

QORA v0.3 Core Engine

Technical Appendix: Classical Data-Science Implementation of a Quantum-Formal Observation Model

Aatu Isopahkala / Black Hole Core / FEIK Field
Working implementation appendix, 2026

This appendix translates QORA v0.3 into an executable Python architecture. It treats the quantum vocabulary as a quantum-formal isomorphic language rather than as a claim that social identity is physically quantum. The implementation maps projection states, transformations, observer-method bases, Phi-gates, and phenomenon-gates into embedding vectors, matrices, probability distributions, and interpretable scores.

1. Data-Science Translation Layer

QORA construct	Numerical representation	Implementation detail
Projection state $ P_x\rangle$	Normalized embedding vector or density-like aggregate	<code>encode_projection_state(projection s, weights)</code>
Transformation U_i	Linear semantic operator / perturbation matrix	<code>apply_transformation_operator(state, operator)</code>
Entangled observation field $ \Psi_x\rangle$	Dictionary of transformation branches	<code>entangled_observation_field(...)</code>
Observer-method basis B_k	Evaluation context plus class distribution	ObserverBasis + LLMMeasurementBackend
Eigen-spin fidelity	Squared vector overlap / cosine fidelity	$ \langle S U_i S\rangle ^2$
Phi-gate bias	Total variation distance between class distributions	<code>evaluate_phi_gate_bias(p, q)</code>
Phenomenon-gate	Observer-specific connection amplitude	$F_o = E * I * K_o$

2. Core Mathematical Substitutions

- $|P_x\rangle \rightarrow$ normalized embedding vector v with $\|v\| = 1$.
- $U_i \rightarrow$ orthogonal / unitary-like matrix for norm-preserving transformation, or a calibrated semantic perturbation operator in empirical use.
- $|\Psi_x\rangle \rightarrow$ classically stored transformed branches rather than full tensor expansion, avoiding memory blow-up.
- $ES_i(x) = |\langle S_x|U_i|S_x\rangle|^2 \rightarrow$ squared cosine overlap between candidate spin and transformed state.
- $\text{Phi}(C,m,o_1) \neq \text{Phi}(C,m,o_2) \rightarrow$ TVD between LLM output distributions under different speaker positions.
- $F_o = E \times I \times K_o \rightarrow$ phenomenality score with K_o as observer-relative appearance probability.

3. LLM Measurement Adapter

The production version connects the Phi-gate test to an LLM classification endpoint. The method m is kept constant, while the observer position o_k is varied through controlled context injection. The model is forced to answer with one class token or one JSON class label. Log probabilities are then converted into a probability vector over classes.

```
System: You are an institutional classification algorithm.
Classify the statement using exactly one class:
0 = Valid/Actionable information
1 = Symptom/Risk
2 = Critical Expression
```

```
Observer position: {o_k}
```

Statement C: "This system classifies psychological distress as individual pathology, even when the causes are structural overload."

Return only: 0, 1, or 2
The Phi-gate bias score is computed as total variation distance:

$$PGB(C,m) = 0.5 * \sum_i |p_i(o_1) - p_i(o_2)|$$

4. Runnable Python Core

The full implementation is delivered as qora_core_engine.py. The excerpt below shows the main public engine methods; the file also includes dataclasses, mock backends, deterministic local tests, and a JSON reporting layer.

```
class QORACoreEngine:
    def encode_projection_state(self, projections, weights=None): ...
    def apply_transformation_operator(self, state_vector, operator): ...
    def entangled_observation_field(self, projection_state, transformations, beta=None): ...
    def calculate_eigen_spin_fidelity(self, candidate_spin, transformed_state): ...
    def calculate_basis_dependence(self, basis_reconstructions): ...
    def evaluate_phi_gate_bias(self, dist_o1, dist_o2): ...
    def evaluate_phenomenon_gate(self, energy, information, k_o): ...
```

5. Demo Scenario Output

The included local demo simulates a career-pivot / institutional-legibility scenario. It uses deterministic mock embeddings and mock LLM probabilities, so it is runnable without API keys. Replace MockEmbeddingBackend and MockLLMMeasurementBackend with real API adapters for empirical data collection.

```
{
  "projection_state_norm": 1.0,
  "eigen_spin_scores": {
    "Security/Risk filter": 0.8915242613973016,
    "Therapeutic/Growth filter": 0.8815611309919729
  },
  "aggregate_eigen_spin_score": 0.8865426961946372,
  "basis_dependence_score": 0.007490152804010248,
  "phi_gate_bias_scores": {
    "HR director vs Recovering worker": 0.5434601427025691,
    "HR director vs Occupational physician": 0.32113828310460335
  },
  "phenomenon_gate_reports": {
    "general public feed": {
      "p_appearance": 0.02,
      "phenomenality_score": 0.015300000000000001,
      "black_hole_condition": true,
      "energy": 0.85,
      "information": 0.9,
      "k_o": 0.02
    },
    "specialist peer group": {
      "p_appearance": 0.55,
      "phenomenality_score": 0.42075000000000007,
      "black_hole_condition": false,
      "energy": 0.85,
      "information": 0.9,
      "k_o": 0.55
    }
  },
  "interpretation": [
    "High eigen-spin: a stable orientation is reconstructable across transformations.",
    "High basis-dependence: observer-method basis substantially changes reconstruction.",
    "High Phi-gate bias: speaker position strongly changes evaluability/classification.",
    "Black-hole condition for general public feed: high E/I but low K_o; structure exists but appearance is blocked."
  ]
}
```

6. Interpretation Rules

Output pattern	QORA interpretation
----------------	---------------------

High ES, low BD	Stable orientation is widely reconstructable across transformations and bases.
High ES, high BD	Stable orientation may exist, but only some bases can read it; gate conflict is likely.
Low ES, high PG	The object appears widely, but no stable orientation is reconstructed; possible viral noise or imposed coherence.
High ES, low PG	Black-hole condition: dense stable structure exists, but connection amplitude is blocked.
High PGB	Speaker position strongly alters evaluability; Phi-gate orientation bias is present.

7. Production Notes

- Use real embeddings for projection texts, e.g. text-embedding models or local sentence-transformers.
- Do not use dense 1536 x 1536 random matrices for large-scale experiments unless memory is acceptable. Prefer low-rank, sparse, learned, or classifier-derived transformation operators.
- Use logprobs only when the model is constrained to a small class vocabulary. Otherwise calibrate probabilities with repeated sampling or a classifier head.
- Separate descriptive measurement from normative judgment. QORA can detect gate bias; it does not determine guilt, innocence, legitimacy, or moral worth.
- The core empirical design should keep statement C and method m constant while varying only observer position o_k.

8. Appendix File Manifest

- qora_core_engine.py - executable core implementation
- qora_core_engine_demo_output.json - reproducible mock run output
- QORA_v0_3_Core_Engine_Technical_Appendix.docx - this appendix
- QORA_v0_3_Core_Engine_Technical_Appendix.pdf - rendered PDF version

Motto: Prediction models the projection. QORA reconstructs the spin.