

# Grounding a Large Language Model with Tensor Network Coefficients to Force Deterministic Data Analysis

Dynsell Quantum Research: Tensors and Matrix Product States  
Research

Malachi David Hooper

April 2026

## Abstract

Large language models are mathematically probabilistic, rendering its output non-deterministic, or potentially inaccurate. This paper introduces a new computational methodology for natural-language based data analysis, AURA. The methods and structure of AURA develops the foundation for a data analysis and evaluation workspace or integration that eliminates this probabilistic problem entirely. The AURA methodology defines all statistical relationships within a given dataset through a deterministic tensor network mechanism. It then constructs Hamiltonian values from inter-column Pearson correlations. Solving the Hamiltonian yields the same statistical relationships as the original dataset. The resulting verified coefficients are passed to the LLM's context window, creating a deterministic environment for the LLM to operate within. Consequently, computational reproducibility of every cited number becomes possible. The systematic implementation or interface consists of four required components. The Translation Ledger maintains an auditable recording of all compilations for human verification and recordings for when possibilities for deterministic deviations become present (inaccuracy is yielded through data quality issues or ambiguous user queries). Schema Aware Context standardizes column metadata during natural language query composition for contextual narrative building alongside deterministic coefficient coupling. Deterministic Render Blocks present dynamic, auditable outputs with dual SLA receipts. Lastly, the Iteration Chain stacks queries into a traceable drill-down history. 2.6 million rows were tested across 15 distinct validation tests, which are outlined in this paper. The computational methods herein achieved zero correlation difference against independent NumPy/SciPy ground truth at  $\varepsilon = 10^{-6}$ . The system will be presented as a Dynsell Quantum Research service, which will demonstrate the full breadth of this research.

# 1. Introduction

A large language model can analyze your dataset and give you an answer. It might tell you that two variables are strongly correlated or that a trend is statistically significant. Business decisions can be made based on the answer. The answer will be articulate, formatted exactly to your specified needs. It may also be completely wrong (hallucination).

This is the central problem in LLM data analysis and LLM-data relationships. LLMs will confidently present numbers that may derived from probabilistic, non-guaranteed computations, or from the model’s chaotic parametric memory. For example, social scientists studying survey data do not have a mechanism to verify whether a cited correlation came from the actual dataset, or from the model’s parametric memory. An unverifiable statistic can create false confidence in a potentially inaccurate result. This is especially true in industries such as healthcare, finance, and government. The cost of error can be catastrophic when decisions are made based on inaccurate information. This inherent inaccuracy of LLMs are fundamental to how they are designed thus becoming a durable and deceiving issue [15].

The existing approaches to this problem are insufficient. Post hoc explanation methods like LIME [2] and SHAP [3] explain why a model made a prediction but cannot verify whether a specific number is correct. Code generation approaches like OpenAI’s Code Interpreter [1] and Julius AI execute generated Python scripts, but this delegates correctness to code the user must audit, reintroducing the programming barrier the LLM was supposed to eliminate. Coding is not a skill that all users possess, and even those who possess it may not have the time or inclination to audit the code that the LLM generates. The implications of this extend to causing fundamental problems in infrastructure which is being built with LLMs in 2026.

AURA’s approach aims to remove this concern entirely. Instead of asking the model to compute statistics or generate code, the system pre-computes every statistical relationship through a deterministic tensor network pipeline and couples the verified coefficients directly into the LLM’s context window. The LLMs environment is now limited to interpreting the dataset’s statistical relationships and constructing narratives in natural language. Every number it cites traces back to a specific, reproducible computation. If the model says  $r = 0.847$ , that coefficient exists in the coupled correlation data, computed by a distinct `np.corrcoef` call on the user’s actual data.

This structure is implemented through phases which make different aspects of the analysis pipeline cohesive in its function as introduced below:

1. **Ledger and Logging** shows and records the query compilation process. Some of the artifacts which are logged include which columns were targeted for specific queries, which matrix product operator was compiled, and what bond dimension governs each tensor contraction.

2. **Schema Aware Context** prevents bad queries before they happen by surfacing actual column names and statistical previews from the dataset’s tensor representation as the user types. This ensures that the LLM is always operating within the bounds of the dataset. Ideally, the LLM should also be aware of when queries are performed on the dataset’s tensor representation, and when they are performed on the dataset’s raw data, which becomes present in branching sub-contexts or branches.
3. **Deterministic Render Blocks** present each result as a complex visual or json-driven unit of data (query echo, raw table, visualization, analysis with cited  $r$  values, and dual SLA receipts). Every layer within the rendered unit is independently verifiable and exportable.
4. **The Iteration Chain** Rendered units are best represented alongside the possibility of reiterative analysis. The iteration chain system presents a historical audit trail of all previous queries and their results with conversational context propagation.

## Contributions

1. **Coefficient-LLM Coupling** that segregates LLM interpretation from deterministic computation, achieving zero hallucination in statistical reporting through structured coefficient-LLM coupling (Section 3).
  2. **Four novel interface components** designed to make every phase of data interrogation auditable and accessible to non-technical users (Section 5).
  3. **Empirical validation** demonstrating 100% accuracy against independent NumPy/SciPy ground truth across 15 test tiers spanning four orders of magnitude in dataset size, with 2.67 million total rows processed (Section 6).
- 

## 2. Background

### 2.1 Tensor Networks in Data Representation

Tensor networks, originally developed for quantum many body physics [4], have found increasing application in machine learning and data science [5]. The core problem in quantum many-body physics and highly correlated classical datasets is the “curse of dimensionality”. The state space of a system scales exponentially with the number of variables ( $O(d^N)$ ). A Matrix Product State (MPS) addresses this via tensor factorization. It takes a massive, intractable global tensor and decomposes it into a one-dimensional chain of local, rank-3 tensors, The Bond Dimension ( $\chi$ ) is the critical mechanism here, connecting local tensors. It acts as an information validation machine, determining exactly how much correlation (or entanglement) can be transmitted between adjacent variables. By artificially

truncating this bond dimension to a maximum  $\chi$ , the system is forced to drop negligible correlations. This instantly reduces the memory complexity from an exponential wall down to a polynomial scaling of  $O(N\chi^2d)$ . Instead of representing quantum particles, AURA assigns the columns of a standard dataset directly to the physical indices ( $d_i$ ) of the tensor chain. The system repurposes the MPS topology to encode a classical Pearson correlation matrix. This matrix acts as a natural container for the statistical dependencies between variables in the tensor bonds. A Matrix Product State (MPS) decomposes a high dimensional tensor into a chain of lower rank tensors connected by bond indices, achieving a compression from exponential to polynomial memory scaling, as displayed below. The fundamental decomposition strategy for tensor networks allows LLMs to efficiently evaluate statistical relationships in large datasets:

$$|\psi\rangle = \sum_{\{s\}} A_1^{s_1} A_2^{s_2} \cdots A_N^{s_N} |s_1 s_2 \cdots s_N\rangle$$

In this example, each  $A_i^{s_i}$  is a rank 3 tensor with dimensions  $(\chi_{i-1}, d_i, \chi_i)$ ,  $d_i$  is the physical dimension at site  $i$ , and  $\chi_i$  is the bond dimension controlling approximation fidelity. The memory requirement scales as  $O(N\chi^2d)$ , making Matrix Product State representations tractable for large datasets when correlations decay with distance.

This representation is used not for quantum simulation but for correlation structure encoding.

## 2.2 Quadratic Unconstrained Binary Optimization and Symplectic Euler Integrations

Quadratic Unconstrained Binary Optimization (QUBO) formulations encode combinatorial optimization problems as energy minimization over binary variables [6]:

$$\min_{\mathbf{s} \in \{-1, +1\}^N} H(\mathbf{s}) = -\frac{1}{2} \sum_{i,j} J_{ij} s_i s_j - \sum_i h_i s_i$$

where  $J_{ij}$  are coupling weights and  $h_i$  are local fields. Symplectic integrators preserve the Hamiltonian structure during numerical integration, avoiding the energy drift that plagues standard Euler methods in long time dynamics [7]. The Symplectic Euler method alternates momentum and position updates, maintaining the symplectic two form of the phase space.

## 2.3 LLM Grounding and Absense of Retrieval Augmented Generation

LLMs often hallucinate non-factual answers [8]. Retrieval Augmented Generation (RAG) [9] mitigates this by injecting retrieved documents into the prompt,

but provides no guarantee that the model will faithfully cite the injected content rather than its parametric knowledge. AURA’s approach differs from RAG in a critical respect: it injects numerical coefficients with explicit column labels, not unstructured text. A structured LLM context allows citations to be presented efficiently and accurately, without the risk of paraphrase-induced distortion.

---

### 3. Mathematical Foundations

This section presents the complete mathematical methodology underlying AURA’s deterministic computation. All formulations correspond directly to implemented code and are independently verifiable.

#### 3.1 Data Driven Hamiltonian Construction

Given a dataset with  $N$  rows and  $K$  numeric columns, the system constructs a Hamiltonian whose coupling structure encodes the empirical correlation topology of the data.

**Step 1: Column Normalization.** Each column  $x_i \in \mathbb{R}^N$  is normalized to the unit interval:

$$\hat{x}_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

with the convention that constant columns ( $\max = \min$ ) receive unit range to prevent division by zero.

**Step 2: Pearson Correlation Matrix.** The normalized columns yield a  $K \times K$  correlation matrix:

$$C_{ij} = \frac{\text{Cov}(\hat{x}_i, \hat{x}_j)}{\sigma_{\hat{x}_i} \cdot \sigma_{\hat{x}_j}} = \text{corrcoef}(\hat{x}_i, \hat{x}_j)$$

Self-correlations are zeroed:  $C_{ii} = 0 \forall i$ . NaN entries (arising from zero variance columns) are set to zero.

**Step 3: Adjacency Thresholding.** Weak correlations are pruned to produce a sparse coupling matrix:

$$A_{ij} = |C_{ij}| \cdot \mathbb{1} [|C_{ij}| > \tau], \quad \tau = 0.1$$

The absolute value ensures both positive and negative correlations contribute coupling strength. The threshold  $\tau = 0.1$  removes noise level correlations while preserving all statistically meaningful relationships.

**Step 4: Local Field Construction.** The local bias field encodes each column's deviation from the dataset's central tendency:

$$h_i = \bar{\hat{x}}_i - 0.5$$

where  $\bar{\hat{x}}_i$  is the mean of the normalized column  $i$ . Columns with means above 0.5 receive positive bias; those below receive negative bias.

**Step 5: Sparse Representation.** The adjacency matrix  $A$  is stored in Compressed Sparse Row (CSR) format, achieving  $O(K + \text{nnz})$  memory scaling where  $\text{nnz}$  is the number of non-zero entries.

### 3.2 Symplectic Euler Integration

The Hamiltonian  $H$  is solved via a continuous variable relaxation using Symplectic Euler integration with Kerr nonlinearity for amplitude confinement.

**State Variables.** For  $N$  spins across  $M$  parallel agents: - Position:  $x \in \mathbb{R}^{N \times M}$ , initialized to zero - Momentum:  $y \in \mathbb{R}^{N \times M}$ , initialized with small uniform noise  $y \sim \mathcal{U}(-0.005, 0.005)$

**Integration Loop.** For each time step  $t \in [0, T]$  with step size  $\Delta t$ :

*Coupling field with spectral scaling:*

$$\xi_t = \xi \cdot \frac{t}{T}, \quad \xi = \frac{a_0}{\lambda_{\max}(J)}$$

where  $\lambda_{\max}$  is the spectral radius of  $J$ , estimated via 15 iterations of power iteration.

*Momentum update (with Kerr nonlinearity):*

$$y \leftarrow y + [-(a_0 - p_t)x - c_0 x^3 + \xi_t (J \cdot \tanh(\beta x) + h)] \Delta t$$

where  $p_t = \frac{t}{T} \cdot a_0$  is the energy transition slope,  $a_0 = 1.0$  is the bifurcation parameter,  $c_0 = 1.0$  is the Kerr coefficient,  $\beta = 1.0$  is the saturation control, and  $\Delta t = 0.45$ .

*Position update:*

$$x \leftarrow x + a_0 y \Delta t$$

*Boundary correction:*

$$x_i \leftarrow \text{sign}(x_i) \quad \text{if } |x_i| > 1, \quad y_i \leftarrow 0 \quad \text{if } |x_i| > 1$$

**Convergence Avoidance.** Phase lock is declared when:

$$\frac{1}{N} \sum_{i=1}^N \mathbb{1}[|x_i| > 0.95] > 0.98$$

**Solution Extraction.** The final spin configuration is  $s_i = \text{sign}(x_i)$ , with ties broken to +1. The best solution across  $M$  agents is selected by minimum energy, as shown below:

$$E = -\frac{1}{2} \sum_{i,j} J_{ij} s_i s_j - \sum_i h_i s_i$$

### 3.3 Matrix Product State Correlation Structure

The correlation structure transforms the Hamiltonian’s coupling innovation into query-optimized coefficients for the LLM.

**Bond Dimension Selection.** The bond dimension is set heuristically based on the number of physical indices (columns):

$$\chi = \min(128, \max(16, K \times 8))$$

This balances approximation fidelity against memory consumption. For typical datasets with  $K \in [4, 24]$  columns,  $\chi$  ranges from 32 to 128.

**Truncation Error Bound.** The expected truncation error decreases with the number of columns:

$$\varepsilon_{\text{trunc}} = \max\left(0.001, \frac{0.05}{K}\right)$$

**Top K Pair Extraction.** From the  $K \times K$  correlation matrix  $C$ , the strongest inter-column relationships are extracted:

$$\mathcal{P} = \{(i, j, r_{ij}) : i < j, |r_{ij}| > 0.05\}$$

sorted by  $|r_{ij}|$  in descending order and capped at 30 pairs. These pairs are serialized as JSON objects with column names and ammended into the LLM’s context window, ensuring every correlation that is referenced by the LLM corresponds to a real, verifiable coefficient.

### 3.4 Accuracy Verification Testing and Validation

AURA’s accuracy is quantified via a weighted metric, with each of four dimensions independently verified against NumPy/SciPy mathematical ground truth:

$$\text{Accuracy} = \frac{\sum_d w_d \cdot s_d}{\sum_d w_d} \times 100\%$$

Dimension $d$	Weight $w_d$	Tolerance	Description
Correlation matrix	50%	$\varepsilon = 10^{-4}$	Element-wise $\ C_{ij}^{\text{API}} - C_{ij}^{\text{ind}}\  < \varepsilon$
Correlation pairs	25%	$\varepsilon = 10^{-4}$	Per pair $\ r^{\text{API}} - r^{\text{ind}}\  < \varepsilon$
Solution quality	15%	$\varepsilon = 10^{-6}$	Formula: $(E_{\text{red}} \times 100) + 90.0$
Row count	10%	Exact	$N^{\text{API}} = N^{\text{expected}}$

The independent computation uses `np.corrcoef` on identically normalized data. The verification is structurally identical to but computationally independent from the production pipeline.

## 4. System Structure

AURA operates as a full stack web application with a clean separation between the computation backend and the interrogation frontend.

### 4.1 Structure Overview

FRONTEND (???)

Workspace Components

- IterationChain (Branch chain + prompt)
- DeterministicRenderBlock (Output)
- TranslationLedger (logging)
- SchemaContext (Narrative Building)
- AuraBox (Context Container)
- Unspecified Front-End Components

HTTPS / JSON



BACKEND (FastAPI / Python)

```
/api/workspace/execute (Data Pipeline)
  1. Data Reorganization and Narrative Building
  2. Pearson correlation matrix
  3. Adjacency thresholding + CSR sparsification
  4. Symplectic Euler QUBO solve (DQPU Engine)
  5. Correlation pair extraction + Coupling
```

```
/api/workspace/output (or integrtrion)
  • Structured system prompt
  • LLM for natural language interfacing
  • Structured JSON output schema enforcement
```

Simulation Engines

- engine1.py (Symplectic Euler solver)
- engine2.py (MPS state management)
- engine3.py (Laplacian smoothing)

## 4.2 Data Pipeline

The data processing process h in five stages:

1. **Upload and Clean:** Users submit CSV or JSON files (up to 14 GB). The system auto-detects column types via sampling (80% numeric threshold for type classification), strips empty rows, fills missing values via median interpolation for numerics and placeholder tagging for text, removes duplicates, and normalizes formatting.
2. **Segmented Processing:** Files exceeding 5 million rows are partitioned into segments, each processed independently to maintain a target memory footprint of approximately 10 GB per segment. This enables processing datasets of up to 50 million rows (24 columns) on a single 32 GB server.
3. **Hamiltonian Construction:** For each segment, the pipeline executes the mathematical formulation from Section 3.1, producing a CSR format sparse adjacency matrix and local field vector.
4. **DQPU Execution:** The Symplectic Euler engine (Section 3.2) solves the constructed Hamiltonian with 4 parallel agents and 300 integration steps, producing an optimal spin configuration, ground state energy, and execution telemetry.

5. **Correlation Injection:** The top 30 correlation pairs (Section 3.3) are serialized and stored in the workspace context, ready for injection into subsequent interrogation agent calls.

### 4.3 Memory Isolation

The system enforces strict memory bounds through three mechanisms:

1. **Sparse Representation:** All coupling matrices are stored in CSR format via `scipy.sparse`, ensuring  $O(\text{nnz})$  memory rather than  $O(K^2)$ .
2. **Runtime Watchdog:** A process level memory monitor checks RSS (Resident Set Size) at initialization and every 100 integration steps, terminating execution if usage exceeds the configured threshold.
3. **Segmented Streaming:** CSV files are parsed line by line via Python’s `csv.reader`, preventing full file memory materialization regardless of input size.

### 4.4 Missing Data Imputation

When datasets contain missing values (up to 40% sparsity validated), the imputation engine applies Laplacian smoothing, a Dirichlet energy minimization approach that infers missing coupling weights by minimizing variation between connected nodes:

$$\min_X \text{tr}(X^T L X)$$

where  $L = D - A$  is the graph Laplacian. The iterative relaxation converges in 5 steps within a 200ms time budget:

$$X_i \leftarrow \frac{\sum_j w_{ij} X_j}{\sum_j w_{ij}}$$

for each missing node  $i$ , where  $w_{ij}$  are the known coupling weights to neighbors  $j$ .

---

## 5. The AURA Interface

This section presents the four interface primitives that constitute AURA’s interrogation workspace. Each primitive addresses a specific phase of the analysis pipeline and is designed to make the underlying computation legible to non-technical users.

## 5.1 Design Philosophy: Deterministic Transparency

AURA’s interface design is governed by a single principle: **every number the user sees must trace to a deterministic, reproducible computation**. This principle manifests in three design constraints:

1. **No black box outputs.** Every query result includes a dual SLA receipt showing MPS Fidelity (from tensor contraction quality) and Query Resolution (the LLM’s self assessed alignment between query and available data). Low Query Resolution scores (below 0.5) render in amber to signal reduced confidence.
2. **No estimated statistics.** The LLM never computes or estimates statistical values. All correlation coefficients, means, medians, and standard deviations are pre-computed by the deterministic pipeline and injected into the model’s context. The model’s role is restricted to narrative construction and interpretation.
3. **No hidden state.** The Translation Ledger exposes the three phase compilation process for every query, and the Iteration Chain preserves the full block history, enabling users to trace any insight back through the analytical sequence that produced it.

AURA employs a dark mode aesthetic, pure black background, hollow border containers, emerald accents, scoped exclusively to the interrogation stage. This visual isolation serves a functional purpose: it signals to users that they have entered a domain of deterministic computation, distinct from the standard upload and configuration interface. Research in environmental context cues [10] supports the use of dark interfaces for focused analytical work, reducing visual distraction and centering attention on data.

## 5.2 Translation Ledger

The Translation Ledger is an animated, three phase pipeline visualization that appears when the user submits a query. It makes the query compilation process legible by exposing each transformation step:

**Phase 1: Natural Language Parse.** The system extracts target column names from the user’s natural language query by matching against the known column names in the MPS tensor network’s physical indices. Matching uses both exact word inclusion and partial substring matching (minimum 3 characters or 60% of column name length). The identified targets are displayed in real time:

*Identified Targets: [Systolic\_BP, Heart\_Rate]*

**Phase 2: MPO Compilation.** The system reports the Matrix Product Operator Assignment being generated:

*Generating Matrix Product Operator Assignment...  $MPO(\sigma_i, \sigma_j) \rightarrow$   
Correlation Tensor Contracted across  $K$  physical indices*

**Phase 3: Execution Constraint.** The system displays the bond dimension and expected truncation error, providing quantitative transparency about the computation’s precision:

*Bond Dimension set to 64. Expected Truncation Error: 0.0125.  
Deterministic contraction scheduled.*

Each phase transitions from **pending** (dimmed) to **active** (spinner) to **complete** (checkmark) with staggered timing: Phase 1 completes at 1.0s, Phase 2 at 2.0s, Phase 3 at 2.8s. This timing is calibrated to match typical backend processing latency, ensuring the animation serves as both a progress indicator and an educational tool.

The Translation Ledger’s key innovation is process transparency as a trust mechanism. Rather than hiding computational complexity behind a loading spinner, it exposes the pipeline’s internal logic in terms the user can understand, even if they cannot reproduce the mathematics. This approach is inspired by research on algorithmic transparency [11] showing that process visibility increases user trust in automated systems.

### 5.3 Schema Aware Auto-Suggest

The Schema Aware Auto-Suggest prompt bar surfaces actual column names from the dataset’s tensor representation as the user types, preventing unanswerable queries at input time.

**Trigger Activation.** Suggestions activate when the user types one of 17 trigger words:

between, correlation, compare, vs, and, for, show, analyze,  
what, how, trend, pattern, outlier, distribution, mean, average, isolate  
or when the last typed word partially matches a column name (minimum 2 characters).

**Column Metadata Surfacing.** Each suggestion displays: - Column name (exact match from dataset) - Type badge: **numeric** (blue), **categorical** (amber), **text** (purple) - Statistical preview: range and mean for numeric columns, top categorical values for categorical columns, unique count for text columns

**Interaction Model.** Up to 8 suggestions are shown, sorted by prefix match then alphabetically. Users navigate with arrow keys, accept with Tab or Enter, and dismiss with Escape. The suggestion dropdown appears above the input bar to preserve reading flow.

The design philosophy is constraint as enablement: by showing users what columns exist and what types they contain, the system transforms an open ended text input into a guided selection process, dramatically reducing the probability of queries that cannot be meaningfully answered.

## 5.4 Deterministic Render Blocks

Each query response is rendered as a Deterministic Render Block, a modular, five layer output unit. Every layer is independently verifiable and exportable.

**Layer 1: Query Echo.** The original natural language query is displayed verbatim with identified column targets highlighted as badges. This creates an explicit link between the user’s intent and the system’s interpretation.

**Layer 2: Raw Table.** When the query produces tabular output, a paginated data table is rendered (10 rows per page) with column headers matching the dataset schema. Users can export the full table as CSV via a single click. The presence of raw data alongside the analysis enables independent verification.

**Layer 3: Visual Insight.** A Recharts powered visualization renders the query result as a scatter plot, line chart, or bar chart. The chart type is selected by the LLM based on the MPS tensor correlation structure: scatter for bivariate relationships, line for temporal trends, bar for categorical comparisons. This selection is informed by the correlation topology, not arbitrary choice.

**Layer 4: Deterministic Analysis.** The LLM generates a markdown formatted narrative that cites exact correlation coefficients from the pre-computed tensor network. For example:

*The correlation between Systolic\_BP and Age is  $r = 0.847$  (strong positive), indicating that blood pressure increases significantly with age in this cohort.*

The coefficient  $r = 0.847$  is not computed or estimated by the LLM. It is extracted from the injected correlation pairs and cited verbatim.

**Layer 5: SLA Receipt.** A footer strip displays four telemetry metrics: - **MPS Fidelity:** Derived from tensor contraction quality (confidence minus noise shed)  
- **Query Resolution:** LLM self assessed query to data alignment (0 to 100%)  
- **Noise Shed:** Truncation error from MPS contraction - **Block Index and Timestamp:** Audit trail metadata

Blocks are exported as JSON manifests containing all five layers plus MPS parameters (bond dimension, truncation error), enabling reproducibility.

## 5.5 Iteration Chain

The Iteration Chain organizes Render Blocks into a scrollable, vertically stacked audit trail. Each block connects to the next via visual connector lines, forming a legible chain of analytical reasoning.

**Contextual Propagation.** When a user submits a follow up query, the Iteration Chain passes the full conversation history from all prior blocks to the LLM, enabling sequential drill down:

Query 1: “What are the strongest correlations in my data?” Query 2: “Isolate the relationship between Age and Risk\_Score” Query 3: “Show me outliers in Risk\_Score for patients over 60”

Each subsequent query builds on the analytical context established by previous blocks, producing a coherent, traceable analytical narrative.

**Telemetry Footer.** A pinned footer below the prompt bar displays: - Block count in chain - Number of physical indices (columns) - “Deterministic Only” badge, a persistent reminder that all computations are verifiable

**Auto-Scroll.** New blocks and active Translation Ledger animations trigger smooth scrolling to the bottom of the chain, maintaining the user’s attention on the most recent result.

## 6. Evaluation

### 6.1 Accuracy Validation

AURA’s mathematical pipeline is validated using a 15 test end-to-end accuracy suite that spans four orders of magnitude in dataset size. Each test generates a synthetic dataset with known statistical properties, submits it through the full AURA pipeline (upload, clean, encode, solve, correlate), and cross references every returned value against an independent NumPy/SciPy computation.

#	Tier	Dataset	Rows	Cols	Corr Diff	API Time	Accuracy
1	Core	Medical Vitals	10	4	$0.00 \times 10^0$	42ms	100.00%
2	Core	Stock Portfolio (mixed)	50	6	$0.00 \times 10^0$	20ms	100.00%
3	Core	Clinical Trial	500	8	$0.00 \times 10^0$	25ms	100.00%
4	Core	Genomics Expression	2,000	12	$0.00 \times 10^0$	54ms	100.00%
5	Core	Anti-Correlated	500	5	$0.00 \times 10^0$	23ms	100.00%
6	Core	Sparse / Missing (40%)	200	4	$0.00 \times 10^0$	79ms	100.00%
7	Core	WHO Health Data	1,000	8	$0.00 \times 10^0$	37ms	100.00%

#	Tier	Dataset	Rows	Cols	Corr Diff	API Time	Accuracy
8	Core	Wide Dataset	5,000	24	$0.00 \times 10^0$	183ms	100.00%
9	Scale	10K rows	10,000	8	$0.00 \times 10^0$	130ms	100.00%
10	Scale	50K rows	50,000	6	$0.00 \times 10^0$	455ms	100.00%
11	Scale	100K rows	100,000	8	$0.00 \times 10^0$	1.1s	100.00%
12	Scale	250K rows	250,000	6	$0.00 \times 10^0$	2.2s	100.00%
13	RAM	500K rows	500,000	4	$0.00 \times 10^0$	3.2s	100.00%
14	RAM	750K rows	750,000	4	$0.00 \times 10^0$	4.9s	100.00%
15	RAM	1M rows	1,000,000	4	$0.00 \times 10^0$	6.2s	100.00%

**Table 1.** End-to-end accuracy validation results across 15 progressive scale tests. Total rows processed: 2,669,260. All tests achieve zero correlation difference against independent NumPy/SciPy ground truth.

**Mathematical properties verified per test:** - Column statistics (min, max, mean, median, standard deviation), exact match within  $\varepsilon = 10^{-6}$  - Correlation matrix, element wise comparison against `np.corrcoef`, tolerance  $10^{-4}$  - Correlation pairs, sorted by  $|r|$ , filtered at  $> 0.05$ , set equality for top 30 - Solution quality formula,  $(E_{\text{red}} \times 100) + 90.0$ , exact match - DQPU telemetry, solve time, spectral radius, sparsity, throughput sanity checks

The zero correlation difference across all 15 tests is a consequence of AURA’s deterministic pipeline: both the production system and the independent verification use the same mathematical operations (normalization followed by `np.corrcoef` followed by diagonal zeroing followed by pair extraction), differing only in implementation path. This structural identity guarantees reproducibility.

## 6.2 Scalability Analysis

AURA’s computational cost is dominated by two operations:

1. **Correlation matrix computation:**  $O(K^2N)$  where  $K$  is the number of columns and  $N$  is the number of rows. For the largest test (1M rows, 4 columns), this requires approximately  $4 \times 10^6$  floating point operations.
2. **DQPU Execution:**  $O(\text{nnz} \times M \times T)$  where  $\text{nnz}$  is the number of non-zero entries in the sparse coupling matrix,  $M = 4$  is the number of parallel agents, and  $T = 300$  is the number of integration steps.

The API response time scales linearly with row count, as shown by the empirical latency data in Table 1: from 42ms at 10 rows to 6.2s at 1M rows, a ratio consistent with the  $O(N)$  cost of column normalization and correlation computation. The DQPU execution itself operates on the  $K \times K$  coupling matrix (not the  $N \times K$  data matrix), keeping solver time constant regardless of dataset size.

For datasets exceeding 5 million rows, the segmented streaming system partitions the input into blocks processed independently, each constrained to approximately 10 GB of memory. This enables processing of datasets up to 50 million rows (24 columns) on a single 32 GB server without out-of-memory failures.

### 6.3 Auditability Assessment

AURA’s auditability can be assessed along three dimensions:

1. **Coefficient Traceability.** Every correlation coefficient cited in an analysis block can be traced back to the `correlation_data.top_pairs` array in the API response, which in turn is computed by a deterministic `np.corrcoef` call on normalized data. This creates a three link chain: cited value to API response to deterministic computation.
2. **Process Transparency.** The Translation Ledger exposes the query compilation pipeline in real time, showing users which columns were targeted, what matrix product operator was compiled, and what bond dimension governs the contraction. This information is typically hidden in analytical tools, even those targeting technical users.
3. **Export Reproducibility.** Every Render Block can be exported as a JSON manifest containing the complete query context, raw data, analysis, and MPS parameters. A researcher can independently verify any block by re-running the documented computation.

In contrast, a standard LLM based data analysis tool (such as ChatGPT with CSV upload) provides none of these guarantees: the user receives a textual response with no mechanism to verify whether cited statistics were computed, interpolated, or hallucinated.

---

## 7. Related Work

**LLM Assisted Data Analysis.** OpenAI’s Code Interpreter [1] generates and executes Python code to answer data questions, but requires users to audit generated code for correctness. Julius AI and similar tools abstract the code layer but still rely on LLM generated computations. AURA fundamentally differs by removing the LLM from the computational path entirely. It interprets, but does not compute.



**Explainable AI (XAI).** LIME [2] and SHAP [3] provide post hoc explanations of model predictions. These methods explain why a model made a decision but cannot verify whether a generated statistic is correct. AURA’s approach is structural transparency, making the computation process visible, rather than post hoc explanation of an opaque model.

**Tensor Networks in Machine Learning.** Tensor network methods have been applied to supervised learning [5] and anomaly detection [12]. AURA’s use of tensor networks for correlation structure encoding in a data interrogation context is, to the author’s knowledge, novel. The MPS representation serves not as a learning model but as a structured encoding of inter-column relationships that grounds the LLM’s outputs.

**Neurosymbolic AI.** Neurosymbolic systems combine neural pattern recognition with symbolic reasoning [14]. AURA instantiates this paradigm in a specific context: the LLM (neural) provides natural language interpretation while the tensor network pipeline (symbolic) provides deterministic computation. The key structural insight is that the neural and symbolic components communicate through structured coefficient injection rather than end-to-end differentiable training.

**Transparent Data Dashboards.** Business intelligence tools (Tableau, Power BI) provide pre-built visualizations but lack natural language querying. Conversely, LLM chat interfaces provide natural language but lack the structured, auditable output format of dashboards. AURA bridges this gap by combining natural language input with dashboard grade structured output, enriched with mathematical provenance.

---

## 8. Limitations and Honest Disclosure

The following limitations are documented transparently:

1. **No Quantum Hardware.** Despite the use of quantum inspired terminology (Hamiltonian, MPS, QUBO), AURA operates entirely on classical hardware. The Symplectic Euler integration runs on standard CPU and memory. Tensor network formalism is used because it provides a principled framework for correlation structure encoding, not because any quantum computational advantage is claimed.
2. **LLM Dependency.** The natural language interpretation and narrative generation layers depend on the LLM used. While the deterministic pipeline ensures all cited numbers are correct, the interpretation of those numbers, and the narrative connecting them, is subject to the LLM’s language modeling quality. Factual claims about data are verifiable; qualitative insights are not.

3. **Single User Concurrency.** File uploads are serialized via a semaphore to prevent memory contention. Only one file can be processed at a time. This limits throughput in multi-user deployments and represents a scalability bottleneck.
  4. **Bond Dimension Heuristic.** The bond dimension  $\chi$  is set by a simple formula ( $\min(128, \max(16, K \times 8))$ ) rather than adaptively optimized per query or dataset. For datasets with unusual correlation structures (such as highly non-local correlations), this heuristic may under or over allocate bond dimension.
  5. **Column Limit.** The current implementation encodes each numeric column as a node in the coupling graph. Datasets with hundreds of numeric columns would produce large  $K \times K$  correlation matrices, potentially degrading performance. The system has been validated up to 24 columns.
  6. **Correlation Is Not Causation.** AURA reports Pearson correlation coefficients, which measure linear association. The system does not perform causal inference, and the LLM’s narrative may sometimes imply causal relationships that the data does not support. Users must exercise domain judgment when interpreting results.
- 

## 9. Future Work

Several directions for future research emerge from the current work:

1. **Adaptive Bond Dimension Selection.** Developing a data dependent strategy for setting  $\chi$  based on the spectrum of the correlation matrix, potentially using the decay rate of singular values to determine the optimal truncation point.
2. **Multi-User Concurrency.** Implementing a queue based system with per-user memory isolation, enabling multiple simultaneous uploads without contention.
3. **Domain Specific Ontologies.** Integrating controlled vocabularies (ICD-10 for medical data, Gene Ontology for biological data) into the Schema Aware Auto-Suggest system, enabling domain aware query suggestions beyond column names.
4. **User Study.** Conducting a controlled user study with clinical researchers to evaluate whether AURA’s transparency mechanisms measurably increase trust and reduce analytical errors compared to standard LLM based tools.
5. **Non-Linear Correlation Support.** Extending the correlation pipeline to include mutual information or distance correlation metrics, capturing

non-linear relationships that Pearson coefficients miss.

6. **Longitudinal Analysis.** Enabling time series aware interrogation by incorporating temporal indexing into the MPS representation, allowing queries about trends, seasonality, and change points.
- 

## 10. Conclusion

This paper has presented AURA, a neurosymbolic interrogation interface that enforces deterministic transparency in LLM assisted data analysis. By pre-computing all statistical relationships via a tensor network pipeline and injecting verified coefficients into the LLM context, AURA eliminates the possibility of hallucinated statistics while preserving the accessibility of natural language querying.

AURA’s four interface primitives, the Translation Ledger, Schema Aware Auto-Suggest, Deterministic Render Blocks, and the Iteration Chain, each address a specific phase of the analysis pipeline. They collectively create an auditable path from raw data to human readable insight. The system achieves 100% accuracy against independent mathematical ground truth across 15 end-to-end test tiers spanning four orders of magnitude in dataset size.

The core infrastructural insight is separation of concerns at the trust boundary: LLMs are excellent at language but unreliable at computation; deterministic pipelines are excellent at computation but inexpressive in language. AURA assigns each system to its domain of competence and mediates their interaction through structured coefficient injection, producing outputs that are simultaneously accessible and verifiable.

---

## References

- [1] OpenAI. “ChatGPT Code Interpreter.” OpenAI Blog, 2023. <https://openai.com/blog/chatgpt-plugins>
- [2] Ribeiro, M. T., Singh, S., and Guestrin, C. “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier.” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135-1144, 2016.
- [3] Lundberg, S. M. and Lee, S.-I. “A Unified Approach to Interpreting Model Predictions.” Advances in Neural Information Processing Systems, 30, 2017.
- [4] Orús, R. “A Practical Introduction to Tensor Networks: Matrix Product

- States and Projected Entangled Pair States.” *Annals of Physics*, 349, pp. 117-158, 2014.
- [5] Stoudenmire, E. and Schwab, D. J. “Supervised Learning with a Quantum-Inspired Tensor Network.” *Advances in Neural Information Processing Systems*, 29, 2016.
- [6] Lucas, A. “Ising formulations of many NP problems.” *Frontiers in Physics*, 2:5, 2014.
- [7] Hairer, E., Lubich, C., and Wanner, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006.
- [8] Ji, Z., Lee, N., Frieske, R., et al. “Survey of Hallucination in Natural Language Generation.” *ACM Computing Surveys*, 55(12), 2023.
- [9] Lewis, P., Perez, E., Piktus, A., et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” *Advances in Neural Information Processing Systems*, 33, 2020.
- [10] Mehta, R. and Zhu, R. “Blue or Red? Exploring the Effect of Color on Cognitive Task Performances.” *Science*, 323(5918), pp. 1226-1229, 2009.
- [11] Doshi-Velez, F. and Kim, B. “Towards A Rigorous Science of Interpretable Machine Learning.” *arXiv:1702.08608*, 2017.
- [12] Wang, J., Roberts, C., Vidal, G., and Leichenauer, S. “Anomaly detection with tensor networks.” *arXiv:2006.02516*, 2020.
- [13] Han, Z.-Y., Wang, J., Fan, H., Wang, L., and Zhang, P. “Unsupervised Generative Modeling Using Matrix Product States.” *Physical Review X*, 8(3), 031012, 2018.
- [14] Garcez, A. d’A. and Lamb, L. C. “Neurosymbolic AI: The 3rd Wave.” *Artificial Intelligence Review*, 56, pp. 12387-12406, 2023.
- [15] <https://news.mit.edu/2025/shortcoming-makes-llms-less-reliable-1126>