VAAS-X SDK · SCIENTIFIC COMPUTING

Memory Applied to Physics, Not Just Telemetry

Three independent systems — a quantum error-correction decoder, a lattice field theory Monte Carlo simulation, and a transverse-field Ising model — tested against the vaas-x SDK, with the honest result reported for all three, including where it fell short.

1	3	2	0
RECURRING FAULT LEARNED FROM ONE EXAMPLE	INDEPENDENT PHYSICS SYSTEMS TESTED	INDEPENDENT GROUND TRUTHS, RECOMPUTED PER QUERY	GPU REQUIRED

VAAS-X Ltd (SC873624) · August 2026 · vaasx.com

Why test this at all

Every other vaas-x whitepaper validates the same underlying claim — zero-config channel classification and outcome-grounded retrieval — against telemetry: industrial sensors, wearable IMUs, agent session logs. That's the market. But it left an open question: does retrieval-based memory transfer to something with no sensor stream at all — a domain where the "state" is a scientific simulation's own parameters, and there's no threshold-alarm status quo to beat? We ran three independent physics/computing systems through the SDK to find out, held every claim to the same bar as the rest: a real ground truth computed independently of vaas-x for every single test point, and the honest result reported whether or not it flattered us.

What Bootstrap actually does

The same `vaas-x` ingestion and retrieval path used throughout — ingest an episode, query it back by similarity, report the outcome — applied unchanged to simulation output instead of sensor readings.

```
from vaasx import Bootstrap

brain = Bootstrap(api_key="YOUR_API_KEY", device_id="physics_qec")

brain.ingest([{
    "state": {"text": "Syndrome: 1000000100010001010010011011000000..."},
    "action": {"text": "correction=0000000000111111100000000..."},
    "outcome": {"text": "learned recurring burst fault pattern", "success": True},
}])

hits = brain.query("Syndrome: 1000000100010001010010011011000000...", k=1)
print(hits[0]["score"], hits[0]["text"])
```

Validated: learning one recurring fault, cleanly

A 64-bit random linear code (n=64, k=32) mathematically cannot correct an arbitrary weight-6 error — it exceeds the code's guaranteed correction distance. But real hardware often produces the *same* high-weight fault repeatedly: a stuck sensor line, a recurring burst on the same physical bits. We queried a specific weight-6 burst syndrome before vaas-x had ever seen it, ingested exactly one episode describing that syndrome and its correction, then queried the identical syndrome again.

> **RESULT**

> Before ingestion: no match, exactly as expected — the code genuinely cannot solve this algebraically. After one ingested episode: the correct correction was recalled and independently verified against the code's own `check_correction()` — not just "a hit came back", the recalled bits were checked to actually fix the error.

This is a recall test, not a generalization test, and we're not presenting it as one: it demonstrates memory remembers a specific pattern it has been shown once, not that it predicts corrections for syndromes it has never seen.

What we found on the two continuous-parameter systems

The other two systems ask a harder question: given a physics simulation's continuous input parameter (a coupling constant, a field ratio), can a text-embedding query predict the correct phase for a value memory has never been shown — not recall, real interpolation? We tested this against two independent ground truths: a real Metropolis Monte Carlo lattice simulation (13 kappa values ingested, seeded and averaged across 3 independent MC chains per point to smooth out the genuine sampling noise a single chain has near a continuous transition) for a phi4 scalar field theory, and real exact diagonalization (20 h/J ratios ingested) for a transverse-field Ising model's quantum phase transition. Both ground truths were recomputed fresh for every held-out query — nothing was taken on faith.

> **RESULT, REPORTED HONESTLY**
>
> Lattice phi4: 5 of 8 held-out kappa values correctly classified. Transverse-field Ising: 4 of 10 held-out h/J ratios correctly classified. In both systems, every miss was the same shape — memory defaulted toward whichever phase had more representation in what was ingested (7 of 13 phi4 episodes were "ordered"; 7 of 20 TFIM episodes were "critical"), rather than discriminating finely near the actual phase boundary. We then tried a legitimate fix before publishing this: k=7 retrieval with a score-weighted majority vote instead of trusting a single top-1 hit — standard practice, not a way of engineering the answer. It changed nothing. The TFIM run predicted "critical" for **every single query**, both before and after the fix, identical down to the query. That rules out top-1 noise as the explanation: the retrieved neighbourhood itself is dominated by the majority class regardless of the query value.

Our read: this is a real, useful limitation, not a bug. Short, numerically-dense query strings like `"h/J=0.350"` differ from a training example by only a few characters in an otherwise near-identical string — a general-purpose sentence embedding model has little reason to weight that span heavily, and it shows. Retrieval does coarse region classification correctly (every miss landed on the wrong side of a real, close decision boundary, not on a wildly wrong answer), but it is not a substitute for a numerical model when the question is "exactly where is the line." Categorical or recurring-pattern memory (QEC's syndrome recall, or industrial/wearable channel classification, where the signal dominates the description) is where this SDK is validated to work well. Fine interpolation across a

continuous parameter from a bare numeric label in the query text is not, at least not without more deliberate encoding work than a raw f-string.

Scope, honestly

What this validates: the ingest/query/outcome path works unmodified against non-telemetry, purely computational domains, and memory reliably learns and recalls a specific recurring pattern once shown it — the QEC result. What it does **not** show: that vaas-x is a substitute for a real numerical or ML model when the task is fine-grained interpolation across a continuous parameter space from short text alone. Anyone evaluating vaas-x for a scientific-computing use case should match the shape of their problem to what's proven here — recurring/categorical pattern memory, not curve-fitting.

Deployment model

Identical to every other vertical: ingestion and retrieval run against the hosted resolver behind a single API key, with no local GPU or vector database to operate. The same ingest/query/outcome calls shown above are what a research or engineering team would call directly from a simulation pipeline, a CI job, or a notebook — nothing simulation-specific was added to the SDK to make this work.

Get started

Free tier: instant API key, no credit card — `pip install vaas-x`, connect a source, query your first episode in under ten minutes.

Full guide: vaasx.com/docs/getting-started.html | Enterprise: hello@vaasx.com