PhaseBridge: Strict Discrete ↔ Phase Mapping as a Foundation for Cognitive Systems

Anton Fedotov
SynqraTech
anvifedotov.biz@gmail.com
ORCID: 0009-0004-2313-6530

Sergey Efimov
SynqraTech
se.de.efimov@gmail.com

October 3, 2025

Abstract

Data pipelines increasingly span heterogeneous modalities (time series, images, tokens), storage formats, and processing stacks, which complicates reliable interchange and validation across systems. We present *PhaseBridge*, a minimal, strict, and lossless interpreter that maps discrete symbols to a continuous phase representation and back, enabling a uniform interchange layer with verifiable round-trip guarantees. The core mapping is $n \in \{0, \ldots, M-1\} \mapsto \theta = 2\pi n/M$ with nearest-grid decoding; phases are stored as IEEE-754 float64 and validated by a lightweight schema. We define the *Phase Interchange Format* (PIF), provide a reference SDK and CLI, and introduce a coherence metric κ as a read-only diagnostic signal available uniformly across modalities. We demonstrate feasibility on time series and 8-bit images, and outline a roadmap for additional adapters. The aim is not to replace domain-specific encodings, but to offer a small, dependable core that supports strict round-trip, integrity checks, and unified diagnostics where a common representational substrate benefits cognitive and data-analysis workflows.

1 Introduction

Modern data systems routinely integrate heterogeneous sources—discrete time series, tabular records, images, token streams—each carried by different on-wire formats (e.g., CSV/JSON/Parquet) and processed by modality-specific tooling. While these formats serve storage and transport well, they do not by themselves provide a *uniform representational substrate* that can support strict round-trip guarantees, lightweight runtime validation, and cross-modal diagnostics in a common space. As pipelines become more compositional and model-centric, the absence of such a substrate introduces friction: fragile adapters,

inconsistent semantics, and difficulty verifying that transformations are strictly reversible where required.

This work. We introduce *PhaseBridge*, a minimal lossless interpreter that maps between a discrete alphabet and a continuous phase space. The core mapping is defined by a fixed alphabet size $M \in \mathbb{N}$ and

$$n \in \{0, \ldots, M-1\} \quad \longleftrightarrow \quad \theta = \frac{2\pi}{M}\, n \in [0, 2\pi),$$

with decoding by nearest grid point,

$$\hat{n} \;=\; \mathrm{round}\!\left(\frac{M}{2\pi}\,\theta\right) \bmod M, \tag{2}$$

and phases stored in IEEE-754 float64 wrapped to $[0, 2\pi)$. On top of this mapping, we define a compact *Phase Interchange Format* (PIF) with a minimal schema (alphabet and optional metadata), runtime validation, and an integrity hook via SHA-256 of the original raw byte stream. We further expose a modality-agnostic diagnostic signal—the circular coherence metric $\kappa = \left|\frac{1}{N}\sum_j e^{i\theta_j}\right| \in [0,1]$ (and its windowed form)—which can be computed uniformly across modalities without altering data.

Motivation. The goal is a small, dependable core that can serve as a stable *interchange layer* in cognitive and data-analysis workflows: strict reversibility when needed; explicit separation of concerns between conversion and any subsequent processing; and a simple, uniform diagnostic available everywhere. This is conceptually akin to how UTF-8 provides a tiny, strict core for textual interchange: not a replacement for all domain encodings, but a common basis that reduces friction and ambiguity across systems.

Scope. PhaseBridge targets settings where (i) lossless round-trip is a functional requirement for the declared alphabet; (ii) integrity verification and lightweight runtime validation are valuable; and (iii) a uniform diagnostic signal aids monitoring and analysis. We demonstrate feasibility on time series and 8-bit grayscale images, and outline adapters for additional modalities. Domain-specific encodings and formats remain appropriate for their respective tasks; the proposed layer complements them by offering a strict, verifiable representational bridge.

Contributions.

- A strict, lossless discrete↔phase mapping with nearest-grid decoding and float64 storage, with discussion of numerical considerations.
- The *Phase Interchange Format* (PIF): a minimal schema, runtime validation, and integrity checks via `hash_raw`.

- A reference SDK and CLI (encode/decode/validate/κ), plus conformance tests.
- Uniform coherence diagnostics (κ and windowed κ) applicable across modalities without modifying data.
- Case studies on time series and images, and a roadmap for additional adapters.

2 Mathematical Foundations

Alphabet and mapping. Fix an alphabet size $M \in \mathbb{N}$, $M \geq 2$, and define the discrete alphabet $\mathcal{A}_M = \{0, 1, \ldots, M-1\}$. The core mapping is

$$f : \mathcal{A}_M \to [0, 2\pi), \qquad f(n) \;=\; \theta \;=\; \frac{2\pi}{M}\, n, \tag{3}$$

with decoding by nearest grid point on the uniform lattice $\{2\pi k/M\}_{k=0}^{M-1}$:

$$g : [0, 2\pi) \to \mathcal{A}_M, \qquad g(\theta) \;=\; \hat{n} \;=\; \mathrm{round}\!\left(\frac{M}{2\pi}\,\theta\right) \bmod M. \tag{4}$$

We also use the wrap operator $\mathrm{wrap}_{2\pi}(\cdot)$ mapping angles to $[0, 2\pi)$ by modulo 2π, with the convention that values numerically equal to 2π are set to 0.

Lossless round-trip (exact arithmetic). In exact arithmetic, $g(f(n)) = n$ for all $n \in \mathcal{A}_M$. Indeed, $f(n) = 2\pi n/M$ is exactly a lattice point; scaling by $M/(2\pi)$ yields n, and rounding to the nearest integer returns n (ties do not occur).

Invariances. Let $\Delta = 2\pi/M$ be the lattice step. For any $k \in \mathbb{Z}$, $\mathrm{wrap}_{2\pi}(\theta + k\Delta)$ decodes to $(n + k) \bmod M$. Moreover, adding multiples of 2π leaves decoding invariant: $g(\mathrm{wrap}_{2\pi}(\theta + 2\pi\ell)) = g(\theta)$ for all $\ell \in \mathbb{Z}$.

2.1 Finite-precision analysis (IEEE-754 float64)

In implementation, θ is stored as IEEE-754 float64. We analyze correctness when (3)–(4) are computed in binary64 with round-to-nearest.

We use the standard rounding model [2]: for a basic operation $\circ \in \{+, -, \times, /\}$, $\mathrm{fl}(a \circ b) = (a \circ b)(1 + \delta)$ with $|\delta| \leq u$, where $u = 2^{-53} \approx 1.11 \times 10^{-16}$ is machine epsilon (unit roundoff). For a short sequence of k such operations, the accumulated relative error is bounded by $\gamma_k = ku/(1 - ku)$.

Encoding error. Compute $\tilde{\theta} = \mathrm{fl}\Big(\frac{2\pi}{M}\, n\Big)$. For $0 \le n \le M-1$, $0 \le \theta \le 2\pi$. Assuming (2π) is a fixed binary64 constant, and one multiplication and one division,

$$|\tilde{\theta} - \theta| \;\le\; \gamma_2\, \theta \;+\; |\mathrm{fl}(2\pi) - 2\pi|\, \frac{n}{M} \;\le\; C\, u, \tag{5}$$

for a modest constant C (empirically $C < 10$ suffices on typical compilers; the absolute error is $\mathcal{O}(u)$ uniformly because $\theta \le 2\pi$).

Decoding correctness condition. Let $\Delta = 2\pi/M$. Decoding by (4) is correct provided the absolute error at decode input is less than $\Delta/2$:

$$\Big|\, \tilde{\theta} - \theta \,\Big| \;<\; \frac{\Delta}{2} \;=\; \frac{\pi}{M}. \tag{6}$$

Since $|\tilde{\theta} - \theta| \le Cu$ and $\pi/M \ge \pi/2^{32} \approx 7.3 \times 10^{-10}$ for $M \le 2^{32}$, while $Cu \lesssim 10^{-15}$, inequality (6) holds with a margin exceeding 10^5 for all $M \le 2^{32}$.

[Binary64 correctness for $M \le 2^{32}$] Let $\tilde{\theta} = \mathrm{wrap}_{2\pi}(\mathrm{fl}(2\pi n/M))$ be computed in IEEE-754 binary64 with round-to-nearest, and let decoding be $\hat{n} = \mathrm{round}(M\tilde{\theta}/(2\pi)) \bmod M$ computed in binary64. Then for all integers M with $2 \le M \le 2^{32}$ and for all $n \in \{0, \ldots, M-1\}$,

$$\hat{n} \;=\; n.$$

Proof sketch. By the rounding model, the absolute encoding error $|\tilde{\theta} - \theta|$ is bounded by Cu for a small constant C independent of M and n, since $\theta \le 2\pi$. The nearest-neighbor lattice spacing is $\Delta = 2\pi/M$, so the mid-point gap is $\Delta/2 = \pi/M$. For $M \le 2^{32}$, $\pi/M \gg Cu$; thus $\tilde{\theta}$ remains within the Voronoi cell of the true lattice point and rounding recovers n. The subsequent scaling by $M/(2\pi)$ preserves classification because it is monotone and computed with comparable (smaller) absolute error. □

Wrapping convention. We implement $\mathrm{wrap}_{2\pi}(\tilde{\theta}) = \tilde{\theta} \bmod (2\pi)$ and, for numerical stability, map values numerically equal to 2π (within machine epsilon) to 0 to maintain the half-open interval $[0, 2\pi)$.

2.2 Coherence metric κ

Given phases $\{\theta_j\}_{j=1}^{N} \subset [0, 2\pi)$, define the (unweighted) circular coherence

$$\kappa \;=\; \left|\, \frac{1}{N} \sum_{j=1}^{N} e^{\,i\theta_j} \,\right| \;\in\; [0, 1]. \tag{7}$$

For nonnegative weights $w_j \geq 0$ with $S = \sum_j w_j > 0$, the weighted version is

$$\kappa_w \;=\; \left| \frac{1}{S} \sum_{j=1}^{N} w_j \, e^{\,i\theta_j} \right|. \tag{8}$$

We also consider a windowed version along an index axis (e.g., time). For window size W and hop H, define windows $\mathcal{W}_k = \{\, t : kH \leq t \leq kH + W - 1 \,\}$ for $k = 0, \ldots, K-1$ where $K = \lfloor (N - W)/H \rfloor + 1$, centers $c_k = kH + (W-1)/2$, and

$$\kappa^{(k)} \;=\; \begin{cases} \left| \frac{1}{|\mathcal{W}_k|} \sum_{t \in \mathcal{W}_k} e^{\,i\theta_t} \right|, & \text{unweighted,} \\ \left| \frac{\sum_{t \in \mathcal{W}_k} w_t e^{\,i\theta_t}}{\sum_{t \in \mathcal{W}_k} w_t} \right|, & \text{weighted (if } w \text{ available).} \end{cases} \tag{9}$$

If $W > N$, we define a single window $\mathcal{W}_0 = \{0, \ldots, N-1\}$.

3 Phase Interchange Format (PIF)

PIF is a compact, minimally constrained object describing the phase representation and its decoding context. We model a PIF record as a tuple

$$p \;=\; (\texttt{schema},\, \theta,\, \texttt{amp},\, \texttt{meta}),$$

where $\theta \in \mathbb{R}^N$ (stored as float64), $\texttt{amp} \in \{\text{scalar } a \geq 0\} \cup \mathbb{R}^N_{\geq 0}$ (optional weighting), and the remaining fields are JSON objects with the following semantics.

3.1 Schema

The `schema` object declares at least the alphabet:

- `schema.alphabet.type` `"uint"` (required).
- `schema.alphabet.M` integer, $2 \leq M \leq 2^{32}$ (required).

Optional, non-normative fields may be included for downstream use (do not affect lossless conversion), e.g.,

- `schema.sampling.fs` number, > 0 (sampling frequency).
- `schema.image.{height,width,mode}` for image-shaped data.

PIF v1 core treats unspecified fields as opaque; producers may include additional metadata that does not alter the conversion semantics.

3.2 Numeric representation

- **Phases** θ: stored as IEEE-754 float64, wrapped to $[0, 2\pi)$, with the wrapping convention of §2.1.
- **Amplitude** `amp`: either a nonnegative scalar (interpreted as unweighted) or a nonnegative array of length N (used as weights for κ_w).

3.3 Meta and integrity

The `meta` object includes:

- `meta.note` string, matching `"no_processing"` or `"processed:<ops>"` (required).
- `meta.hash_raw` optional string of the form `"sha256:<64 hex>"`: the SHA-256 of the original raw byte stream (e.g., the discrete input array).
- `meta.codec` optional string naming the codec (e.g., `S1_phase_code_M256`).
- `meta.codec_hash` optional string `"sha256:<64 hex>"` identifying the codec payload.

When `meta.note` is `"no_processing"`, the round-trip invariant *must* hold for the declared M:

$$\text{decode}(\text{encode}(x)) \;=\; x \quad \text{(bit-exact for the alphabet)}.$$

Consumers can verify integrity by decoding to x and comparing `meta.hash_raw` to sha256(x.bytes).

3.4 Serialization and validation

Normative on-wire format. PIF v1 uses JSON as the normative on-wire representation (numbers as JSON numbers corresponding to float64 in implementations). A MessagePack representation may be provided as an *optional, byte-compact mirror*; it must be semantically equivalent to the JSON form.

Validation. Validation proceeds in two layers:

1. **Schema check** (optional but recommended): JSON Schema validation of the document shape and basic constraints (e.g., `alphabet.type="uint"`, bounds on M, nonnegativity of `amp`).
2. **Runtime check**: load into a PIF object with strict checks on numeric invariants (float64 phases wrapped to $[0, 2\pi)$, matching lengths of θ and array `amp`, etc.).

If `meta.hash_raw` is present, consumers should verify it after decode.

Determinism. Given fixed `schema.alphabet.M` and `meta.note="no_processing"`, the encode/decode operations are deterministic; by Lemma 2.1, binary64 computations recover the original symbols for all $M \leq 2^{32}$.

4 Implementation

We release a reference implementation at:

`https://github.com/synqratech/phasebridge`

The repository provides a Python SDK, CLI tools, schemas, examples, and conformance tests.

4.1 SDK (Python)

The SDK exposes the strict discrete↔phase codec and core utilities:

- `S1PhaseCodec`: lossless mapping $n \leftrightarrow \theta$ with nearest-grid decoding; float64 storage.
- `PIF`: a dataclass for Phase Interchange Format records with runtime validation.
- `kappa_timeseries`, `kappa_timeseries_windowed`: coherence metrics (§2.2).
- Utilities: SHA-256 helpers (`hash_raw`), dtype selection, phase wrapping.

A minimal usage example:

```
from phasebridge.codec_s1 import S1PhaseCodec
from phasebridge.pif import PIF
from phasebridge.kappa import kappa_timeseries

codec = S1PhaseCodec(M=256)
x = ...  # uint array in [0..M-1]
schema = {"alphabet": {"type": "uint", "M": 256}}
p = codec.encode(x, schema)      # -> PIF (theta float64)
x_rec = codec.decode(p)          # -> original symbols (lossless)
k = kappa_timeseries(p)          # -> coherence in [0,1]
```

4.2 CLI tools

Command-line utilities support encode/decode/validate and diagnostics:

- `pb-encode`: raw (`bin/csv/npy`) → PIF (`json/msgpack`).

- `pb-decode`: PIF → raw (`bin/csv/npy`).
- `pb-validate`: schema/runtime validation, decode, hash checks; optional raw comparison.
- `pb-kappa`: compute global or windowed κ.

Example:

```
# Encode CSV(uint) -> PIF(JSON)
pb-encode --in series.csv --in-fmt csv --dtype uint8 --M 256 > series.pif.json
# Validate + compare with raw
pb-validate --in series.pif.json --raw series.csv --in-fmt csv --dtype uint8 --report json
#  (windowed)
pb-kappa --in series.pif.json --win 256 --hop 128 --fmt csv > kappa_windowed.csv
```

4.3 Testing and conformance

The test suite covers:

- Strict round-trip invariants for declared alphabets (§2).
- κ bounds and windowed profiling on synthetic signals.
- Schema/runtime validation (`PIF` loading, wrapping, type checks).
- Property-based tests (Hypothesis) across M and random arrays.

Tests can be reproduced via `pytest`; property-based tests are enabled when `hypothesis` is installed.

5 Experiments and Case Studies

We present three compact case studies illustrating lossless round-trip, integrity checks, and uniform diagnostics across modalities. Reproduction scripts are provided in `examples/` and notebooks in `demo/`.

5.1 Time series round-trip and windowed coherence

We synthesize a discrete time series in $[0, M-1]$, encode to PIF, decode to CSV, and verify bit-exact equality and `hash_raw` consistency. We also compute windowed κ to illustrate coherence as a simple health indicator.

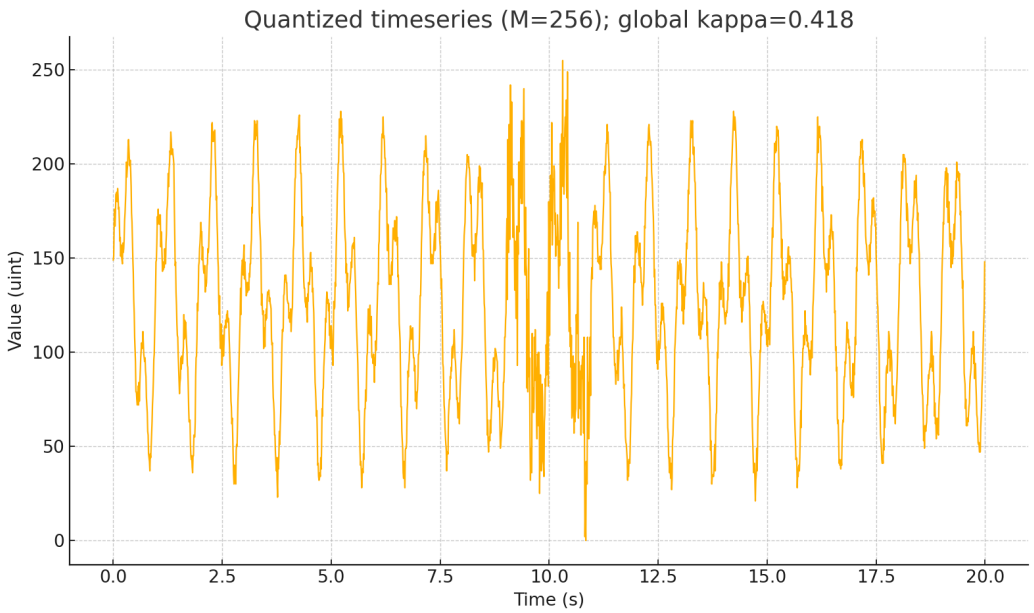

Figure 1: Discrete (quantized) time series used in the round-trip experiment. Left: signal overview; Right: zoomed segment.

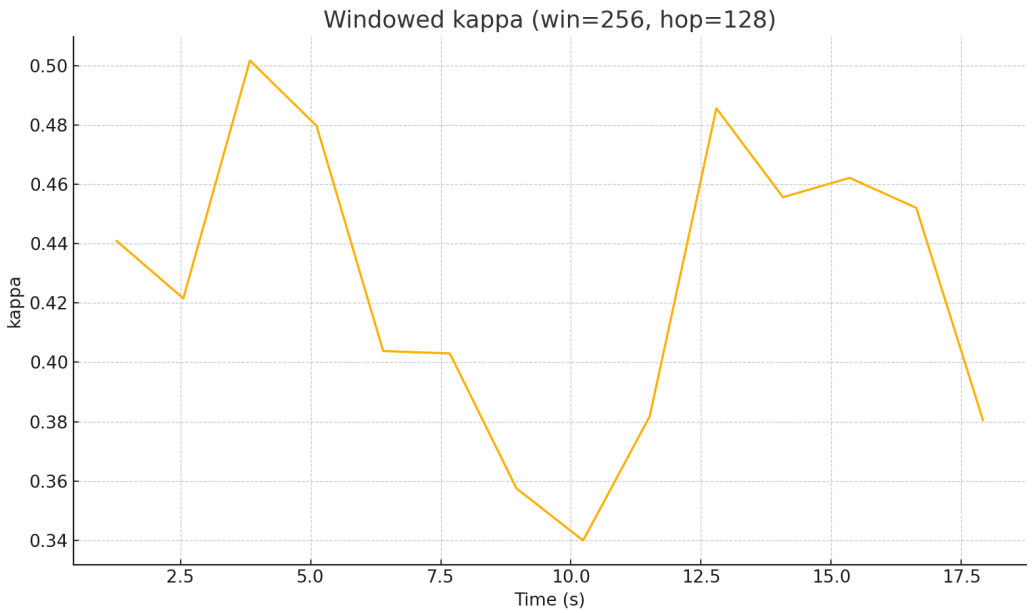

Figure 2: Windowed coherence κ along time. Periods of higher regularity yield larger κ; injected noise reduces κ.

Outcome. Decoding recovers the exact discrete series; `meta.hash_raw` matches sha256 of the decoded array. Windowed κ tracks local regularity without modifying data.

5.2 Image round-trip (8-bit grayscale)

We convert an 8-bit grayscale image to PIF and back to PNG. The phase map visualizes θ (normalized colormap), while the reconstruction verifies strict equality and hash integrity.

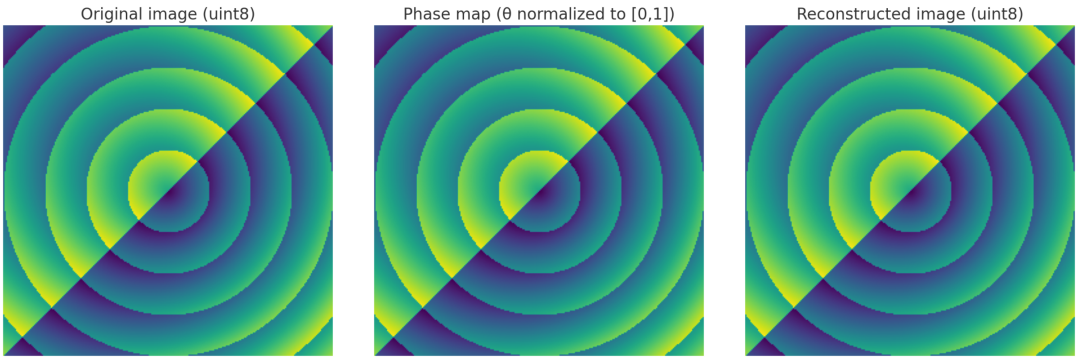

Figure 3: Grayscale image round-trip. Left: original; Middle: phase map θ; Right: reconstruction (bit-exact). The absolute difference image is identically zero.

Outcome. The reconstructed image matches the original bit-for-bit; `hash_raw` confirms integrity. The phase map provides an interpretable visualization of symbol placement on the unit circle.

5.3 Cross-modality illustration

We apply the same pipeline to time series and images, emphasizing that while on-disk encodings differ, the PIF representation is uniform (float64 phases plus minimal schema). The same diagnostic κ applies unchanged.

Modality	Round-trip equality	Hash check	Diagnostics
Time series (uint)	Exact	Matches (`hash_raw`)	κ, windowed κ
Image (8-bit L)	Exact	Matches (`hash_raw`)	κ over rasterized phases

Table 1: Cross-modality summary: distinct inputs, uniform PIF representation and diagnostics.

Reproducibility. We provide runnable scripts:

- `examples/timeseries_roundtrip.py`: CSV(uint) $\leftrightarrow$ PIF; strict equality and κ report.
- `examples/image_roundtrip.py`: 8-bit grayscale $\leftrightarrow$ PIF; strict equality and κ report.

Generated figures (phase maps, windowed κ) are included in the artifact bundle; see repository instructions for exact commands and seeds.

6 Discussion

Relation to JSON/Parquet and data formats. Conventional container formats (e.g., JSON, Parquet, CSV) specify *structure*, typing, and efficient storage/transport. They are well-suited for interchange at the schema level but do not provide a *uniform representational substrate* on which strict round-trip, cross-modal diagnostics, or simple algebraic operations can be carried out in a common space. PhaseBridge is orthogonal: it defines a minimal, reversible *representation* (discrete $\leftrightarrow$ phase) that can be serialized via JSON (normative) or MessagePack (optional) while preserving a common phase geometry for downstream analysis. In short, PIF complements—rather than replaces—existing container formats by providing a compact, verifiable representational layer.

Relation to ML embeddings. Latent embeddings in $\mathbb{R}^d$ (e.g., learned by neural networks) are powerful but generally lack strict reversibility to original symbols, and they vary across architectures, training runs, and objectives. PhaseBridge addresses a different need: a *lossless*, model-agnostic mapping tied to an explicit alphabet and a fixed lattice in $[0, 2\pi)$. Where embeddings aim to capture task-oriented semantics, the discrete$\leftrightarrow$phase mapping aims to provide a dependable, verifiable substrate with a simple diagnostic (κ) and a precise inverse. These approaches are complementary: phase representations can serve as a stable interchange/diagnostic layer, while embeddings can be layered on top for task-specific inference.

Potential and opportunities.

- **Phase-encoded latent spaces.** Phase-coordinate parameterizations (per-dimension angles on a torus $\mathbb{T}^d$) enable circular statistics, concentration analysis (e.g., von Mises), and coherence-based regularization, offering new handles for interpretability and monitoring.
- **Universal cognitive operability.** A common phase space allows simple, modality-agnostic operations (e.g., circular shifts, windowed aggregation, coherence profiling) and uniform health checks in pipelines where strict reversibility is a requirement.
- **Simple, read-only diagnostics.** The coherence metric κ provides a lightweight indicator of order/disorder that can be computed uniformly across modalities without altering data, aiding monitoring, alerting, and QA.

Limitations and scope. PhaseBridge targets settings where a strict, verifiable interchange layer is beneficial; it does not aim to supersede domain encodings optimized for compression, high-throughput analytics, or task-specific semantics. Its numeric guarantees rely on IEEE-754 binary64 (§2.1) and a declared alphabet bound $M \leq 2^{32}$. Extensions

beyond this scope (e.g., mixed-precision storage, lossy processing pipelines) require explicit labeling and are out of the core.

7 Related Work and Comparison

We contrast PhaseBridge with (i) lossless containers and encodings (bit packing, entropy coding, CBOR/Parquet with checksums), (ii) domain-specific reversible transforms (e.g., image/audio codecs), and (iii) learned representations (embeddings).

Lossless containers and encodings. Bit packing and entropy coders (e.g., DEFLATE, ANS families) achieve efficient storage and transport and can be layered under structured containers such as CBOR or Parquet, often with integrity mechanisms (checksums, column/page CRCs). These approaches optimize *compression and layout*, and Parquet/CBOR additionally specify structural typing and schema evolution. They do not, however, define a *common representational geometry* across modalities, nor do they provide a simple uniform diagnostic signal at the symbol level. PhaseBridge is orthogonal: it defines a reversible mapping to a phase space with float64 θ and a modality-agnostic coherence diagnostic κ, and can be serialized via JSON (normative) or mirrored in compact forms (e.g., MessagePack). In practice, PIF may be *embedded* inside conventional containers when columnar analytics or storage tiering are required.

Domain-specific reversible transforms. Lossless media codecs (e.g., PNG for images, FLAC for audio) and related reversible transforms target particular data domains, leveraging structure for compression or fidelity guarantees. They typically do not expose a cross-modal representational substrate nor a unified symbol-level diagnostic. PhaseBridge does not compete with domain codecs on compression or domain semantics; it provides a small, verifiable interchange layer that can sit *beside* such codecs where strict round-trip, uniform diagnostics, and explicit separation between conversion and processing are desirable. Domain-specific reversible methods (including bespoke pipelines akin to "RAZOR"-style reversible transforms) can still be used upstream/downstream of PIF.

Learned embeddings and model-centric pipelines. Learned latent vectors in $\mathbb{R}^d$ are task-oriented and powerful but typically lack strict reversibility to original symbols and depend on training objectives. PhaseBridge addresses a different need: a model-agnostic, strictly invertible mapping tied to an explicit alphabet and lattice in $[0, 2\pi)$, with a simple diagnostic (κ) that can be computed without modifying data. These approaches are complementary: embeddings can be layered on top of the phase substrate for semantics, while PIF ensures dependable interchange and verification.

When PhaseBridge is preferable

- A strict, verifiable round-trip for a declared alphabet is a *functional requirement*.
- Cross-modal pipelines benefit from a *uniform* representational substrate and a read-only diagnostic (κ).
- Explicit separation of concerns is required: conversion must be lossless/idempotent; any processing must be labeled.
- Lightweight runtime validation and integrity hooks (`hash_raw`) are desirable at interchange points.

When conventional methods are preferable

- The primary goal is *compression ratio* or *columnar analytics*; Parquet/CBOR+checksums or domain codecs are a better fit.
- Rich domain semantics and task-specific features are needed (learned embeddings, feature stores).
- Alphabets or numeric ranges fall outside the current guarantees (e.g., $M \gg 2^{32}$ or mixed-precision θ without explicit analysis).
- End-to-end workflows already standardize on a container/analytic stack and do not need a representational substrate.

Summary. PhaseBridge complements containers and codecs by supplying a minimal, reversible *representational* layer with uniform diagnostics. It is not a replacement for compression- or analytics-oriented formats; rather, it reduces friction where strict round-trip and cross-modal consistency are required, and can be embedded alongside existing infrastructure.

8 Roadmap and Future Work

Multimodal adapters. We plan adapters beyond time series and grayscale images, including:

- **RGB images.** Two strictly reversible strategies: 24-bit packing per pixel ($M = 256^3$) or per-channel mapping ($M = 256$ with $3\times$ phases).
- **Text tokens / tables.** Fixed vocabularies and categorical columns naturally map to discrete alphabets.
- **Audio.** Quantized waveforms and symbolic representations (e.g., event tokens) provide discrete inputs for PIF-based interchange and monitoring.

Operational layer in phase space. We aim to expose basic, modality-agnostic operators that act directly on phases: circular shifts, windowed reductions, coherence-based filters, and sidecar services that compute diagnostics on streams without mutating data. This supports deployment in production pipelines with explicit separation between conversion and processing.

Integrations.

- **Streaming systems.** Source/sink operators for Kafka/Flink that convert to/from PIF at ingress/egress while emitting κ for monitoring.
- **Sidecar services.** HTTP/gRPC endpoints for encode/decode/validate and κ computation, suitable for polyglot stacks.
- **Language bindings.** Reference implementations in Go/C++/Rust to broaden adoption and enable zero-copy integrations.

Research directions.

- **Explainability via phase-encoded embeddings.** Studying toroidal latent parameterizations with circular statistics and coherence regularizers for monitoring and interpretation.
- **Numerical extensions.** Analysis of mixed-precision storage (e.g., float32 θ) and robust decoding bounds; adaptive wrapping and error control.
- **Conformance and benchmarks.** Public datasets and tasks for round-trip validation, κ profiling, and cross-modality stress tests.

9 Conclusion

We introduced *PhaseBridge*, a small, strict, and lossless discrete↔phase mapping with a minimal interchange format (PIF), a reference SDK/CLI, and a uniform coherence diagnostic κ applicable across modalities. The core is intentionally narrow: a fixed-alphabet lattice in $[0, 2\pi)$ with nearest-grid decoding, float64 storage, and verifiable integrity hooks. This provides a dependable representational substrate—conceptually akin to how UTF-8 serves text—that complements existing container formats rather than replacing them. By enforcing round-trip guarantees, separating conversion from processing, and exposing lightweight diagnostics, PhaseBridge reduces friction in cognitive and data-analysis workflows and opens a path toward phase-space operators and multimodal adapters.

References

[1] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 2000.

[2] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2nd edition, 2002.

[3] J.-P. Lachaux, E. Rodriguez, J. Martinerie, and F. J. Varela. Measuring phase synchrony in brain signals. *Human Brain Mapping*, 8(4):194–208, 1999.

[4] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1995.

[5] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, IETF, 2017. `https://www.rfc-editor.org/rfc/rfc8259`

[6] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 3629, IETF, 2003. `https://www.rfc-editor.org/rfc/rfc3629`

[7] Apache Parquet. Columnar storage format (project documentation). `https://parquet.apache.org/` (accessed 2025).

[8] K. Claessen and J. Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *Proc. ICFP*, pages 268–279. ACM, 2000.