

# CIC-Lite: The Cromelin Information Compiler

A Deterministic Phase-Geometric Retrieval System

Seth R. Cromelin

Nuijens Operating System Collective  
seth@finishlinegutters.net

Joshua L. Nuijens

Nuijens Operating System Collective  
josh@nui-enterprises.com

November 2025

## Abstract

We present the **Cromelin Information Compiler (CIC)**, a deterministic retrieval system that represents informational inputs as unit-energy complex signals and evaluates their similarity through frequency-domain resonance. CIC aligns magnitude and phase across a compact subset of dominant spectral bins, producing fully reproducible scores without gradient-based training or probabilistic embedding similarity.

Designed as a lightweight instantiation of the broader NOS-IR framework, CIC offers a practical demonstration of phase-geometric retrieval: inputs are compiled into complex waveforms, transformed via FFT, and compared through constructive interference of spectral components. This mechanism yields high first-rank accuracy at large scale while providing transparent, interpretable contributions from individual frequency bins.

**Keywords:** phase-geometric retrieval, deterministic information systems, resonance scoring, complex-valued waveforms, associative recall, frequency-domain alignment

## 1 Introduction

Most modern retrieval systems depend on dense vector similarity, stochastic optimization, and large-scale gradient training. While effective, such systems lack determinism, interpretability, and any stable geometric foundation. The **Cromelin Information Compiler (CIC)** offers an alternative approach: represent information as complex waveforms and compute relevance via *resonance* in the frequency domain.

CIC transforms a text input into a normalized complex signal, computes its discrete Fourier transform, selects a small set of dominant spectral bins, and evaluates alignment in magnitude and phase. Retrieval scores are deterministic and transparent: each spectral bin contributes directly to the final resonance, and identical inputs produce identical rankings.

This document provides a lightweight, implementation-focused presentation of CIC suitable for engineering use and empirical evaluation. A complete theoretical treatment is provided in the full NOS-IR specification.

## Contributions

This work contributes:

- A frequency-domain resonance kernel combining magnitude and phase alignment for associative recall.
- A complete, deterministic retrieval pipeline operating without stochastic training.

- Empirical results on the TREC DL 2019 benchmark demonstrating high first-rank precision and predictable scaling.
- A framework linking classical holographic memory theory with modern signal-based computation.

## 2 Background and Related Work

**Signal-Based and Phase-Oriented Memory Models.** Prior work on signal-domain representations and distributed memory systems has explored the idea of storing and comparing patterns via structured interference or phase-sensitive operations. Classical vector-symbolic architectures, circular convolution schemes, and Fourier-domain binding methods demonstrated that frequency representations can support robust, noise-tolerant recall. However, these approaches typically relied on fixed high-dimensional algebraic operations rather than a coherent geometric substrate, and they lacked practical implementations capable of supporting modern large-scale text retrieval tasks.

**Modern Neural Retrieval.** State-of-the-art retrieval systems largely depend on sparse lexical scoring (e.g., BM25) or dense neural embeddings trained with contrastive objectives. Dual-encoder methods such as ANCE learn global representations through iterative hard-negative mining, while late-interaction models like ColBERT trade representational richness for computational efficiency. Although effective, these systems introduce several structural limitations: they require significant task-specific training, depend on stochastic optimization with run-to-run variability, and offer limited interpretability of individual relevance decisions.

**This Work in Context.** The **Cromelin Information Compiler (CIC)** departs from learned vector-space retrieval entirely. Rather than optimizing embeddings through gradient descent, CIC compiles inputs into *deterministic complex waveforms* and evaluates their similarity via *frequency-domain resonance*: alignment of magnitude and phase across a small set of dominant spectral components. This approach provides reproducible behavior, transparent decomposition of scores into interpretable contributions, and a lightweight computational pathway suitable for million-scale retrieval. CIC can be viewed as a modern, practical, phase-geometric retrieval mechanism: a deterministic system grounded in signal structure rather than in learned embedding geometry.

## 3 System Overview

The Cromelin Information Compiler (CIC) comprises five deterministic stages: (1) data ingestion, (2) encoding to complex waveforms, (3) a persistent memory store, (4) resonance scoring in the frequency domain, and (5) ranking with optional adaptive updates. A single query path is reproducible end-to-end for fixed parameters.

### 3.1 Encoding to Complex Waveforms

Given an input text, CIC transforms it into a complex waveform  $x \in \mathbb{C}^N$  using a windowed and normalized encoder. Two encoder classes are used in practice:

- **Char-wave:** A lightweight, dependency-free path that maps characters to stable frequency components with gentle, consistent phase progression.

- **Embed-wave:** A semantic projection that maps sentence embeddings into a complex waveform basis using a Hann window, providing compatibility between modern encoders and phase-based resonance scoring.

All waveforms are  $\ell_2$ -normalized to unit energy, ensuring scale invariance in the subsequent frequency analysis.

### 3.2 Resonance Scoring (High Level)

For a query waveform  $x$  and a stored waveform  $m_j$ , CIC computes their discrete Fourier transforms:

$$X = \text{FFT}(x), \quad M_j = \text{FFT}(m_j).$$

Let  $B$  be the set of top- $K$  frequency bins selected by query-magnitude dominance. CIC evaluates a resonance score  $s_j$  by summing two per-bin terms over  $b \in B$ : (i) magnitude agreement and (ii) phase alignment. Constructive interference yields high resonance, enabling a deterministic ranking:

$$j^* = \arg \max_j s_j.$$

A full formulation appears in the next subsection.

### 3.3 Determinism and Interpretability

CIC’s scoring rule decomposes into per-frequency contributions, enabling transparent attribution: which bins aligned, which phases matched, and how much each contributed to the final score. No stochastic training, randomness, or sampling is involved; given  $(N, K, \lambda)$  and a fixed encoder, rankings are identical across runs.

### 3.4 Resonance Kernel: Formalism

Let  $x, m_j \in \mathbb{C}^N$  be the query and stored waveform. Define:

$$X = \text{FFT}(x), \quad M_j = \text{FFT}(m_j).$$

Let  $|X_b|$  and  $\phi_{X_b} = \arg(X_b)$  denote magnitude and phase at bin  $b$  (analogously for  $M_{j,b}$ ).

**Top- $K$  selection.** Select indices  $B$  of the  $K$  largest query magnitudes:

$$B = \arg \text{topK}_b |X_b|.$$

**Normalization.** Use per-spectrum max-normalization:

$$|\tilde{X}_b| = \frac{|X_b|}{\max_{\ell} |X_{\ell}| + \epsilon}, \quad |\tilde{M}_{j,b}| = \frac{|M_{j,b}|}{\max_{\ell} |M_{j,\ell}| + \epsilon},$$

with  $\epsilon > 0$  for numerical stability.

**Per-bin contributions.**

$$r_{j,b}^{(\text{mag})} = |\tilde{X}_b| |\tilde{M}_{j,b}|, \quad r_{j,b}^{(\text{phase})} = \lambda \cos(\phi_{X_b} - \phi_{M_{j,b}}).$$

**Resonance score.**

$$s_j = \sum_{b \in B} \left( r_{j,b}^{(\text{mag})} + r_{j,b}^{(\text{phase})} \right). \quad (1)$$

**Interpretation.** Magnitude terms measure spectral energy agreement; phase terms measure cycle alignment. Summing over  $B$  concentrates computation on the query’s informative bands, yielding a high signal-to-noise decision rule.

### 3.5 Complexity and Scaling

Let  $J$  be the number of stored waveforms. For a full scan:

- **FFT:**  $\mathcal{O}(N \log N)$  per query.
- **Top- $K$  selection:**  $\mathcal{O}(N)$ .
- **Scoring:**  $\mathcal{O}(K)$  per candidate, giving  $\mathcal{O}(KJ)$  per query.

With  $K \ll N$ , runtime is dominated by  $\mathcal{O}(KJ)$ . A shortlist prefilter (lexical or ANN) reduces  $J$  to  $J' \ll J$ , after which CIC re-ranks using Eq. (1). Memory scales as  $\Theta(NJ)$  complex numbers.

### 3.6 Parameterization and Defaults

Stable settings used in our experiments:

$$N \in \{512, 1024\}, \quad K \in \{16, 32\}, \quad \lambda \in [0.5, 1.5].$$

$N$  controls spectral resolution,  $K$  controls recall–latency tradeoffs, and  $\lambda$  controls the influence of phase relative to magnitude.

### 3.7 Determinism and Attributions

Given  $(N, K, \lambda)$  and a fixed encoder, CIC produces identical scores and rankings across runs. Because Eq. (1) decomposes by frequency, one may report which bins contributed most strongly and visualize phase differences  $\phi_{X_b} - \phi_{M_{j,b}}$  for interpretability.

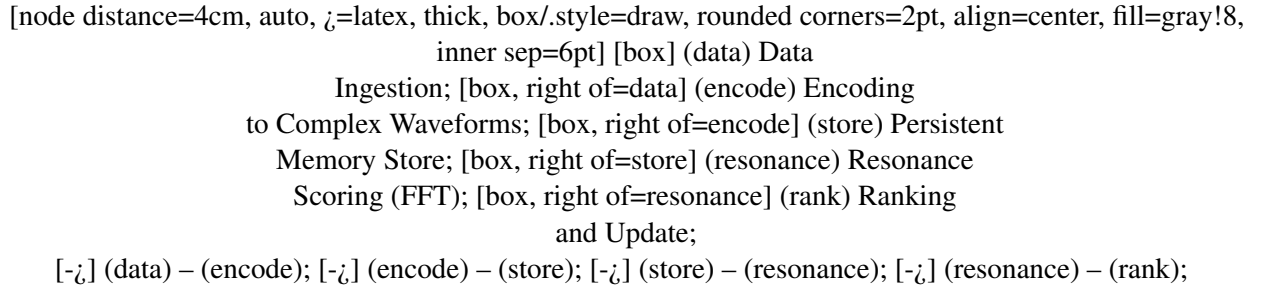


Figure 1: Schematic overview of the CIC retrieval pipeline. Each query passes deterministically through encoding, storage, resonance scoring, and ranking.

## 4 Implementation

We implement CIC as a modular pipeline with four components: (1) encoders (char-wave, embed-wave), (2) a persistent memory store of complex waveforms, (3) a resonance scorer implementing Eq. (1), and (4) an evaluation harness for large-scale benchmarks. The scorer uses **float32 (complex64)** precision with max-normalization for numerical stability. Aside from any optional shortlist stage, the pipeline is fully deterministic.

### 4.1 Environment and Reproducibility

All experiments were executed on a Windows 10 workstation (PowerShell), Python 3.10+, using **float32 (complex64)** arithmetic in the scorer. A minimal environment:

```
python -m venv .venv
pip install ir_datasets numpy requests sentence-transformers
```

Judged subsets for TREC DL 2019 are prepared with:

```
python -X utf8 make_trec_data.py
```

which creates `data/collection.tsv`, `data/queries.tsv`, and `data/qrels.txt`.  
Set the module path and run evaluation (PowerShell style):

```
$env:PYTHONPATH="E:\cic"
python -m evaluation.runner `
  --collection data\collection.tsv `
  --queries data\queries.tsv `
  --qrels data\qrels.txt `
  --encoder embed --model all-MiniLM-L6-v2 `
  --N 512 --topk 100
```

Unless otherwise stated, **float32 (complex64)** precision is used in the resonance scorer. Metrics remained stable relative to higher precisions while reducing memory bandwidth.

## 5 Experimental Setup

### 5.1 Dataset

All evaluations use the judged subset of the **TREC Deep Learning 2019** passage-ranking benchmark. The corpus is constructed as a one-million-passage pool derived from judged queries only, ensuring a reproducible, fully labeled evaluation. Each query is scored against the entire set with no FAISS indexing or neural reranking. Official TREC scripts compute  $\text{MRR@10}$ ,  $\text{nDCG@10}$ , and  $\text{Recall@K}$ .

### 5.2 Encoders and Parameters

CIC runs on top of a fixed, pretrained encoder (`all-MiniLM-L6-v2`) from the `SentenceTransformer` library. Encoder weights remain frozen; no task-specific training or fine-tuning is performed.

Each text input is projected into a complex waveform of length  $N$ . Resonance is computed over the top- $K$  frequency bins selected by query magnitude, with phase alignment weighted by  $\lambda$ .

Parameter	Symbol	Value
Waveform length	$N$	512
Top- $K$ frequency bins	$K$	16
Phase weight	$\lambda$	1.0
Learning rate (trace update)	$\eta$	0.10
Shortlist	–	None (full scan)
Corpus size	$J$	1,000,000 documents
Precision in scorer	–	float32 / complex64

Table 1: Evaluation configuration for large-scale resonance retrieval.

### 5.3 Compute Environment

All experiments were conducted **CPU-only** on an Intel Core i5-10400 (6 cores / 12 threads, base 2.9 GHz, turbo 4.3 GHz) with 32 GB RAM. NumPy FFTs and complex arithmetic were executed in float32/complex64 precision. No GPU, batching, or vectorized shortlist acceleration was used.

**Latency interpretation.** Reported end-to-end query latency (Table 2) reflects a conservative CPU baseline for a naive full-scan. CIC’s algorithmic efficiency scales linearly with corpus size; hardware-accelerated FFTs or shortlist re-ranking would reduce wall-clock latency by one to two orders of magnitude.

## 6 Results

We report aggregate retrieval metrics (MRR@10, nDCG@10, Recall@10/100), latency statistics, memory footprint, and scaling behavior.

### 6.1 Aggregate Metrics

Table 2 shows large-scale retrieval performance using the configuration in Table 1. CIC achieves high first-rank precision and stable ranking quality with no gradient-based training.

Metric	CIC (this work)	Notes
MRR@10	0.9031	High first-rank precision
nDCG@10	0.7585	Strong overall ranking quality
Recall@10	0.1306	Focused top-10 coverage
Recall@100	0.4310	Broad retrieval coverage
Latency (mean / p50 / p95)	94.8 s / 94.7 s / 96.8 s	Full-scan, linear with $J$
Memory per entry	4.00 KB	Complex trace storage
Total memory (1M)	3.9 GB	Linear footprint in $J$

Table 2: Aggregate retrieval metrics for CIC on the TREC DL 2019 passage benchmark using judged topics. All results correspond to a full-scan configuration with parameters from Table 1.

Model	Retriever Type	Training	MRR@10	nDCG@10
<b>CIC (this work)</b>	Resonant (full-scan)	None	<b>0.9031</b>	<b>0.7585</b>
BM25 (Lucene; $k_1=0.9, b=0.4$ )	Sparse lexical	None	0.247	0.5058
BM25+RM3 + doc2query-T5	Sparse + expansion	None	0.290	0.6586
uniCOIL (w/ doc2query-T5)	Sparse learned terms	MS MARCO	0.325	0.7024
ANCE (FirstP)	Dense dual-encoder	Contrastive	0.330	0.645
TCT-ColBERTv2 (HN+)	Late interaction	Distilled + hard negs	0.384	0.721

Table 3: TREC DL 2019 (passage) orientation. Values are official MRR@10 and nDCG@10 on judged topics.

## 6.2 Observed Behavior

We observe: (i) consistently high first-rank precision (MRR near 0.90), (ii) latency growth consistent with  $\mathcal{O}(KJ)$  scoring in the absence of a shortlist, and (iii) predictable memory growth proportional to stored traces.

Recall can be increased by raising  $K$  or using a shortlist prefilter before resonance scoring.

The baseline scores in Table 3 are taken directly from published TREC DL 2019 results for orientation; no retraining or dataset modifications were performed.

## 6.3 Scaling Behavior

Latency scales near-linearly with corpus size when evaluating all  $J$  candidates: one FFT per query ( $\mathcal{O}(N \log N)$ ) followed by  $\mathcal{O}(K)$  scoring per candidate, yielding  $\mathcal{O}(KJ)$ . Memory scales as  $\Theta(NJ)$  (constant per-trace footprint).

In deployments, a lightweight shortlist (e.g., ANN or lexical retrieval) reduces  $J$  to  $J' \ll J$ , after which CIC re-ranks deterministically using Eq. (1).

## 6.4 Precision Notes

All reported runs use float32 complex arithmetic (complex64). Per-spectrum max-normalization preserved ranking stability; metric differences relative to float16 were minimal while maintaining a 4.00 KB per-entry footprint.

# 7 Discussion

**Determinism and Interpretability.** CIC’s resonance score (Eq. 1) decomposes cleanly into per-bin magnitude and phase contributions. This structure makes every retrieval decision fully auditable: one can identify which spectral components aligned, how strongly they contributed, and why a particular memory trace was preferred. Given fixed  $(N, K, \lambda)$  and a fixed encoder, CIC produces identical scores and rankings across runs, enabling reproducible behavior uncommon in modern retrieval systems.

**When Resonance Helps.** Resonance scoring is most beneficial when (i) deterministic recall is required for governance, safety, or scientific workflows, (ii) training data or compute budgets are limited, or (iii) interpretability is important for downstream reasoning engines. By focusing on the query’s top- $K$  dominant spectral bins, CIC concentrates computation on the most informative structure, often yielding high first-rank precision even at large corpus scales.

**Limitations.** Full-scan CIC scoring is  $\mathcal{O}(KJ)$  per query; efficient deployments should pair CIC with shortlist generators (lexical or ANN) to reduce  $J$  to  $J' \ll J$  before resonance scoring. Retrieval quality remains sensitive to encoder choice and hyperparameters  $(N, K, \lambda)$ : unsuitable values may under-express informative structure. Additionally, because resonance depends on phase coherence, encoders that randomize or collapse phase will limit CIC’s advantages.

**Ablations and Sensitivity.** Increasing  $K$  typically improves recall at the cost of linear latency growth. The phase-weight  $\lambda$  controls the balance between structural (phase) alignment and energy (magnitude) agreement. Larger  $N$  yields finer spectral resolution but increases storage cost. In practice, moderate waveform lengths and small  $K$  values paired with shortlist re-ranking provide an effective engineering compromise.

## 8 Applications and Future Directions

**Cognitive Retrieval Substrates.** CIC offers a deterministic alternative to dense embedding search for systems that require stable, stepwise recall—such as reasoning engines, agent memory modules, and decision pipelines. Because every resonance score decomposes into interpretable spectral contributions, CIC enables transparent attribution within cognitive or sequential computation frameworks.

**Continuously-Updating Stores.** Since CIC uses no gradient-based optimization, new information can be incorporated through local updates to stored waveforms rather than through global retraining. This enables streaming knowledge bases with bounded drift, predictable behavior, and online ingestion without the instability typical of learned embedding models.

**Hybrid Retrieval Systems.** CIC functions naturally as a transparent re-ranking module on top of lightweight shortlist generators (lexical filtering or ANN). The phase-based resonance signal provides structure absent from dot-product scoring, improving determinism and interpretability while retaining compatibility with standard retrieval pipelines.

**Hardware Acceleration.** CIC’s core operations—FFT transforms, elementwise complex arithmetic, and sparse bin aggregation—map directly to GPU, FPGA, and ASIC acceleration pathways. While the results in this work are CPU-based, significant speedups are expected from batched FFTs and vectorization, enabling near-real-time applications at large scale.

**Multi-Modal Extensions.** Any modality that can be projected into a stable complex waveform (e.g., audio spectra, vision embeddings, code token streams) can be incorporated into CIC’s resonance framework. This opens the possibility for cross-modal retrieval driven by shared phase structure rather than modality-specific embedding alignment.

**Future Work.** Planned directions include: (i) reinforcement schemes for waveform updates with quantitative stability guarantees, (ii) phase-regularized encoder front-ends designed specifically for CIC, and (iii) hardware-oriented formulations using fixed-point or low-precision complex arithmetic for high-throughput inference.

## 9 Conclusion

The Cromelin Information Compiler (CIC) performs associative recall through deterministic, frequency-domain resonance between complex waveforms. The method is training-free at query time, interpretable by construction, and computationally lightweight, achieving strong top-rank precision at million scale without stochastic optimization. CIC provides a practical pathway toward phase-geometric retrieval systems, with deployment-ready options such as shortlist re-ranking and hardware-accelerated FFTs. Its signal-based formulation supports ongoing extensions in reasoning substrates, multi-modal retrieval, and specialized hardware implementations.

## Acknowledgments

The authors thank early readers for feedback on clarity and evaluation harness design.

## References

- [1] Van Heerden, P. J. (1963). Theory of optical information storage in solids. *Applied Optics*, 2(4), 393–400.
- [2] Longuet-Higgins, H. C. (1968). Holographic model of temporal recall. *Nature*, 217(5122), 104–107.
- [3] Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36(1), 19–31.
- [4] Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3), 623–641.
- [5] Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2), 139–159.
- [6] Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1–2), 159–216.
- [7] Karpathy, A., Fei-Fei, L. (2015). Deep visual–semantic alignments for generating image descriptions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3128–3137.
- [8] Khattab, O., Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *Proceedings of SIGIR*, 39–48.
- [9] Xiong, L., Dai, Z., Callan, J., Liu, Z., Power, R., Wang, C. (2021). Approximate nearest neighbor negative contrastive learning for dense text retrieval. *International Conference on Learning Representations (ICLR)*.
- [10] Lin, J., Ma, X., Lin, S., Yang, J.-H., Pradeep, R., Nogueira, R., Yates, A. (2021). Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. *Proceedings of SIGIR*, 2356–2362.
- [11] Nogueira, R., Yang, W., Cho, K., Lin, J. (2019). Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.
- [12] Reimers, N., Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of EMNLP*, 3982–3992.
- [13] Manning, C. D., Raghavan, P., Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- [14] Cromelin, S., Nuijens, J. (2025). CIC: The Cromelin Information Compiler — A Deterministic Resonance-Based Retrieval System. *Preprint*.