The Journal of Open Source Software

ArrowSpace: introducing Spectral Indexing for vector search

Lorenzo Moriondo [1]

1 Independent Researcher (London, UK / Tokyo, Japan) - tuned.org.uk

DOI: 10.21105/joss.09002

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @sstadick
- @DiogoRibeiro7

Submitted: 29 August 2025
Published: 27 September 2025

Summary

arrowspace (Moriondo, 2025) is a Rust (Rust Team, 2015) library (and relative data structure ArrowSpace) for vector similarity search that goes beyond traditional distance metrics by incorporating spectral graph properties to find structural patterns in high-dimensional data. ArrowSpace adds a spectral dimension that captures structural patterns, enabling more nuanced similarity matching for scientific and structured data applications. arrowspace combines traditional semantic similarity with graph-based spectral properties (Mahadevan, 2006; Spielman, 2007). The library introduces taumode (in mathematical expressions $\lambda\tau$, lambda-tau) indexing, which blends Rayleigh quotient smoothness energy from graph Laplacians (Bai, 2007; Bai & Hancock, 2010) with edge-wise dispersion statistics to create bounded, comparable spectral scores. This enables similarity search that considers both semantic content and spectral characteristics of high-dimensional vector datasets.

Statement of Need

Traditional vector similarity search relies on geometric measures like cosine similarity or Euclidean distance. These methods capture semantic relationships but ignore spectral structure (You, 2025), the patterns in how data points relate to each other across the entire dataset. For scientific applications like protein analysis or signal processing, this limitation means that structurally similar samples may be ranked limited to traditional distance metrics, while geometrically close but spectrally different samples rank highly. ArrowSpace addresses this gap by providing the first integrated spectral-aware indexing for vector databases, enabling similarity search that considers both content and spectral context.

Existing vector databases and similarity search systems lack integrated spectral-aware indexing capabilities. While spectral methods exist in graph theory and signal processing (for spectral clustering, see von Luxburg (2007)), they are typically computationally expensive and they are not considered for database applications. With the increasing demand for vector searching though (in particular, at current state, for the components called "retrievers" in RAG applications (Lewis et al., 2020)), the research on spectral indexing gains traction for database applications. ArrowSpace addresses this gap by providing:

1. **Spectral-aware similarity search** that combines semantic and spectral properties
2. **Bounded synthetic indexing** that produces comparable scores across datasets
3. **Memory-efficient representation** that avoids storing graph structures at query time
4. **High-performance Rust implementation** with potentially zero-copy operations and cache-friendly data layouts

Data Model and Algorithm

`ArrowSpace` provides an API to use taumode that is a single, bounded, synthetic score per signal that blends the Rayleigh smoothness energy on a graph with an edgewise dispersion summary; enabling spectra-aware search and range filtering. Operationally, `ArrowSpace` stores dense features (inspired by CSR (Kelly, 2020) and `smartcore` (Orlov, 2019)) as rows over item nodes, computes a Laplacian on items, derives per-row Rayleigh energies, compresses them via a bounded map $E/(E+\tau)$, mixes in a dispersion term, and uses the resulting taumode both for similarity and to build a λ-proximity item graph used across the API. This way, the taumode score can rely on a synthesis of the characteristics proper of diffusion models and geometric/topological representation of graphs.

Motivation

From an engineering perspective, there is increasing demand for vector database indices that can spot vector similarities beyond the current available methods (L2 distance, cosine distance, or more complex algorithms like HNSW (Malkov & Yashunin, 2020) that require multiple graphs, or typical caching mechanism requiring hashing). New methods to search vector spaces can lead to more accurate and fine-tunable procedures to adapt the search to the specific needs of the domain the embeddings belong to. Furthermore, the most popular embeddings search algorithms focus on single-vector search that has been proved to have theoretical limits (Weller et al., 2025); spectral algorithms like `ArrowSpace` can provide a base for multi-vector search by allowing to index sub-vectors of embeddings.

Foundation

The starting score is Rayleigh as described in Chen (2020). Chen emphasises that the Rayleigh quotient provides a variational characterisation of eigenvalues: it offers a way to find eigenvalues through optimisation rather than solving the characteristic polynomial. This perspective is fundamental in numerical linear algebra and spectral analysis. The synthetic taumode index is built on this base and used to index vector spaces. The treatment is particularly valuable for understanding how spectral properties of matrices emerge naturally from optimisation problems, which connects to applications in data analysis, graph theory, and machine learning.

Basic points:

- Definition: for a feature row x and item-Laplacian L, the smoothness is $E = \frac{x^\top L x}{x^\top x}$, which is non-negative, scale-invariant in x, near-zero for constants on connected graphs, and larger for high-frequency signals; the Rayleigh quotient is the normalised Dirichlet Energy, the discrete Dirichlet energy normalised by signal power.
- Physical Interpretation: Dirichlet energy measure the "potential energy" or "stiffness" of a configuration while the Rayleigh quotient normalises this by the total "mass" or "signal power". The result is a scale-invariant measure of how much energy is required per unit mass (in our case the items-nodes).

taumode and bounded energy

The main idea for this design is to *build a score that synthesises the energy features and geometric features of the dataset* and apply it to vector searching.

Rayleigh and Laplacian as bounded energy transformation score become a bounded map: raw energy E is compressed to $E' = \frac{E}{E+\tau} \in$ using a strictly positive scale τ, stabilising tails and making scores comparable across rows and datasets while preserving order within moderate ranges.

Additional τ selection: taumode supports `Fixed`, `Mean`, `Median`, and `Percentile`; non-finite inputs are filtered and a small floor ensures positivity; the default `Median` policy provides robust

scaling across heterogeneously distributed energies.

Rayleigh, Laplacian and τ selection enable the taumode score, this score can be used as an indexing score for dataset indexing.

Purpose of τ in the Bounded Transform

The τ parameter is crucial for the bounded energy transformation: $\mathbf{E = E/(E+\tau)}$. This maps raw Rayleigh energies from [0,∞) to [0,1), making scores:

- **Comparable across datasets** with different energy scales
- **Numerically stable** by preventing division issues with very small energies
- **Bounded** for consistent similarity computations

Usage Example

```rust
use arrowSpace::builder::ArrowSpaceBuilder;
use arrowSpace::core::ArrowItem;

// Build ArrowSpace from item vectors
let items = vec![
    vec![1.0, 2.0, 3.0],  // Item 1
    vec![2.0, 3.0, 1.0],  // Item 2
    vec![3.0, 1.0, 2.0],  // Item 3
];

let (aspace, _graph) = ArrowSpaceBuilder::new()
    .with_rows(items)
    .with_lambda_graph(1e-3, 6, 2.0, None)
    .build();

// Query with lambda-aware similarity
let query = ArrowItem::new(vec![1.5, 2.5, 2.0], 0.0);
// with alpha=1.0 and beta=0.0, same results as cosine similarity
let results = aspace.search_lambda_aware(&query, 5, 1.0, 0.0);
```

Practical Impact on Search

The choice of taumode affects how the bounded energies E' distribute in $[0, 1)$:

```rust
// Low-energy feature with different τ values
let energy = 0.01;
let tau_small = 0.001;  // E' = 0.01/0.011 ≈ 0.91 (high sensitivity)
let tau_large = 0.1;    // E' = 0.01/0.11 ≈ 0.09 (low sensitivity)
```

Summary and Conclusion

Performance Characteristics

Computational Complexity

- **Index Construction** is $O(N^2)$ for similarity graph (already identified a solution to make this into $O(NlogN)$), and $O(F \cdot nnz(L))$ for taumode computation.
- **Query Time** is $O(N)$ for linear scan, O(1) for taumode lookup, and $O(log(N) + M)$ for range-based lookup.
- **Memory Usage** is $O(F \cdot N)$ for dense storage, and $O(N)$ for taumode indices.

Benchmarks

The library includes benchmarks comparing ArrowSpace with baseline cosine similarity, the benchmark baseline shows 25-45% overhead for taumode-aware index building (lambda_similarity method) compared to a pure cosine index. This is a computational cost to pay for allowing the extension in search capabilities that the additional indexing layer enables. Considering the novelty of the implementation, these measurements are not very meaningful and have to be taken just as a starting reference. The library and the paper aim to find usable differences in results returned by the novel search harness and this is achieved, as demonstrated in compare_cosine example where the index returned by the query are comparable but not the same as cosine similarity (index 30 being the outlier not spotted by cosine similarity). Result of the simple test:

```
Baseline cosine top-3:
  1. idx=3 (P0004) score=1.000000
  2. idx=6 (P0007) score=0.999573
  3. idx=0 (P0001) score=0.999325
ArrowSpace shape after transpose: (24, 64)

ArrowSpace (alpha=1, beta=0) top-3 (equivalent ArrowSpace):
  1. idx=3 (P0004) score=1.000000
  2. idx=6 (P0007) score=0.999573
  3. idx=0 (P0001) score=0.999325

ArrowSpace (alpha=0.9, beta=0.1) top-3 (spectral-adjusted ArrowSpace):
  1. idx=6 (P0007) score=0.970372
  2. idx=30 (P0031) score=0.970268
  3. idx=3 (P0004) score=0.967810
  4. idx=0 (P0001) score=0.967502

Match (baseline vs Arrow cosine): OK
Jaccard(baseline vs taumode-aware): 0.750
```

Results

ArrowSpace has substantial potential for raw improvements plus all the advantages provided to downstream more complex operations like matching, comparison, and search due to the λ spectrum. Capabilities are demonstrated in the other tests present in the code. The proteins_lookup example demonstrates the functionality in a small dataset. The time complexity for a range-based lookup is the same as a sorted set $O(log(N) + M)$. As demonstrated in the proteins_lookup example, starting from a collection of λs with a standard deviation of 0.06, it is possible to sort out the top-k nearest neighbours with a range query on an query interval of $\lambda \pm 10^{-7}$.

Conclusion

ArrowSpace provides a novel approach to vector similarity search by integrating spectral graph properties with traditional semantic similarity measures. The taumode indexing system offers a memory-efficient way to capture spectral characteristics of vector datasets while maintaining practical query performance. The library's design emphasises both mathematical rigor and computational efficiency, making it suitable for scientific applications requiring spectral-aware similarity search.

The combination of Rust's performance characteristics with innovative spectral indexing algorithms positions ArrowSpace as a valuable tool for researchers and practitioners working with high-dimensional vector data where both semantic content and structural properties matter.

The definition of a core library to be used to develop a database solution based on spectral indexing is left to another paper that will include further improvements in terms of algorithms and idioms to make this approach to indexing feasible and efficient in modern cloud installations.

Acknowledgements

For this research, LLMs have been used extensively in the ideation and development phase.

References

Bai, X. (2007). *Heat kernel analysis on graphs* [PhD thesis]. University of York.

Bai, X., & Hancock, E. R. (2010). Heat kernels, manifolds and graph embedding. *Pattern Recognition*. https://doi.org/10.1007/978-3-540-27868-9_20

Chen, G. (2020). *Math 253: Rayleigh quotient lecture notes*. San Jose State University.

Kelly, T. (2020). Compressed sparse row format for representing graphs;*Login: The Magazine of USENIX & The Advanced Computing Systems Association*, *45*(4), 76–83. https://www.usenix.org/system/files/login/articles/login_winter20_16_kelly.pdf

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv Preprint*. https://doi.org/10.48550/arXiv.2005.11401

Mahadevan, S. (2006). *Spectral graph theory lecture notes*. University of Massachusetts, Amherst. https://people.cs.umass.edu/~mahadeva/cs791bb/lectures-s2006/lec4.pdf

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Moriondo, L. (2025). *ArrowSpace-rs: Spectral vector search with lambda-tau indexing*. https://doi.org/10.36227/techrxiv.175751921.18542359/v1

Orlov, V. (2019). *SmartCore: Machine learning library for Rust*. https://smartcorelib.org/

Rust Team. (2015). *The Rust programming language*. https://www.rust-lang.org/

Spielman, D. A. (2007). *Spectral graph theory - Lecture 7*. Yale University.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*, 395–416. https://doi.org/10.1007/s11222-007-9033-z

Weller, O., Boratko, M., Naim, I., & Lee, J. (2025). *On the theoretical limitations of embedding-based retrieval*. https://arxiv.org/abs/2508.21038

You, K. (2025). Semantics at an angle: When cosine similarity works until it doesn't. *arXiv Preprint arXiv:2504.16318*. https://doi.org/10.48550/arXiv.2504.16318