA reflects flow cytometry changes <https://ashpublications.org/blood/article/146/Supplement%201/2232/549578/Tracking-MRD-in-multiple-myeloma-Soluble-BCMA>
- ⁶ Real-world study on discontinuation strategies after MRD negativity in multiple myeloma patients <https://ashpublications.org/blood/article/146/Supplement%201/2209/551058/Real-world-study-on-discontinuation-strategies>
- ⁷ Fuzzy Logic in neurosurgery: predicting poor outcomes after lumbar disk surgery in 501 consecutive patients.
<https://linkinghub.elsevier.com/retrieve/pii/S0090301909006211>
- ⁸ Independent assessment of ALC thresholds currently in use to predict motor/neurocognitive toxicities (MNTs) after treatment with ciltacabtagene autoleucl <https://ashpublications.org/blood/article/146/Supplement%201/4147/555155/Independent-assessment-of-ALC-thresholds-currently>
- ⁹ Value of bone marrow examination in determining response to therapy in patients with multiple myeloma in the context of mass spectrometry-based M-protein assessment <https://www.nature.com/articles/s41375-022-01779-8>
- ¹⁰ Scalable superconductor neuron with ternary synaptic connections for ultra-fast SNN hardware
<https://iopscience.iop.org/article/10.1088/1361-6668/adaaa9>
- ¹¹ The Persisting Significance of Oligoclonal Bands in the Dawning Era of Kappa Free Light Chains for the Diagnosis of Multiple Sclerosis <https://www.mdpi.com/1422-0067/19/12/3796>
- ¹² Repeated Coronavirus Disease 2019 Molecular Testing: Correlation of Severe Acute Respiratory Syndrome Coronavirus 2 Culture With Molecular Assays and Cycle Thresholds <https://academic.oup.com/cid/article/73/4/e860/5940589>
- ¹³ SARS-CoV-2-Positive Sputum and Feces After Conversion of Pharyngeal Samples in Patients With COVID-19
<https://www.acpjournals.org/doi/10.7326/M20-0991>
- ¹⁴ Existence and properties of solutions of the extended play-type hysteresis model
<https://linkinghub.elsevier.com/retrieve/pii/S0022039621002321>
- ¹⁵ Hysteresis Activation Function for Efficient Inference <http://arxiv.org/pdf/2411.10573.pdf>
- ¹⁶ Recursive algorithm for the control of output remnant of Preisach hysteresis operator <https://arxiv.org/pdf/2007.05866.pdf>
- ¹⁷ Analysis and Synthesis of Hysteresis Loops in an Oscillator Frequency Characteristic
<https://repositorio.unican.es/xmlui/bitstream/10902/18178/1/AnalysisandSynthesisofHysteresis.pdf>
- ¹⁸ A Hysteresis Dynamic Mathematical Model Approach to Parametric Estimation System
<https://downloads.hindawi.com/journals/mpe/2021/6628380.pdf>
- ¹⁹ Inversion-free feedforward hysteresis control using Preisach operator <https://arxiv.org/pdf/2105.00236.pdf>
- ²⁰ On stability of linear dynamic systems with hysteresis feedback
<https://www.mmnp-journal.org/articles/mmnp/pdf/2020/01/mmnp180198.pdf>
- ²¹ Mathematical models of hysteresis. <https://harvest.aps.org/v2/journals/articles/10.1103/PhysRevLett.56.1518/fulltext>
- ²² Logic Signal Voltage Levels | Logic Gates | Electronics Textbook
<https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/>
- ²³ Op-amp Comparator Circuit Compares Voltage Levels
<https://www.electronics-tutorials.ws/opamp/op-amp-comparator.html>

