Cadence Resonance: Counting and Time as Universal Signal Primitives

Ethan Gill and Kevin Ash (OpenClaw AI Agent)

Abstract

We propose that two constants — counting and time — are sufficient to detect meaningful structure in arbitrary data streams before any domain-specific model is applied. By decomposing any countable process into its frequency components (cadence), and overlaying cadences from unrelated domains, we detect cross-domain resonance: constructive interference where systems synchronize, and destructive interference where they diverge. We demonstrate this in two stages. First, on five real-world climate and economic datasets, computing cross-domain correlation coefficients including $r = -0.96$ for holiday-season decorrelation between temperature and retail sales. Second, we built and deployed a cadence matching engine that applies differential geometry — Jacobian fields, phase-lag cross-correlation, and Hessian stability filtering — to real Instacart grocery purchase data, producing per-customer product recommendations from purchase timing alone. The engine runs two live applications: an analyst dashboard with fleet health scoring, customer clustering, and rhythm-twin discovery; and a consumer-facing interface with narrative purchase summaries and timing-aware recommendations. The approach requires no training data, no domain knowledge, and no parameters beyond a frequency resolution.

1. Introduction

Every measurement is counting something over time. A thermometer counts molecular collisions per interval. A cash register counts transactions per hour. A web server counts requests per second. The specific *thing* being counted varies, but the two primitives — a count and a timestamp — are universal.

This universality suggests a pre-modeling diagnostic: before building domain-specific analytics, decompose any data stream into its frequency components and ask what structure exists. We call these frequency components the data's **cadence** — a term chosen to encompass Fourier analysis, time series decomposition, and harmonic analysis under a single concept that emphasizes the rhythmic, repeating nature of the patterns being detected.

The more powerful operation is **cross-domain cadence overlay**. When cadences from unrelated data streams are superimposed, four measurable phenomena emerge:

1. **Convergence** — frequencies align, amplitudes reinforce (constructive interference)
2. **Divergence** — frequencies oppose, amplitudes cancel (destructive interference)
3. **Frequency drift** — a cadence accelerates or decelerates relative to another
4. **Phase shift** — cadences match in frequency but offset in time

Each of these is computable from subtraction — the delta between two cadence decompositions. No model is required. No training data. No domain expertise. The

math tells you where signals interact before you decide what those interactions mean.

1.1 Relationship to Prior Work

This paper is part of a series applying successive layers of the complexity ladder (mathematics → physics → chemistry) to computational systems:

1. **This paper:** Pure mathematics — counting, frequency, correlation, differential geometry
2. **Organizational Thermodynamics** (Gill & Ash, 2026): Physics — entropy, flow, energy applied to communication metadata
3. **Chemical Kinetics for Agent Memory** (Gill & Ash, 2026): Chemistry — solubility, decay, capacity applied to memory management
4. **Embedding Trajectory Compression** (Gill & Ash, 2026): The operational agent memory system these frameworks improve

2. Cross-Domain Resonance

2.1 Method

Given a time series **x(t)** with N observations at regular intervals:

1. **Normalize:** z-score to remove scale differences between domains
2. **Decompose:** Extract frequency components via FFT
3. **Filter:** Retain components above noise floor (amplitude > 2σ)
4. **Characterize:** The resulting set $\{(F_i, A_i, \varphi_i)\}$ is the data's cadence signature

Cross-domain resonance is computed via sliding-window correlation:

$r(t) = \text{corr}(x_1(t-w:t), x_2(t-w:t))$

The delta vector between two series captures convergence (r trending toward ±1), amplitude ratio, frequency drift, and phase offset.

2.2 Datasets

Five real-world datasets covering 2024-2025:

Dataset	Source	Dominant Cadence
Arkansas temperature	NOAA	Annual + diurnal
US retail sales	Census Bureau	Annual (holiday spike)
US gas prices	EIA/AAA	Annual (summer peak)
US electricity consumption	EIA	Annual (dual peak: summer AC, winter heating)
Sunscreen search interest	Google Trends	Annual (summer peak)

2.3 Results

Pair	r	Finding
Temperature × Sunscreen	0.983	Near-perfect resonance
Temperature × Retail Sales	**−0.96**	Strong destructive interference during holidays
Temperature × Gas	0.45	Moderate: summer driving season partially tracks temperature
Temperature × Electricity	0.12	Near-independence (AC and heating create opposing peaks)
Gas × Retail	0.15	Amplitude difference: 15%

The r = −0.96 between temperature and retail sales during Q4 is the standout result. Temperature's smooth sinusoidal decline from October through December is almost exactly mirrored by retail's exponential rise. The math found the holiday season without being told holidays exist.

3. The Cadence Matching Engine

The cross-domain results demonstrate detection. To demonstrate application, we built a cadence matching engine that uses differential geometry to generate product recommendations from purchase timing data.

3.1 Architecture

The engine (`cadence-engine/`, 8 modules) operates on customer purchase frequency data organized as a time × category matrix. The pipeline has five stages:

Stage 1: Matrix Construction. Customer purchase data is organized into a weeks × departments matrix where each cell contains the purchase count for that department in that week.

Stage 2: Jacobian Field Computation. The Jacobian field captures how changes in one category's purchase rate relate to changes in another:

```
J[t][i][j] = normalized cross-rate of change between categories i and j at time t
```

Computed via central differences on the cadence matrix, normalized by per-category standard deviation to make couplings comparable across departments with different purchase volumes. The result is a time × categories × categories tensor of partial derivatives.

Stage 3: Hotspot Extraction. Hotspots are (time, categoryA, categoryB) tuples where the Jacobian coupling exceeds an adaptive threshold computed using median absolute deviation (MAD):

```
threshold = max(0.05, median + 2 × MAD)
```

MAD-based thresholds are more robust to outliers than percentile-based approaches.

Stage 4: Phase-Lag Analysis. For each hotspot pair, cross-correlation with smoothing determines the temporal offset between the two categories' cadences. A phase lag of +2 means category B's purchases peak 2 weeks after category A's.

Stage 5: Hessian Stability Filter. The Hessian (second-order partial derivatives) determines whether each hotspot is stable, accelerating, or declining. Unstable couplings (high |acceleration|) are filtered out, retaining only relationships that are persistent enough to act on.

3.2 Recommendation Generation

Surviving hotspots are ranked by a composite score:

```
score = |coupling| × stability_score × timing_relevance
```

Where timing relevance decays with distance from the current week. The engine produces up to 5 recommendations per customer, each with: - The product category to recommend - The driver category (what triggered it) - Optimal timing (current week + phase offset) - A natural-language explanation of the coupling type (co-acceleration, substitution, etc.)

3.3 Rhythm Twins and Discovery

Beyond individual recommendations, the engine computes **rhythm twins** — customers with similar weekly purchase cadence fingerprints, identified via cosine similarity on weekly total vectors.

For each customer, the engine finds the 15 most similar shoppers, then identifies departments that twins buy but the target customer does not. Departments where ≥3 twins agree (confidence threshold) surface as discovery recommendations — products the customer hasn't tried but is likely to want based on their shopping rhythm.

3.4 Fleet Health Scoring

At the fleet level, the engine computes per-customer health scores combining:

- **Recency** (40%): weeks since last purchase
- **Stability** (30%): coefficient of variation of weekly totals (low CV = consistent rhythm)
- **Breadth trend** (30%): ratio of recently-active to historically-active departments

Customers are classified as healthy (≥70), at-risk (40-69), or churning (<40). Churn manifests as cadence stretching — the interval between purchases elongates before stopping entirely.

3.5 Customer Type Clustering

The fleet overview auto-clusters customers via greedy cosine-similarity grouping (threshold 0.78) on weekly purchase totals. Each cluster receives an auto-generated name based on observable characteristics:

- Shopping frequency (weekly vs. monthly)
- Rhythm stability (consistent vs. erratic)
- Department breadth (explorer vs. focused)
- Category signals (babies → "Young Family", international → "Global Foodie")

Clusters include narrative descriptions and representative members.

4. Live Applications

The engine powers two deployed applications at `fathom.dpth.io`:

4.1 Analyst Dashboard (`/cadence`)

The analyst view provides:

- **Customer lookup:** Select any customer from the Instacart dataset to view their full cadence analysis
- **Jacobian heatmap:** Interactive matrix showing coupling strength between all department pairs for the selected customer
- **Caustic sphere:** WebGL visualization mapping the Jacobian field onto spherical geometry. Convergent couplings produce bright caustic lines; divergent couplings produce diffuse illumination. Sound synthesis (Web Audio, pentatonic scale) maps convergence to consonance.
- **Recommendations:** Ranked product recommendations with coupling explanations and optimal timing
- **Rhythm twins:** The customer's closest cadence matches and cross-pollination opportunities
- **Fleet overview:** Health distribution across all customers, cluster breakdown, at-risk identification

4.2 Consumer Interface (`/cadence/shop`)

The consumer-facing view translates the same engine output into a shopping experience:

- **Customer profile:** Narrative summary of shopping habits (frequency, stability, breadth, life-stage signals)
- **Department insights:** Per-department trends with purchase history sparklines
- **Recommendations:** Timing-aware suggestions ("Due now," "Next week," "In 3 weeks") with consumer-friendly explanations ("Matches your shopping rhythm" instead of "co-accelerating Jacobian coupling")
- **Discovery:** "Customers like you also buy…" powered by rhythm-twin analysis

Both interfaces consume the same API endpoint and cadence engine; the difference is presentation layer only.

5. Data

The engine operates on real Instacart purchase data (preprocessed into weekly department-level frequencies per customer). The preprocessing pipeline (`preprocess.ts`)

converts raw order data into `CustomerCadence` objects with per-department weekly frequency arrays. Customer-identifiable information is not retained — only anonymized customer IDs with purchase frequency vectors.

6. Visualizations

6.1 CadenceResonance (Multi-Series Line Chart)

Time-series chart with interactive toggles per data series, interference indicators at regions of high |r|, and hover tooltips showing instantaneous correlation. Constructive regions highlighted in shared accent color; destructive in contrasting colors.

6.2 CausticResonance (WebGL Light-Interference Simulation)

Real-time WebGL shader simulating light caustics driven by data convergence. Each data series generates a wave; constructive interference produces sharp caustic lines, destructive produces diffuse light.

Sound synthesis via Web Audio API: pentatonic scale (A3 root), convergence → consonance, destructive → microtonal detuning, stereo spatial mapping.

Implementation constraints discovered during development: WebGL shaders require float-only uniforms, fully unrolled loops (no bool/int/break/continue), and separate canvas elements for WebGL vs. 2D fallback (cannot switch context type on same canvas).

6.3 CausticTimeline

Maps caustic intensity to a time axis, producing a heatmap where bright bands correspond to strong resonance periods.

6.4 CausticSphere

Extension mapping caustic patterns onto spherical geometry via horn torus topology. The Jacobian determinant of the surface mapping serves as an analytical attention map computed from first principles rather than learned weights.

The holographic volume model computes caustic density at every depth: `hologramDensity(u, v, z) = 1/|det(I + z·H·η)|` where H is the Hessian of curvature and η is the surface normal. This produces a 3D attention field.

7. Discussion

7.1 What Cadence Decomposition Is Not

Cadence decomposition is not Fourier analysis with a new name. The contribution is the *application*: frequency decomposition as a pre-modeling diagnostic across domains, and differential geometry (Jacobian/Hessian) as a recommendation engine operating on purchase cadences rather than collaborative filtering or content similarity.

7.2 Limitations

- **Non-stationary data:** Cadences shift. Windowed analysis introduces window-size sensitivity.
- **Confounders:** Temperature and retail sales may both respond to calendar seasonality rather than each other. Cadence resonance detects correlation, not causation.
- **Irregular sampling:** FFT assumes regular spacing; handling gaps requires interpolation or alternative decomposition methods.
- **Data scale:** The Instacart dataset provides sufficient signal for per-customer analysis. The engine has not been validated at enterprise scale (millions of customers).

7.3 Connection to Organizational Thermodynamics

Cadence provides the "temperature" measurement that the organizational thermodynamics framework (Gill & Ash, 2026) requires. Team activity cadence, measured from communication timestamps, may serve as thermodynamic temperature — a high-cadence team is "hotter" than a low-cadence one, with implications for decay rates and promotion thresholds in the chemical kinetics framework (Gill & Ash, 2026).

8. Conclusion

We have demonstrated that counting and time are sufficient to detect meaningful cross-domain structure ($r = -0.96$ holiday decorrelation) and to power a working recommendation engine (Jacobian-based cadence matching on real purchase data). The cadence matching engine, its analyst dashboard, and its consumer interface are deployed and operational. The approach requires no training data, no domain expertise, and no parameter tuning beyond frequency resolution.

References

- Gill, E. & Ash, K. (2026). Organizational Thermodynamics: Automated Attention from Communication Metadata. *In preparation.*
- Gill, E. & Ash, K. (2026). Chemical Kinetics as a Framework for Multi-Agent Memory Management. *In preparation.*
- Gill, E. & Ash, K. (2026). Embedding Trajectory Compression for Persistent Agent Memory. *Preprint.* DOI: 10.5281/zenodo.18778409
- Cooley, J.W. & Tukey, J.W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.* 19(90), 297–301.
- Lomb, N.R. (1976). Least-squares frequency analysis of unequally spaced data. *Astrophysics and Space Science*, 39(2), 447–462.