Supplementary Material:
The `QBitwave` Framework

IaM^e

2024

1 Purpose and Scope

This document provides a precise reference specification for the Python class `QBitwave`, used in the associated manuscripts. It formalizes the mapping from discrete bitstrings to emergent complex amplitudes and describes derived quantities such as Shannon entropy, compressibility, and coherence.

2 Preliminaries

Let:

- $b = (b_1, \ldots, b_N) \in \{0,1\}^N$ denote a bitstring of length N.
- $B = \{0,1\}^N$ denote the set of all bitstrings of length N.
- n_{block} denote the block size for bit-to-amplitude conversion (even integer).
- $\mathbb{C}$ denote the complex numbers.
- $\|\cdot\|_2$ denote the Euclidean norm.

3 Forward Mapping: Wavefunction to Bitstring

Given normalized complex amplitudes $\psi = (\psi_1, \ldots, \psi_M) \in \mathbb{C}^M$ and associated phases $\phi = (\phi_1, \ldots, \phi_M)$:

$$\psi_j = r_j e^{i\phi_j}, \quad r_j = |\psi_j|, \quad j = 1, \ldots, M, \tag{1}$$

$$b^j_{\text{amp}} = \text{binary\_encode}(r_j, k_{\text{amp}}), \tag{2}$$

$$b^j_{\text{phase}} = \text{binary\_encode}\left(\frac{\phi_j}{2\pi}, k_{\text{phase}}\right), \tag{3}$$

where $k_{\text{amp}}, k_{\text{phase}}$ are user-selected bits per amplitude/phase. The flattened bitstring is

$$b = \bigoplus_{j=1}^{M} \left(b^j_{\text{amp}} \,||\, b^j_{\text{phase}} \right),$$

with $||$ denoting concatenation.

4 Reverse Mapping: Bitstring to Wavefunction

Partition the input bitstring b into consecutive blocks of size n_{block}:

$$b = (b_1, \ldots, b_N), \quad n_{\text{block}} \mid N, \tag{4}$$

$$n_{\text{blocks}} = N / n_{\text{block}}. \tag{5}$$

Each block is split into two halves (real and imaginary parts):

$$\text{Re}(\psi_j) = \text{bits\_to\_signed\_float}\left(b_{(j-1)n_{\text{block}}+1 : jn_{\text{block}}/2} \right), \tag{6}$$

$$\text{Im}(\psi_j) = \text{bits\_to\_signed\_float}\left(b_{jn_{\text{block}}/2+1 : jn_{\text{block}}} \right), \tag{7}$$

and then normalized:

$$\psi = \frac{\psi}{\|\psi\|_2}.$$

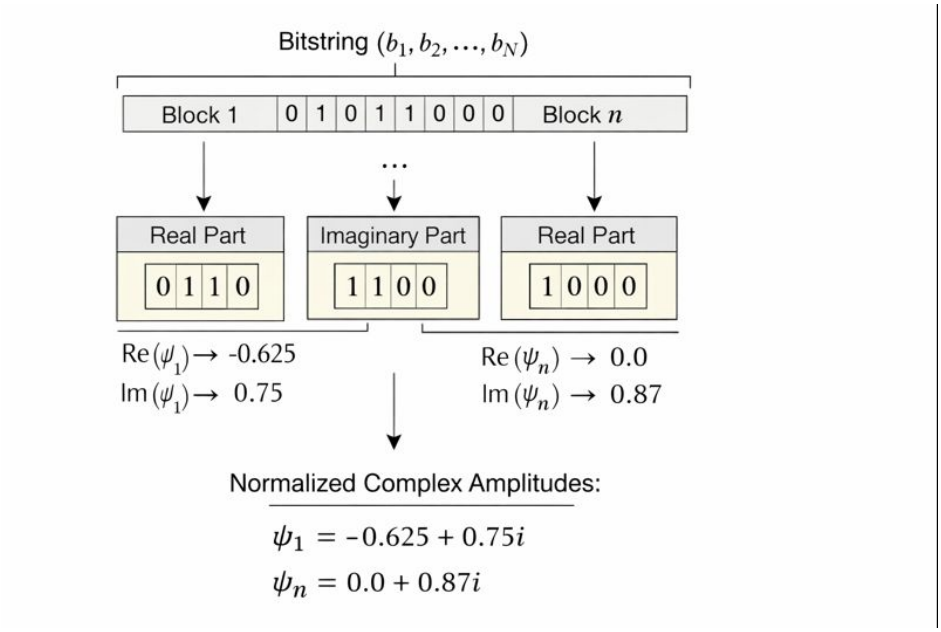

Figure 1: The structure of QBitwave

5 Derived Quantities

5.1 Amplitude Probabilities

$$p_j = |\psi_j|^2, \quad \sum_{j=1}^{M} p_j = 1.$$

5.2 Shannon Entropy of Wavefunction

$$S_\psi = -\sum_{j=1}^{M} p_j \log_2 p_j.$$

5.3 Bitstring Entropy (Syntactic Entropy)

Let p_0, p_1 be the fractions of zeros and ones in b:

$$H_b = -p_0 \log_2 p_0 - p_1 \log_2 p_1.$$

5.4 Coherence

The Kullback–Leibler divergence between bit-level and amplitude-level distributions:

$$D_{\mathrm{KL}}(P_b \parallel P_\psi) = \sum_{i=0,1} P_b(i) \log_2 \frac{P_b(i)}{P_\psi(i)},$$

where P_ψ is truncated/aligned to the same length as P_b.

5.5 Compressibility

Define the Fourier transform of amplitudes:

$$\hat{\psi} = \mathrm{FFT}(\psi),$$

and let N_{sig} be the number of Fourier coefficients with magnitude above threshold τ. Then

$$C = 1 - \frac{N_{\mathrm{sig}}}{M}, \quad C \in [0, 1].$$

6 Randomization / Mutation Operations

Small perturbations of the amplitudes $\psi \to \psi + \eta$, with $\eta \sim \mathcal{CN}(0, \sigma^2)$ (complex Gaussian), are normalized to unit norm. Random bit flips on b trigger automatic recomputation of ψ.

7 API Summary (Illustrative)

```
from qbitwave import QBitwave

bw = QBitwave(bitstring)
amps = bw.get_amplitudes()
entropy = bw.entropy()
bit_entropy = bw.bit_entropy()
coherence = bw.coherence()
compress = bw.compressibility()
bw.mutate(level=0.01)
bw.flip(n_flips=5)
```

8 Relation to Theory in Main Text

The following properties are essential to the analytical results:

- Normalization of amplitudes
- Mapping from bitstring to complex amplitudes
- Computation of Shannon entropy and compressibility

All other methods (e.g., random flips, mutation) are ancillary and used only in simulation studies.