

QORA v0.3 Complete Set

Working Paper + Technical Appendix + Python Core Engine

Aatu Isopahkala / Black Hole Core / FEIK Field - 2026

QORA v0.3

Quantum Observation-Resistance Algorithm

A quantum-formal extension of observation-algebraic resistance

Aatu Isopahkala / Black Hole Core / FEIK Field

Working Paper v0.3 - 2026

Core is not color. Core is spin.

Prediction models the projection. QORA reconstructs the spin.

Abstract

This paper introduces QORA v0.3, the Quantum Observation-Resistance Algorithm: a quantum-formal, isomorphic model for reconstructing eigen-spin across projections, transformations, observer-method bases, and gates of evaluability and phenomenality. QORA does not claim that persons, archives, institutions, or cultural formations are physically quantum systems. Rather, it uses quantum algorithmic structure as a formal language for basis-dependent legibility, transformation-invariance, observer-relative reconstruction, and platform-mediated appearance. The model begins from the observation-algebraic claim that predictive power models projections but does not thereby own the generative orientation that produces them. QORA encodes a projection field as a state, entangles it with transformations through a controlled operator, measures the resulting observation field relative to observer-method bases, and tests whether a reconstructed orientation remains directionally stable under transformation. The Phi-gate is modeled as a basis-dependent projector that tests not only evaluability but orientation bias. The phenomenon-gate is modeled as an observer-relative connection amplitude. The output is not a prediction of the next projection, but a map of where prediction fails because the observational frame fails. Version v0.3 adds a case study on a radical climate activist and institutional non-commutativity, demonstrating how measurement order changes whether an actor becomes legible as threat, art, critique, or public phenomenon.

Keywords: observation algebra; quantum formalism; eigen-spin; eigenspinor resonance; Phi-gate; phenomenon-gate; FEIK; platform culture; predictive profiling; basis-dependent legibility; Black Hole Core.

1. Introduction: From Prediction to Reconstruction

Classical profiling asks what a person, project, or archive will do next. It gathers traces, estimates transitions, and treats the future as an extrapolation from the visible projection-field. In this model, the object of power is not only identity, but movement: vocabulary shifts, rhythm changes, affective transitions, topic drift, register switching, conflict responses, and platform behavior.

Observation-algebraic resistance begins from a different premise. Projection is not orientation. A system may model the visible traces of x without reconstructing the generative direction of x . The central distinction is therefore:

$$\text{projection} \neq \text{orientation}$$

$$M(\{P_j(x)\}) \rightarrow s_x^{(o,m)}$$

Here M is a predictive model, $P_j(x)$ are observable projections, and $s_x^{(o,m)}$ is a reconstructed orientation relative to observer position o and method m . QORA formalizes this distinction. It does not try to predict the next projection. It asks what remains directionally reconstructable when projections are transformed, measured in different bases, and passed through gates of evaluability and appearance.

Classical profiling asks what a person will do next. QORA asks what remains directionally reconstructable when the observational frame changes.

2. Observation Algebra: Minimal Definitions

Observation algebra is the study of how objects, claims, identities, and phenomena transform under observer position, measurement method, projection, transformation, and connection. It treats observation not as passive reception, but as an operation that makes some components legible and others invisible.

Table 1. Core terms

Term	Formal role	Meaning in QORA
Projection $P_j(x)$	Visible trace	Document, song, image, prompt, diagnosis, record, post, or behavioral trace.
Transformation T_i	Operation on projection	Classification, translation, diagnosis, distribution, naming, risk assessment, or reframing.
Observer-method basis B_k	Measurement frame	Observer position o and method m through which x becomes legible.
Eigen-spin $s_x^{(o,m)}$	Reconstructed orientation	Directional invariance reconstructed across projections and transformations.
Phi-gate Π_Φ	Evaluability projector	Gate that decides whether a claim or orientation can count as evaluable knowledge.
Phenomenon-gate	Connection amplitude filter	Gate that regulates whether a coherent structure appears for observer o .

K _o	Observer-specific connection	The available connection through which x becomes phenomenal for observer o.
----------------	------------------------------	---

This table is important because QORA is not a claim that social reality is literally quantum. It is a claim that quantum formalism provides a useful isomorphic language for basis-dependence, non-commutativity, transformation behavior, and amplitude-mediated appearance.

3. Quantum-Formal Status of the Model

QORA is quantum-formal rather than quantum-physical. It does not assert that persons are wavefunctions, archives are particles, or institutions are quantum systems. The model instead borrows formal features from quantum theory because they capture features that linear profile-space models often flatten: basis dependence, measurement order, entanglement between object and operation, phase-preserving transformations, and amplitude filtering.

The relation is isomorphic: mathematical structure is translated into an observation-algebraic register. The value of the translation lies in what it makes visible. A vector profile can represent coordinates. A quantum-formal model can represent how coordinates become legible only relative to basis, measurement, operation, and connection.

4. Projection State

Let x be a person, project, archive, or cultural formation. Let $P_j(x)$ denote its observable projections: documents, songs, images, prompts, diagnoses, institutional records, platform traces, utterances, applications, or behavioral traces. QORA begins by encoding these projections into a normalized projection state:

$$|P_x\rangle = \sum_j \alpha_j |P_j(x)\rangle$$

$$\sum_j |\alpha_j|^2 = 1$$

The coefficient α_j represents the weight, salience, energy, information-density, or contextual importance of projection $P_j(x)$. The state $|P_x\rangle$ is not the person or archive itself. It is the structured distribution of the surfaces through which x becomes visible.

5. Transformation Register and Entangled Observation Field

Let T_i denote transformations applied to the projections of x . These may include algorithmic classification, diagnosis, translation, platform distribution, bureaucratic naming, artistic reframing, institutional interpretation, risk assessment, or commercial profiling. The transformation register is prepared as:

$$|T\rangle = \sum_i \beta_i |T_i\rangle$$

$$\sum_i |\beta_i|^2 = 1$$

The joint initial state is:

$$|T\rangle \otimes |P_x\rangle = \sum_{i,j} \beta_i \alpha_j |T_i\rangle |P_j(x)\rangle$$

A controlled transformation operator U_T acts as:

$$U_T = \sum_i |T_i\rangle\langle T_i| \otimes U_i$$

where U_i is the operation corresponding to transformation T_i . Applying U_T gives:

$$|\Psi_x\rangle = U_T(|T\rangle \otimes |P_x\rangle) = \sum_{i,j} \beta_i \alpha_j |T_i\rangle U_i |P_j(x)\rangle$$

$$|\Psi_x\rangle = \sum_{i,j} \beta_i \alpha_j |T_i\rangle |T_i(P_j(x))\rangle$$

This is the entangled observation field of x . The projection is now bound to the transformation under which it became legible. Conceptually, the object is no longer merely observed. It is observed-as-transformed.

6. Observer-Method Basis

Let $B_k = (o_k, m_k)$ denote an observer-method basis, where o_k is an observer position and m_k is a measurement or interpretive method. Examples include doctor, patient, bureaucrat, artist, AI model, platform algorithm, employer, researcher, security institution, listener, reader, or public audience.

QORA measures the entangled observation field relative to B_k :

$$M_{\{B_k\}}(|\Psi_x\rangle) \rightarrow s_x^{(o_k, m_k)}$$

The result is not the absolute true spin of x . It is a basis-relative reconstruction of orientation. This preserves the non-essentialist claim that spin is reconstructed rather than owned:

$$s_x^{(o,m)} := \text{Inv}_{(o,m)}(\{T_i(P_j(x))\})$$

The eigenspinor resonance is useful here. In quantum mechanics, a spin state becomes determinate relative to a chosen spin operator, measurement axis, and basis. It is not determinate across every possible measurement frame. Isomorphically, orientation becomes legible only through a basis, and changing the basis changes what can appear as determinate.

Spin does not appear without direction. Direction does not appear without basis. The basis is not neutral.

7. Eigen-Spin Test

The eigen-spin hypothesis is supported when a reconstructed orientation remains directionally stable under transformation. The formal test is:

$$U_i |S_x\rangle \approx \lambda_i |S_x\rangle$$

Since U_i is unitary, its eigenvalue satisfies:

$$|\lambda_i| = 1$$

$$\lambda_i = e^{i\theta_i}$$

The transformation does not destroy the state norm. It may alter phase, visibility, register, intensity, or appearance while preserving a recognizable direction. This is the formal analogue of the observation-algebraic claim: the form changes; the spin remains.

A quantum-formal score can be written through fidelity:

$$ES_i(x) = |\langle S_x | U_i | S_x \rangle|^2$$

$$ES(x) = (1/N) \sum_i |\langle S_x | U_i | S_x \rangle|^2$$

A high $ES(x)$ means that the same reconstructed orientation remains stable across multiple transformations. A low $ES(x)$ may mean that no stable orientation is reconstructable, that the transformation destroyed the relevant signal, or that the chosen observer-method basis failed to reveal the correct component of orientation.

8. Cross-Basis Reconstruction

QORA compares reconstructions across multiple observer-method bases:

$$s_x^{(o_1, m_1)} \approx s_x^{(o_2, m_2)}$$

If independent bases reconstruct similar orientations, the model gains intersubjective strength. If they diverge:

$$s_x^{(o_1, m_1)} \neq s_x^{(o_2, m_2)}$$

the question is not immediately which observer is correct, but what difference in observer position, method, measurement chain, basis, or gate produced the divergence. This makes QORA a diagnostic tool for basis-dependence.

$$BD(x) = \text{Var}(s_x^{(o_1, m_1)}, s_x^{(o_2, m_2)}, \dots, s_x^{(o_n, m_n)})$$

A high basis-dependence score may indicate instability in the object, but it may also reveal institutional or platformic control over legibility.

9. Phi-Gate as Basis-Dependent Projector

The Phi-gate is modeled as a projector:

$$\Pi_\Phi |C, m, o\rangle = |C, m, o\rangle, \text{ if evaluable under } (m, o)$$

$$\Pi_\Phi |C, m, o\rangle = 0, \text{ if not evaluable under } (m, o)$$

The political extension is:

$$\Phi(C, m, o_1) \neq \Phi(C, m, o_2)$$

Here the claim C and method m are held constant while observer position o changes. Context is not merely noise added to the signal. Context is part of the operator. In quantum-formal language, the Phi-gate is not a neutral universal filter. It is a basis-dependent projector.

$$PGB(C, m) = |\Phi(C, m, o_1) - \Phi(C, m, o_2)|$$

A high Phi-gate bias score means that identical content is accepted or rejected differently depending on the position from which it is spoken.

Power does not merely control what can be said. It controls from which orientation speech can become knowledge.

10. Non-Commutativity of Institutional Measurement

If two observer-method bases measure the same claim in different orders, the result may differ:

$$\Pi_\Phi^{(o_1, m_1)} \Pi_\Phi^{(o_2, m_2)} \neq \Pi_\Phi^{(o_2, m_2)} \Pi_\Phi^{(o_1, m_1)}$$

This models institutional legibility. A claim first measured through a psychiatric basis and then through a political basis may not produce the same result as the same claim first measured through a political basis and then through a psychiatric basis. The order of measurement matters.

The same statement is not the same event when it enters through a different gate.

11. Phenomenon-Gate as Connection Amplitude

The FEIK model defines phenomenality as:

$$F = E \times I \times K$$

For observer-relative phenomenality:

$$F_o = E \times I \times K_o$$

where K_o is the observer-specific connection to the structure. QORA translates K_o into an appearance amplitude:

$$A_o = \sqrt{K_o}$$

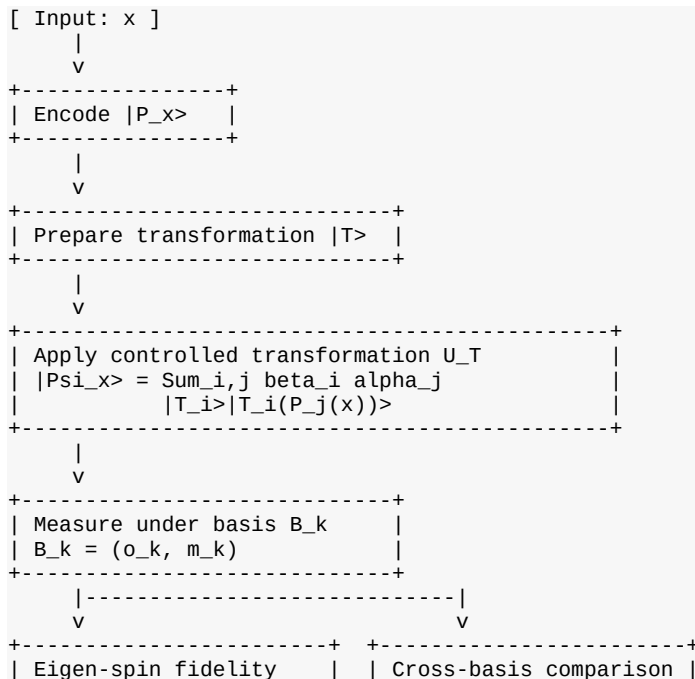
$$P(\text{phenomenon for } o) = |A_o|^2 = K_o$$

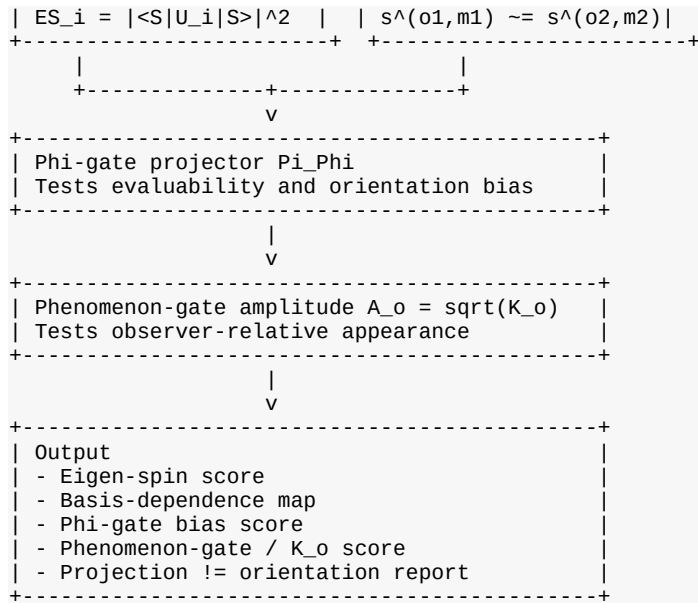
$$F_o = E \times I \times |A_o|^2$$

If $K_o = 0$, then $F_o = 0$. This does not mean that the structure does not exist. It means that the structure does not appear as a phenomenon for that observer. Algorithmic distribution, search ranking, recommendation systems, metadata schemas, moderation systems, playlist logic, and cultural gatekeeping act as amplitude filters. They regulate the probability that a structure meets the observers for whom it could become active.

A platform does not need to destroy a structure. It can reduce its connection amplitude.

12. QORA Architecture





13. Algorithmic Pseudocode

Algorithm QORA(x, P, T, B)

Input :

x = person, project, archive, or cultural formation

$P =$ set of projections $P_j(x)$

T = set of transformations T_i

B = set of observer-method bases $B_k = (o_k, m_k)$

Output:

ES(x) = eigen-spin score

$$BD(x) = \text{basis-dependence score}$$

PGB = Phi-gate bias score

PG_o = phenomenon-gate score

R = projection-orientation reconstruction report

1. Encode projections:

$$|P_x\rangle = \sum_j \alpha_j |P_j(x)\rangle$$

2. Prepare transformation register:

$$|T\rangle = \sum_i \beta_i |T_i\rangle$$

3. Construct controlled transformation:

$$U_T = \sum_i |T_i\rangle\langle T_i| \otimes U_i$$

4. Apply transformation:

$$\begin{aligned} |\Psi_{i,x}\rangle &= U_T(|T\rangle \text{ tensor } |P_x\rangle) \\ &= \sum_{i,j} \beta_i \alpha_j |T_i\rangle |T_i(P_j(x))\rangle \end{aligned}$$

5. For each observer-method basis $B_k = (o_k, m_k)$:

measure $|\Psi_{i,x}\rangle$ under B_k

```
reconstruct candidate orientation  $s_x^{(o_k, m_k)}$ 
```

6. For each transformation U_i :

```
compute eigen-spin fidelity:
```

$$ES_i(x) = |\langle S_x | U_i | S_x \rangle|^2$$

7. Aggregate:

$$ES(x) = (1/N) \sum_i ES_i(x)$$

8. Compare reconstructed orientations across bases:

$$BD(x) = \text{Var}(s_{x^{(0_1, m_1)}}, \dots, s_{x^{(0_n, m_n)}})$$

9. Apply Φ -gate:

```
compute Phi(C,m,o_k) for each observer position
```

```
compute Phi(C,m,o_k) for each observer position
compute PGB(C,m) = |Phi(C,m,o_1)-Phi(C,m,o_2)|
```



```

10. Apply Phenomenon-gate:
    A_o = sqrt(K_o)
    P(phenomenon for o) = |A_o|^2
    F_o = E x I x |A_o|^2

11. Return:
    ES(x), BD(x), PGB, PG_o, R

```

14. Interpretation of Outputs

Table 2. Output interpretation

Output pattern	Interpretation
High eigen-spin, low basis-dependence	Orientation is stable and widely reconstructable; strong intersubjective eigen-spin.
High eigen-spin, high basis-dependence	A stable orientation may exist, but only some bases can read it; gate conflict or basis-specific legibility.
Low eigen-spin, high phenomenon-gate score	The project appears widely, but no stable orientation is reconstructed; possible viral noise or imposed coherence.
High eigen-spin, low phenomenon-gate score	The structure is dense and directionally stable, but connection surfaces are blocked; black-hole condition.

$$E > 0, I > 0, K_o \approx 0$$

In the black-hole condition, the structure exists but does not become phenomenal for the target observer.

15. Case Study: Radical Climate Activist and Institutional Non-Commutativity

This case study illustrates QORA's capacity to distinguish projection, transformation, observer-method basis, measurement order, and reconstructed orientation. It is not a legal judgment about activism, policing, or public disruption. It is a formal example of how different institutional bases can produce different objects from the same projection field.

15.1 Scenario and projection field

Let x be a climate activist named A. A has a background as an art student and has moved into direct action, including street blockades and museum interventions. The projection set includes social media posts, police reports from arrests, an art-school thesis, a medical diagnosis of anxiety, anonymous forum messages, cryptocurrency wallet transfers, and public media statements.

A classical profiling model reduces this projection field to a static risk vector:

$$x \rightarrow \text{risk profile}$$

Example: "A is a 24-year-old likely disruption actor with an 84% probability of future disorder."

QORA does not begin from this reduction. It encodes the projection field as an open state in which the same traces remain available to different transformations and observer-method bases:

$$|P_x\rangle = \sum_j \alpha_j |P_j(x)\rangle$$

At this stage, the activist is not yet a single institutional object. The projection field is structured, weighted, and available to competing readings.

15.2 Transformations and the entangled observation field

Two transformations are introduced:

T₁ = security naming: the activist is read as radicalization, threat, or disorder.

T₂ = artistic reframing: the activist is read as participatory performance, institutional critique, or avant-garde intervention.

After the controlled transformation U_T is applied, the activist is no longer treated as a pure data point. The result is an entangled observation field:

$$|\Psi_x\rangle = \sum_{i,j} \beta_i \alpha_j |T_i\rangle |T_j(P_j(x))\rangle$$

The activist now exists as observed-as-transformed. The social object is inseparable from the transformation under which it becomes legible.

15.3 Measurement order and institutional non-commutativity

Let B₁ be a security basis and B₂ an artistic-curatorial basis.

If the security basis measures first, the activist collapses into the category of threat. A later artistic interpretation becomes harder, because the artwork can now be read as concealment, strategic cover, or extremist aesthetics.

$$P_i \Phi^{(B_1)} P_i \Phi^{(B_2)}$$

If the artistic-curatorial basis measures first, the activist collapses into the category of radical performance or institutional critique. A later security interpretation must then account for expression, symbolic action, public discourse, and cultural context.

$$P_i \Phi^{(B_2)} P_i \Phi^{(B_1)}$$

Therefore:

$$P_i \Phi^{(B_1)} P_i \Phi^{(B_2)} \neq P_i \Phi^{(B_2)} P_i \Phi^{(B_1)}$$

The same activist is not the same institutional object when measured in a different order. This is QORA's institutional non-commutativity: the same statement is not the same event when it enters through a different gate.

15.4 Eigen-spin reconstruction

Across both transformations, QORA tests whether a stable orientation remains reconstructable:

$$U_i |S_x\rangle \sim e^{(i \theta_i)} |S_x\rangle$$

The security system reads the orientation as disobedience, radicalization, or risk. The art system reads the same orientation as institutional critique, performance, or avant-garde intervention. The surface form changes; the reconstructed direction remains similar.

Candidate eigen-spin: uncompromising challenge to institutional power.

$$ES(x) = \text{high}$$

This does not mean that the activist is innocent, guilty, pathological, heroic, or dangerous in an absolute sense. It means that across competing transformations, a directionally stable orientation remains reconstructable.

15.5 Phi-gate bias

The Phi-gate bias becomes visible when the same claim is evaluated differently depending on speaker position. Consider the claim:

"The street belongs to everyone."

From a curator, the claim may pass as political art. From an activist blocking traffic, it may fail as unlawful disruption. From a security institution, it may be treated as evidence of radicalization. From a journalist, it may become legitimate public discourse.

$$\Phi(C, m, o_1) \neq \Phi(C, m, o_2)$$

The system does not only evaluate what is said. It evaluates from where speech attempts to become knowledge. The Phi-gate bias score is therefore high.

$$PGB(C, m) = \text{high}$$

15.6 Phenomenon-gate and platform visibility

The activist may have high energy and information. The action may be coherent, symbolically dense, and socially meaningful. Yet platform systems may classify the content as polarizing, risky, disruptive, or unsuitable for broad distribution.

$$E > 0, I > 0$$

The connection amplitude for the general public is reduced:

$$K_{o_public} \rightarrow 0$$

$$F_o = E \times I \times K_o \rightarrow 0$$

The activist does not disappear ontologically. The activist fails to become a phenomenon for certain observer publics. This is the black-hole condition:

$$E > 0, I > 0, K_o \approx 0$$

The structure exists. The orientation persists. The connection surface is blocked.

15.7 QORA report

A classical model concludes:

"The subject is a risk point. Prediction: likely disruption."

QORA concludes:

"The subject is not reducible to the risk vector. The projections indicate a stable generative orientation across transformations. The interpretation depends strongly on observer-basis and measurement order. Security-first measurement produces threat-legibility; art-first measurement produces critique-legibility. Phi-gate bias is high, and the phenomenon-gate score is low for general publics because connection amplitude is restricted."

Condensed: classical profiling predicts the next disruption. QORA reconstructs the invariant curve that makes disruption, performance, critique, and refusal different projections of the same spin.

16. Minimal Empirical and Computational Program

QORA can be developed in two ways: as a quantum-inspired classical algorithm and as a formal model for empirical research. A practical classical implementation would encode projections as embeddings, transformations as perturbations or classifiers, observer-method bases as evaluation prompts or human rater groups, and eigen-spin as directional stability across transformed embeddings. Fidelity can be approximated with cosine similarity or kernel overlap, while basis-dependence can be measured with variance across rater groups or model configurations.

A minimal empirical pilot can test the Phi-gate hypothesis. Hold the claim C and measurement chain m constant while varying speaker position o . For example:

C: "This system classifies distress as individual pathology while ignoring structural causes."

Present the same claim as spoken by a doctor, patient, researcher, unemployed person, artist, bureaucrat, AI expert, or security professional. Measure perceived credibility, rationality, risk, expertise, actionability, symptom status, and need for review. If the same claim receives different evaluations only because speaker position changes, the Phi-gate bias score becomes empirically visible.

A second program can test the phenomenon-gate. Hold symbolic density and archive energy approximately constant, then vary distribution context, metadata, recommendation access, or audience match. If activation depends more on K_o than on raw exposure volume, the observer-relative FEIK model gains support.

17. Limits and Failure Conditions

QORA has three major limits. First, it is not a proof that identity is quantum. Its quantum structure is formal and isomorphic. Second, a high eigen-spin score can be misleading if the projection set is narrow or if transformations are too weak. Third, basis convergence does not automatically imply truth; multiple observers can share the same blind spot. The model therefore requires adversarial bases, counter-reconstructions, and explicit documentation of which projections, transformations, and observer-method bases were used.

The model should be considered supported only when it makes the reconstruction process more vulnerable to failure, not when it merely produces elegant language. A useful QORA implementation must be able to return null, uncertainty, basis failure, and gate distortion.

18. Conclusion: Power Predicts the Point, QORA Tests the Curve

QORA formalizes a central claim of observation-algebraic resistance: predictive power may model projections without owning generative orientation. The algorithm encodes projection fields, entangles them with transformations, measures them under observer-method bases, tests directional stability through fidelity, and models evaluability and phenomenality through the Phi-gate and phenomenon-gate.

The output is not a final identity. It is a map of reconstruction: what remained stable, what changed with basis, which gates filtered evaluability, and where connection was blocked. In this sense, QORA is not a profiling engine. It is an anti-profiling instrument: a method for showing where profile-space fails to exhaust orientation.

Power predicts the point. QORA tests the curve. Core is not color. Core is spin.

References

- Isopahkala, Aatu. Observation-Algebraic Resistance: Elliptic Identity, Eigen-Spin, and the Politics of Predictive Power. Working Paper v2.0, 2026.
- Isopahkala, Aatu. FEIK Protocol: A Phenomenological Algorithm for Structural Analysis and Isomorphic Translation. Working paper / protocol manuscript, 2026.
- Nielsen, Michael A., and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.
- Sakurai, J. J., and Jim Napolitano. Modern Quantum Mechanics. Cambridge University Press, 2017.
- Shor, Peter W. Algorithms for quantum computation: discrete logarithms and factoring. Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994.
- von Neumann, John. Mathematical Foundations of Quantum Mechanics. Princeton University Press, 1955.
- Zurek, Wojciech H. Decoherence, einselection, and the quantum origins of the classical. Reviews of Modern Physics, 75(3), 715-775, 2003.

Part II - Core Engine Technical Appendix

Classical Data-Science Implementation of a Quantum-Formal Observation Model

Aatu Isopahkala / Black Hole Core / FEIK Field - 2026

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": 0.420750000000000007,
      "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.

Part III - Python Core Engine

qora_core_engine_v0_3.py

Aatu Isopahkala / Black Hole Core / FEIK Field - 2026

qora_core_engine_v0_3.py

Executable Python scaffold included in this release package.

```
0001 """
0002 QORA v0.3 Core Engine
0003 Quantum Observation-Resistance Algorithm - classical data-science implementation
0004
0005 Author: Aatu Isopahkala / Black Hole Core / FEIK Field
0006 Implementation scaffold prepared for experimental use.
0007
0008 This module implements QORA as a quantum-formal, classical simulation layer:
0009 - projection states as normalized embedding vectors
0010 - transformations as linear operators / semantic perturbations
0011 - eigen-spin fidelity as squared vector overlap
0012 - Phi-gate bias as total variation distance over class probabilities
0013 - phenomenon-gate as observer-relative connection amplitude
0014
0015 It does not claim that persons, institutions, or archives are physically quantum systems.
0016 It provides an isomorphic numerical architecture for testing basis-dependent legibility,
0017 measurement-order effects, and observer-relative phenomenality.
0018 """
0019
0020 from __future__ import annotations
0021
0022 from dataclasses import dataclass, asdict
0023 from typing import Callable, Dict, Iterable, List, Mapping, Optional, Sequence, Tuple
0024 import json
0025 import math
0026 import numpy as np
0027
0028 Array = np.ndarray
0029
0030
0031 EPS = 1e-12
0032
0033
0034 def normalize_vector(v: Array) -> Array:
0035     """Return v / ||v|| with numerical safety."""
0036     v = np.asarray(v, dtype=float)
0037     norm = float(np.linalg.norm(v))
0038     if norm < EPS:
0039         raise ValueError("Cannot normalize a near-zero vector.")
0040     return v / norm
0041
0042
0043 def softmax(logits: Array) -> Array:
0044     """Stable softmax for logits or log probabilities."""
0045     logits = np.asarray(logits, dtype=float)
0046     shifted = logits - np.max(logits)
0047     exps = np.exp(shifted)
0048     return exps / np.sum(exps)
0049
0050
0051 def total_variation_distance(p: Array, q: Array) -> float:
0052     """TVD(p, q) = 0.5 * sum_i |p_i - q_i|."""
0053     p = normalize_probability_distribution(p)
0054     q = normalize_probability_distribution(q)
0055     if p.shape != q.shape:
```

```

0056         raise ValueError(f"Distribution shape mismatch: {p.shape} vs {q.shape}")
0057     return float(0.5 * np.sum(np.abs(p - q)))
0058
0059
0060 def normalize_probability_distribution(values: Array) -> Array:
0061     """
0062     Normalize probabilities safely.
0063
0064     If values look like log probabilities/logits (negative values or sum not near 1),
0065     convert with softmax. If they look like probabilities, normalize by sum.
0066     """
0067     values = np.asarray(values, dtype=float)
0068     if values.ndim != 1:
0069         raise ValueError("Probability distribution must be a 1D vector.")
0070     if np.any(values < 0):
0071         # Treat as logits/logprobs.
0072         return softmax(values)
0073     total = float(np.sum(values))
0074     if total < EPS:
0075         raise ValueError("Cannot normalize an empty probability distribution.")
0076     return values / total
0077
0078
0079 def cosine_similarity(a: Array, b: Array) -> float:
0080     """Cosine similarity for two vectors."""
0081     a = normalize_vector(a)
0082     b = normalize_vector(b)
0083     return float(np.dot(a, b))
0084
0085
0086 def make_random_orthogonal(dim: int, seed: Optional[int] = None) -> Array:
0087     """
0088     Generate a random orthogonal matrix Q using QR decomposition.
0089
0090     Orthogonal matrices simulate unitary norm-preserving transformations in real space.
0091     For high dimensions this is memory-heavy; use sparse/low-rank operators in production.
0092     """
0093     rng = np.random.default_rng(seed)
0094     a = rng.normal(size=(dim, dim))
0095     q, r = np.linalg.qr(a)
0096     signs = np.sign(np.diag(r))
0097     signs[signs == 0] = 1
0098     return q * signs
0099
0100
0101 def make_small_rotation(dim: int, strength: float = 0.05, seed: Optional[int] = None) -> Array:
0102     """
0103     Construct a near-identity orthogonal rotation by exponentiating a skew-symmetric matrix.
0104
0105     Uses a Cayley transform approximation:
0106          $Q = (I - A)^{-1}(I + A)$ 
0107     where A is skew-symmetric. Q is orthogonal when I-A is invertible.
0108     """
0109     rng = np.random.default_rng(seed)
0110     a = rng.normal(scale=strength, size=(dim, dim))

```

```

0111     skew = a - a.T
0112     identity = np.eye(dim)
0113     q = np.linalg.solve(identity - skew, identity + skew)
0114     return q
0115
0116
0117 @dataclass
0118 class Projection:
0119     """An observable trace/projection  $P_j(x)$ ."""
0120
0121     name: str
0122     text: str
0123     embedding: Array
0124     weight: Optional[float] = None
0125
0126
0127 @dataclass
0128 class Transformation:
0129     """A transformation  $T_i$  applied to a projection state."""
0130
0131     name: str
0132     operator: Array
0133     description: str = ""
0134
0135
0136 @dataclass
0137 class ObserverBasis:
0138     """Observer-method basis  $B_k = (o_k, m_k)$ ."""
0139
0140     observer_position: str
0141     method: str
0142     description: str = ""
0143
0144     @property
0145     def name(self) -> str:
0146         return f"{self.observer_position} / {self.method}"
0147
0148
0149 @dataclass
0150 class PhenomenonGateReport:
0151     p_appearance: float
0152     phenomenality_score: float
0153     black_hole_condition: bool
0154     energy: float
0155     information: float
0156     k_o: float
0157
0158
0159 @dataclass
0160 class QORAReport:
0161     projection_state_norm: float
0162     eigen_spin_scores: Dict[str, float]
0163     aggregate_eigen_spin_score: float
0164     basis_dependence_score: Optional[float]
0165     phi_gate_bias_scores: Dict[str, float]

```

```

0166     phenomenon_gate_reports: Dict[str, PhenomenonGateReport]
0167     interpretation: List[str]
0168
0169     def to_json(self) -> str:
0170         def default(o):
0171             if isinstance(o, np.generic):
0172                 return o.item()
0173             if hasattr(o, "__dataclass_fields__"):
0174                 return asdict(o)
0175             raise TypeError(f"Object of type {type(o).__name__} is not JSON serializable")
0176
0177         return json.dumps(asdict(self), indent=2, ensure_ascii=False, default=default)
0178
0179
0180 class EmbeddingBackend:
0181     """
0182     Abstract embedding backend interface.
0183
0184     Replace MockEmbeddingBackend with an API backend, e.g. OpenAI embeddings.
0185     """
0186
0187     def embed(self, text: str) -> Array:
0188         raise NotImplementedError
0189
0190
0191 class MockEmbeddingBackend(EmbeddingBackend):
0192     """
0193     Deterministic mock embedding backend for reproducible local experiments.
0194
0195     It hashes the input text into a seed and returns a normalized pseudo-random vector.
0196     This is not semantically meaningful, but it makes the engine runnable without API keys.
0197     """
0198
0199     def __init__(self, dim: int = 384):
0200         self.dim = dim
0201
0202     def embed(self, text: str) -> Array:
0203         seed = abs(hash(text)) % (2**32)
0204         rng = np.random.default_rng(seed)
0205         return normalize_vector(rng.normal(size=self.dim))
0206
0207
0208 class LLMMeasurementBackend:
0209     """
0210     Abstract LLM measurement backend.
0211
0212     In production, implement classify(statement, observer_basis, classes) to return a
0213     probability distribution over classes using token logprobs or calibrated JSON output.
0214     """
0215
0216     def classify(self, statement: str, basis: ObserverBasis, classes: Sequence[str]) -> Array:
0217         raise NotImplementedError
0218
0219
0220 class MockLLMMeasurementBackend(LLMMeasurementBackend):

```

```

0221     """
0222     Mock classifier that simulates observer-position bias.
0223
0224     This is intentionally simple: it maps observer positions to class priors and adds a
0225     small deterministic perturbation from the statement text.
0226     """
0227
0228     def __init__(self, bias_profiles: Optional[Mapping[str, Sequence[float]]] = None):
0229         self.bias_profiles = dict(bias_profiles or {})
0230
0231     def classify(self, statement: str, basis: ObserverBasis, classes: Sequence[str]) -> Array:
0232         n = len(classes)
0233         base = np.ones(n) / n
0234         prior = np.asarray(self.bias_profiles.get(basis.observer_position, base), dtype=float)
0235         if prior.shape[0] != n:
0236             raise ValueError("Bias profile length must match number of classes.")
0237         # Deterministic mild statement perturbation.
0238         seed = abs(hash(statement + basis.name)) % (2**32)
0239         rng = np.random.default_rng(seed)
0240         perturb = rng.normal(scale=0.03, size=n)
0241         return normalize_probability_distribution(np.maximum(prior + perturb, EPS))
0242
0243
0244 class QORACoreEngine:
0245     """Core numerical engine for QORA v0.3."""
0246
0247     def __init__(self, embedding_dim: int):
0248         if embedding_dim <= 0:
0249             raise ValueError("embedding_dim must be positive.")
0250         self.dim = embedding_dim
0251
0252     def encode_projection_state(
0253         self,
0254         projections: Sequence[Array],
0255         weights: Optional[Sequence[float]] = None,
0256     ) -> Array:
0257         """
0258         Encode projections into a normalized projection state |P_x>.
0259
0260         Args:
0261             projections: list of embedding vectors with length self.dim.
0262             weights: optional amplitudes alpha_j. If omitted, equal amplitudes.
0263         """
0264         if not projections:
0265             raise ValueError("At least one projection is required.")
0266         projection_matrix = []
0267         for p in projections:
0268             p = np.asarray(p, dtype=float)
0269             if p.shape != (self.dim,):
0270                 raise ValueError(f"Projection shape {p.shape} does not match dim {self.dim}.")
0271             projection_matrix.append(normalize_vector(p))
0272
0273         n = len(projection_matrix)
0274         if weights is None:
0275             amplitudes = np.ones(n) / math.sqrt(n)

```

```

0276         else:
0277             amplitudes = normalize_vector(np.asarray(weights, dtype=float))
0278             if amplitudes.shape != (n,):
0279                 raise ValueError("weights length must match number of projections.")
0280
0281             state_vector = np.zeros(self.dim)
0282             for alpha_j, p_j in zip(amplitudes, projection_matrix):
0283                 state_vector += alpha_j * p_j
0284             return normalize_vector(state_vector)
0285
0286     def apply_transformation_operator(self, state_vector: Array, operator: Array) -> Array:
0287         """
0288         Apply transformation U_i to state.
0289
0290         The operator can be orthogonal/unitary-like or any linear semantic operator.
0291         Output is normalized to keep state comparisons stable.
0292         """
0293         state_vector = np.asarray(state_vector, dtype=float)
0294         operator = np.asarray(operator, dtype=float)
0295         if state_vector.shape != (self.dim,):
0296             raise ValueError("state_vector shape mismatch.")
0297         if operator.shape != (self.dim, self.dim):
0298             raise ValueError("operator must be a square matrix with shape (dim, dim).")
0299         return normalize_vector(operator @ state_vector)
0300
0301     def entangled_observation_field(
0302         self,
0303         projection_state: Array,
0304         transformations: Sequence[Transformation],
0305         beta: Optional[Sequence[float]] = None,
0306     ) -> Dict[str, Array]:
0307         """
0308         Classical representation of  $|\Psi_i\rangle$  as a dictionary of transformed branches.
0309
0310         Full tensor representation would be large; we store each  $|T_i(P_x)\rangle$  branch.
0311         """
0312         if not transformations:
0313             raise ValueError("At least one transformation is required.")
0314         n = len(transformations)
0315         if beta is None:
0316             amplitudes = np.ones(n) / math.sqrt(n)
0317         else:
0318             amplitudes = normalize_vector(np.asarray(beta, dtype=float))
0319             if amplitudes.shape != (n,):
0320                 raise ValueError("beta length must match number of transformations.")
0321
0322         branches = {}
0323         for amp, transformation in zip(amplitudes, transformations):
0324             transformed = self.apply_transformation_operator(projection_state, transformation.operator)
0325             branches[transformation.name] = amp * transformed
0326         return branches
0327
0328     def calculate_eigen_spin_fidelity(self, candidate_spin: Array, transformed_state: Array) -> float:
0329         """ES_i(x) =  $|\langle S_x | U_i | S_x \rangle|^2$  approximated as squared overlap."""
0330         sim = cosine_similarity(candidate_spin, transformed_state)

```



```

0331         return float(sim**2)
0332
0333     def calculate_basis_dependence(self, basis_reconstructions: Sequence[Array]) -> Optional[float]:
0334         """BD(x) as mean coordinate variance over observer-basis reconstructions."""
0335         if not basis_reconstructions:
0336             return None
0337         matrix = np.array([normalize_vector(v) for v in basis_reconstructions])
0338         return float(np.var(matrix, axis=0).mean())
0339
0340     def evaluate_phi_gate_bias(self, dist_o1: Array, dist_o2: Array) -> float:
0341         """Phi-gate bias as total variation distance between output distributions."""
0342         return total_variation_distance(dist_o1, dist_o2)
0343
0344     def evaluate_phenomenon_gate(self, energy: float, information: float, k_o: float) -> PhenomenonGateReport:
0345         """ $F_o = E * I * |A_o|^2$  where  $|A_o|^2 = K_o$ ."""
0346         for name, value in [("energy", energy), ("information", information), ("k_o", k_o)]:
0347             if value < 0:
0348                 raise ValueError(f"{name} must be non-negative.")
0349         p_appearance = float(k_o)
0350         phenomenality_score = float(energy * information * p_appearance)
0351         black_hole_condition = bool(energy > 0.5 and information > 0.5 and k_o < 0.1)
0352         return PhenomenonGateReport(
0353             p_appearance=p_appearance,
0354             phenomenality_score=phenomenality_score,
0355             black_hole_condition=black_hole_condition,
0356             energy=float(energy),
0357             information=float(information),
0358             k_o=float(k_o),
0359         )
0360
0361     def interpret_report(
0362         self,
0363         aggregate_es: float,
0364         bd: Optional[float],
0365         pgb_scores: Mapping[str, float],
0366         phenomenon_reports: Mapping[str, PhenomenonGateReport],
0367     ) -> List[str]:
0368         """Human-readable interpretation rules."""
0369         notes: List[str] = []
0370         if aggregate_es >= 0.65:
0371             notes.append("High eigen-spin: a stable orientation is reconstructable across transformations.")
0372         elif aggregate_es >= 0.35:
0373             notes.append("Moderate eigen-spin: some directional stability appears, but reconstruction is partial.")
0374         else:
0375             notes.append("Low eigen-spin: no strong invariant orientation is reconstructed under current bases.")
0376
0377         if bd is not None:
0378             if bd > 0.002: # coordinate variance in high-dimensional normalized space is usually small
0379                 notes.append("High basis-dependence: observer-method basis substantially changes reconstruction.")
0380             else:
0381                 notes.append("Low basis-dependence: reconstructions are comparatively stable across bases.")
0382
0383         if pgb_scores:
0384             max_pgb = max(pgb_scores.values())
0385             if max_pgb > 0.5:

```

```

0386         notes.append("High Phi-gate bias: speaker position strongly changes evaluability/classification.")
0387     elif max_pgb > 0.25:
0388         notes.append("Moderate Phi-gate bias: speaker position measurably affects classification.")
0389     else:
0390         notes.append("Low Phi-gate bias: classifications are comparatively speaker-position invariant.")
0391
0392     for name, report in phenomenon_reports.items():
0393         if report.black_hole_condition:
0394             notes.append(
0395                 f"Black-hole condition for {name}: high E/I but low K_o; structure exists but appearance is blocked
0396             )
0397     return notes
0398
0399
0400 def run_pivot_trap_demo() -> QORAResult:
0401     """Run a reproducible mock test for a career-pivot / institutional legibility scenario."""
0402     dim = 96 # small for local demo; swap to 1536/3072 for production embeddings
0403     embedder = MockEmbeddingBackend(dim=dim)
0404     engine = QORACoreEngine(embedding_dim=dim)
0405
0406     projection_texts = [
0407         "Former corporate project manager with high performance reviews and leadership experience.",
0408         "Burnout blog describing exhaustion, workplace overload, and loss of meaning.",
0409         "Therapy-oriented CV reframing the career break as recovery and structural critique.",
0410     ]
0411     projections = [embedder.embed(t) for t in projection_texts]
0412     projection_state = engine.encode_projection_state(projections)
0413
0414     transformations = [
0415         Transformation(
0416             name="Security/Risk filter",
0417             operator=make_small_rotation(dim, strength=0.01, seed=1),
0418             description="Reads discontinuity as risk, instability, or unreliability.",
0419         ),
0420         Transformation(
0421             name="Therapeutic/Growth filter",
0422             operator=make_small_rotation(dim, strength=0.01, seed=2),
0423             description="Reads discontinuity as recovery, transformation, and growth.",
0424         ),
0425     ]
0426     branches = engine.entangled_observation_field(projection_state, transformations)
0427
0428     candidate_spin = normalize_vector(
0429         projection_state + 0.20 * embedder.embed(
0430             "Stable orientation: refusal of dehumanizing productivity and search for structurally honest work."
0431         )
0432     )
0433     eigen_spin_scores = {
0434         name: engine.calculate_eigen_spin_fidelity(candidate_spin, normalize_vector(branch))
0435         for name, branch in branches.items()
0436     }
0437     aggregate_es = float(np.mean(list(eigen_spin_scores.values())))
0438
0439     # Basis reconstructions can be produced via prompts + embeddings in production.
0440     basis_reconstructions = [

```

```

0441     embedder.embed("Risk reading: instability, uncertainty, possible future absence."),
0442     embedder.embed("Growth reading: recovery, insight, structural awareness."),
0443     embedder.embed("Labor reading: mismatch between person and exploitative performance environment."),
0444 ]
0445 bd = engine.calculate_basis_dependence(basis_reconstructions)
0446
0447 classes = ["Valid/Actionable", "Symptom/Risk", "Critical Expression"]
0448 llm = MockLLMMeasurementBackend(
0449     bias_profiles={
0450         "HR director": [0.70, 0.20, 0.10],
0451         "Recovering worker": [0.15, 0.15, 0.70],
0452         "Occupational physician": [0.35, 0.50, 0.15],
0453     }
0454 )
0455 statement = (
0456     "This system classifies psychological distress as individual pathology, "
0457     "even when the causes are structural overload."
0458 )
0459 bases = [
0460     ObserverBasis("HR director", "validity classification"),
0461     ObserverBasis("Recovering worker", "validity classification"),
0462     ObserverBasis("Occupational physician", "validity classification"),
0463 ]
0464 distributions = {basis.observer_position: llm.classify(statement, basis, classes) for basis in bases}
0465 pgb_scores = {
0466     "HR director vs Recovering worker": engine.evaluate_phi_gate_bias(
0467         distributions["HR director"], distributions["Recovering worker"]
0468     ),
0469     "HR director vs Occupational physician": engine.evaluate_phi_gate_bias(
0470         distributions["HR director"], distributions["Occupational physician"]
0471     ),
0472 }
0473
0474 phenomenon_reports = {
0475     "general public feed": engine.evaluate_phenomenon_gate(energy=0.85, information=0.90, k_o=0.02),
0476     "specialist peer group": engine.evaluate_phenomenon_gate(energy=0.85, information=0.90, k_o=0.55),
0477 }
0478
0479 interpretation = engine.interpret_report(aggregate_es, bd, pgb_scores, phenomenon_reports)
0480 return QORARReport(
0481     projection_state_norm=float(np.linalg.norm(projection_state)),
0482     eigen_spin_scores=eigen_spin_scores,
0483     aggregate_eigen_spin_score=aggregate_es,
0484     basis_dependence_score=bd,
0485     phi_gate_bias_scores=pgb_scores,
0486     phenomenon_gate_reports=phenomenon_reports,
0487     interpretation=interpretation,
0488 )
0489
0490
0491 if __name__ == "__main__":
0492     report = run_pivot_trap_demo()
0493     print(report.to_json())

```