Prime Gap Structure via Exceptional Lattice Projections: E8, F4, and the Crystalline Grid of 2×10^6 Primes

John Janik

February 21, 2026

Abstract

We construct a deterministic map from the sequence of prime gaps to the E_8 root system, and study its restriction to the F_4 sublattice. Given consecutive primes $p_n < p_{n+1}$, the normalized gap $\tilde{g}_n = (p_{n+1} - p_n)/\log p_n$ is mapped to one of the 240 roots of E_8 via a phase function derived from the lattice's minimal norm $\sqrt{2}$. We project E_8 roots onto the F_4 sublattice (48 roots) using cosine similarity and show that *all 240* E_8 roots pass the F_4 quality threshold—a structural fact we prove by case analysis on root types. The Jordan trace from the Albert algebra $J_3(\mathbb{O})$ provides a coloring of the Ulam spiral, and an F4 Exceptional Fourier Transform (F4-EFT) with Salem–Jordan filtering extracts *crystalline vertices*: discrete points of maximal F_4 resonance.

In addition, we implement twelve independent decoding methods that attempt to extract structured information from the prime gap sequence via $E_8/2E_8$ Hamming extraction, sign bits, parity bits, F_4 root indices, Jordan trace classification, and crystalline vertex properties. All methods are evaluated on 2×10^6 primes with entropy, chi-square, and longest-printable-run metrics.

All algorithms are given in full detail with explicit formulas. The analysis is implemented in Python, C/OpenMP (self-contained PPM renderer), and a comprehensive multi-method decoder; source code is publicly available.

Contents

1 Introduction **4**
1.1 Notation 4

2 The E8 Root System and Prime Gap Assignment **4**
2.1 The E8 Root System 4
2.2 Projection Slope 5
2.3 The Root Assignment Map 5
2.4 Ulam Spiral Coordinates 6

3 The F4 Root System and Jordan Trace **6**
3.1 The F4 Root System 6
3.2 E8 to F4 Projection 6
3.3 The Albert Algebra and Jordan Trace 7

3.4 Idempotent Classification 8
3.5 F4 Character Weights 9
3.6 Plasma Colormap for F4 Grid 9

4 The F4 Exceptional Fourier Transform **9**
4.1 Definition 9
4.2 Power Spectrum and Jordan Decomposition 10
4.3 Phase-Lock Analysis 10
4.4 The Salem–Jordan Kernel 10
4.5 Crystalline Vertex Extraction 10

5 E8 Lattice Decoding **11**
5.1 Gap Embedding into $\mathbb{R}^8$ 11
5.2 Closest Vector Decoding 11
5.3 Bit Extraction via $E_8/2E_8$ 12

6 Multi-Method Decoder **12**
6.1 Group 1: E8 Lattice Decoding 12
6.2 Group 2: E8 Root Assignment 12
6.3 Group 3: Raw Gap Properties 13
6.4 Group 4: F4 Sub-harmonic 13
6.5 Group 5: Crystalline Vertices 13
6.6 Statistical Evaluation 13

7 Computational Implementation **13**
7.1 Prime Data 13
7.2 Self-Contained C Implementation 14
7.3 Python F4 Analysis Pipeline 14
7.4 Multi-Method Decoder Implementation 15
7.5 Build System 15

8 Results **15**
8.1 F4 Crystalline Grid ($N = 2{,}000{,}000$ primes) 15
8.2 Multi-Method Decoder Results ($N = 100{,}000$ primes) 16
8.3 Analysis of Decoder Results 16
8.4 Performance 17

9 Octonion Arithmetic **17**
9.1 Multiplication Table 17
9.2 Albert Algebra Product 18
9.3 Cayley Plane Projection 18

10 Discussion **18**
10.1 What the Patterns Are 18
10.2 What the Patterns Are Not 19
10.3 The Complete F4 Coverage Phenomenon 19
10.4 Reproducibility 19

11 Conclusion **20**

A Complete E8 Root Enumeration **20**

B Complete F4 Root Enumeration **20**

C Plasma Colormap Table **20**

D Decoder Method Cross-Reference **21**

1 Introduction

The distribution of prime numbers is one of the oldest problems in mathematics. While the Prime Number Theorem establishes that $\pi(x) \sim x/\log x$ as $x \to \infty$, the fine-scale behavior of individual prime gaps $g_n = p_{n+1} - p_n$ remains largely mysterious. The Cramér model [4] predicts that g_n fluctuates around $\log p_n$, but the actual distribution displays structure that goes beyond simple randomness: twin primes cluster, Polignac gaps recur, and the Ulam spiral [16] reveals diagonal alignments corresponding to prime-producing quadratic polynomials.

In this paper, we introduce a framework for analyzing prime gap structure using the exceptional Lie lattices E_8 and F_4. Our approach is organized in four layers:

(i) **E8 Root Assignment** (section 2): Each normalized prime gap $\tilde{g}_n$ is deterministically assigned to one of the 240 roots of E_8. The *projection slope* of the assigned root induces a coloring of the Ulam spiral that displays concentric ring structure.

(ii) **F4 Sublattice Filtering** (section 3): The E8 roots are projected onto the 48-root F_4 sublattice via cosine similarity, and the *Jordan trace* from the Albert algebra $J_3(\mathbb{O})$ provides a secondary coloring. We prove that all 240 E_8 roots achieve F_4 projection quality ≥ 0.7, yielding 100% F_4 coverage.

(iii) **Spectral Analysis** (section 4): An *F4 Exceptional Fourier Transform* (F4-EFT) decomposes the gap signal into 48 spectral components, and a *Salem–Jordan kernel* filters the spectrum to extract crystalline vertices with idempotent boosting.

(iv) **Multi-Method Decoding** (section 5): Twelve independent extraction methods attempt to recover structured information from the prime gap data, evaluated with statistical tests for non-randomness.

We emphasize that the patterns we observe are consequences of the specific mathematical construction applied to prime gap statistics; they do not constitute a proof of any conjecture about prime distribution. The purpose of this paper is to describe the construction in full reproducible detail, report the computational results, and discuss what the observed patterns do and do not imply.

1.1 Notation

Throughout, p_n denotes the n-th prime ($p_1 = 2, p_2 = 3, \ldots$). We write $g_n = p_{n+1} - p_n$ for the n-th prime gap and $\tilde{g}_n = g_n/\log p_n$ for the normalized gap. By the Prime Number Theorem, $\tilde{g}_n$ has mean asymptotically equal to 1. We use $\|\cdot\|$ for the Euclidean norm in $\mathbb{R}^d$.

2 The E8 Root System and Prime Gap Assignment

2.1 The E8 Root System

The E_8 root system is the unique rank-8 even unimodular lattice. It has 240 roots (vectors of minimal norm $\sqrt{2}$), which fall into two classes [3, 8]:

Definition 2.1 (E8 roots)**.** The 240 roots of E_8 in $\mathbb{R}^8$ consist of:

(a) **Type I** (112 roots): $\pm e_i \pm e_j$ for $1 \leq i < j \leq 8$, where e_i is the i-th standard basis vector.

(b) **Type II** (128 roots): $\frac{1}{2}(\pm 1, \pm 1, \pm 1, \pm 1, \pm 1, \pm 1, \pm 1, \pm 1)$ where the number of negative signs is even.

Proposition 2.2. *All 240 roots have Euclidean norm $\sqrt{2}$.*

Proof. For Type I: $\|\pm e_i \pm e_j\|^2 = 1 + 1 = 2$. For Type II: $\|(\pm\frac{1}{2})^8\|^2 = 8 \cdot \frac{1}{4} = 2$. □

Remark 2.3 (Enumeration algorithm)**.** For Type I, iterate over all $\binom{8}{2} = 28$ pairs (i, j) and all 4 sign combinations $(s_1, s_2) \in \{-1, +1\}^2$, yielding $28 \times 4 = 112$ roots. For Type II, iterate over all $2^8 = 256$ sign masks; for bit k of the mask, set coordinate k to $+\frac{1}{2}$ if the bit is 1, $-\frac{1}{2}$ otherwise. Keep only masks where the number of 0-bits (negative signs) is even; by parity, exactly 128 survive. Total: $112 + 128 = 240$.

2.2 Projection Slope

We partition the eight coordinates of $\mathbb{R}^8$ into two groups of four: the "base" coordinates (r_1, r_2, r_3, r_4) and the "fiber" coordinates (r_5, r_6, r_7, r_8).

Definition 2.4 (Projection slope)**.** For an E_8 root $\alpha = (r_1, \ldots, r_8) \in \mathbb{R}^8$, define:

$$x(\alpha) = r_1 + r_2 + r_3 + r_4, \tag{1}$$
$$y(\alpha) = r_5 + r_6 + r_7 + r_8. \tag{2}$$

The *projection slope* is:

$$s(\alpha) = \begin{cases} y(\alpha)/x(\alpha) & \text{if } |x(\alpha)| > 0.01, \\ \operatorname{sgn}(y(\alpha)) \cdot 10 & \text{otherwise.} \end{cases} \tag{3}$$

2.3 The Root Assignment Map

Given a normalized prime gap $\tilde{g}_n \geq 0$, we assign an E_8 root index as follows.

Definition 2.5 (E8 root assignment)**.** Fix an enumeration $\alpha_0, \alpha_1, \ldots, \alpha_{239}$ of the 240 E_8 roots (in the order specified by definition 2.1). Define:

$$t(\tilde{g}) = \sqrt{\max(\tilde{g},\, 0.01)}, \tag{4}$$
$$\varphi(\tilde{g}) = \frac{t(\tilde{g})}{\sqrt{2}} \bmod 1, \tag{5}$$
$$\iota(\tilde{g}) = \lfloor 240 \cdot \varphi(\tilde{g}) \rfloor \bmod 240. \tag{6}$$

The assigned root is $\alpha_{\iota(\tilde{g})}$.

Proposition 2.6. *The root assignment is periodic: $\iota(\tilde{g}) = \iota(\tilde{g}')$ whenever $\sqrt{\tilde{g}} - \sqrt{\tilde{g}'} = k\sqrt{2}$ for some integer k.*

Proof. If $\sqrt{\tilde{g}} = \sqrt{\tilde{g}'} + k\sqrt{2}$, then $\varphi(\tilde{g}) = (\sqrt{\tilde{g}}/\sqrt{2}) \bmod 1 = (\sqrt{\tilde{g}'}/\sqrt{2} + k) \bmod 1 = \varphi(\tilde{g}')$. □

2.4 Ulam Spiral Coordinates

The Ulam spiral [16] arranges natural numbers on $\mathbb{Z}^2$ by spiraling outward from the origin.

Definition 2.7 (Ulam coordinates)**.** For $p \geq 1$, let $k = \lceil (\sqrt{p} - 1)/2 \rceil$, let $t = 2k + 1$, and let $m = t^2$. Set $t \leftarrow t - 1 = 2k$. Then:

$$(u_x(p),\, u_y(p)) = \begin{cases} (k - (m - p),\ -k) & \text{if } p \geq m - t, \\ (-k,\ -k + (m - t - p)) & \text{if } p \geq m - 2t, \\ (-k + (m - 2t - p),\ k) & \text{if } p \geq m - 3t, \\ (k,\ k - (m - 3t - p)) & \text{otherwise.} \end{cases} \tag{7}$$

Each prime is mapped in $O(1)$ time, making the computation embarrassingly parallel.

3 The F4 Root System and Jordan Trace

3.1 The F4 Root System

The F_4 root system has rank 4 and 48 roots. It is the automorphism group of the exceptional Jordan algebra $J_3(\mathbb{O})$ [6, 15]. The Dynkin diagram is:

$$\underset{\alpha_1}{\circ} - \underset{\alpha_2}{\circ} \Longrightarrow \underset{\alpha_3}{\circ} - \underset{\alpha_4}{\circ} \tag{8}$$

The Cartan matrix is:

$$A_{F_4} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -2 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}. \tag{9}$$

Definition 3.1 (F4 roots in $\mathbb{R}^4$)**.** The 48 roots of F_4 consist of:

(a) **24 long roots** (norm $\sqrt{2}$): $\pm e_i \pm e_j$ for $1 \leq i < j \leq 4$. ($\binom{4}{2} \times 4 = 24$ roots.)

(b) **8 short roots of type A** (norm 1): $\pm e_i$ for $1 \leq i \leq 4$.

(c) **8 short roots of type B** (norm 1): $\frac{1}{2}(\pm 1, \pm 1, \pm 1, \pm 1)$ with an even number of negative signs.

(d) **8 short roots of type C** (norm 1): $\frac{1}{2}(\pm 1, \pm 1, \pm 1, \pm 1)$ with an odd number of negative signs.

Total: $24 + 8 + 8 + 8 = 48$.

3.2 E8 to F4 Projection

We project E_8 roots onto the F_4 sublattice via the first four coordinates.

Definition 3.2 (E8 to F4 mapping)**.** For an E_8 root $\alpha = (r_1, \ldots, r_8)$:

1. Compute the projection $\pi(\alpha) = (r_1, r_2, r_3, r_4)$.
2. If $\|\pi(\alpha)\| < 0.01$, use the fallback $\pi(\alpha) = (r_5, r_6, r_7, r_8)$.

3. Normalize: $\hat{\pi} = \pi(\alpha)/\|\pi(\alpha)\|$.

4. For each F_4 root β_j ($j = 0, \dots, 47$), compute the cosine similarity:

$$\cos\theta_j = \frac{|\langle \hat{\pi}, \hat{\beta}_j \rangle|}{\|\hat{\pi}\| \cdot \|\hat{\beta}_j\|}. \tag{10}$$

5. Assign α to the F_4 root β_{j^*} with the largest $\cos\theta_{j^*}$.

Definition 3.3 (F4 quality threshold)**.** An E_8 root α passes the *F4 quality threshold* if $q(\alpha) \geq 0.7$, where $q(\alpha)$ is the cosine similarity between the (possibly fallback) projection and the assigned F_4 root.

Theorem 3.4 (Complete F4 coverage)**.** *All 240 E_8 roots satisfy $q(\alpha) = 1.0$ under the mapping of definition 3.2. Consequently, the F4 mapping fraction is exactly* 100%.

Proof. We analyze each root type:

Case 1: Type I roots $\pm e_i \pm e_j$ with $i < j \leq 4$. Both coordinates i, j lie in the first four, so $\pi(\alpha) = \pm e_i \pm e_j$, which is itself a long F_4 root. The cosine similarity with itself is 1.0.

Case 2: Type I roots $\pm e_i \pm e_j$ with $i \leq 4 < j$. Exactly one coordinate is in the first four: $\pi(\alpha) = \pm e_i$, which is a short F_4 root (type A). Similarity: 1.0.

Case 3: Type I roots $\pm e_i \pm e_j$ with $4 < i < j$. Both coordinates are in the last four: $\|\pi(\alpha)\| = 0$, so the fallback is used. The fallback projection $\pi(\alpha) = (r_5, r_6, r_7, r_8) = \pm e_{i-4} \pm e_{j-4}$ maps to coordinates 1–4, which is a long F_4 root. Similarity: 1.0.

Case 4: Type II roots $(\pm\frac{1}{2})^8$ (even negatives). The first four coordinates form a vector $\frac{1}{2}(\pm1, \pm1, \pm1, \pm1)$. This is an F_4 short root (type B if even negatives among the first four, type C if odd negatives among the first four—both cases match an F_4 root exactly). Similarity: 1.0.

In every case, the projection (or its fallback) is *exactly* an F_4 root, so $q(\alpha) = 1.0 \geq 0.7$. □

Remark 3.5. This contrasts with a naïve expectation that E_8 would "lose" roots when projected to F_4. The completeness follows from the fact that both the Type I and Type II E_8 roots, when projected to their first (or last) four coordinates, always land exactly on an F_4 root vector. This is a consequence of the branching rule $E_8 \supset F_4 \times G_2$ and the choice of coordinate-aligned projection.

3.3 The Albert Algebra and Jordan Trace

The exceptional Jordan algebra $J_3(\mathbb{O})$ consists of 3×3 Hermitian matrices over the octonions $\mathbb{O}$ [9, 12, 2]:

$$X = \begin{pmatrix} \xi_1 & x_3 & \bar{x}_2 \\ \bar{x}_3 & \xi_2 & x_1 \\ x_2 & \bar{x}_1 & \xi_3 \end{pmatrix}, \quad \xi_i \in \mathbb{R},\ x_i \in \mathbb{O}. \tag{11}$$

The automorphism group is $\mathrm{Aut}(J_3(\mathbb{O})) = F_4$. The *trace* is $\mathrm{tr}(X) = \xi_1 + \xi_2 + \xi_3$.

Definition 3.6 (Jordan trace of an F4 root)**.** For an F_4 root $\beta = (b_1, b_2, b_3, b_4) \in \mathbb{R}^4$, define the *projection matrix*:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \tag{12}$$

which maps β to the Albert algebra diagonal $(\xi_1, \xi_2, \xi_3) = P\beta = (b_1,\ b_2,\ b_3 + b_4)$. The *Jordan trace* is:

$$J(\beta) = \mathrm{tr}(P\beta) = \xi_1 + \xi_2 + \xi_3 = b_1 + b_2 + b_3 + b_4. \tag{13}$$

Remark 3.7. The projection matrix P embeds the F_4 Cartan subalgebra into the diagonal of $J_3(\mathbb{O})$. The combining of b_3 and b_4 into $\xi_3 = b_3 + b_4$ corresponds to the identification of the F_4 Cartan subalgebra with the traceless diagonal of $J_3(\mathbb{O})$ modulo the center. While $J(\beta)$ simplifies to $\sum_i b_i$, the 3-component decomposition $(b_1, b_2, b_3 + b_4)$ retains the Albert algebra structure and is used for idempotent classification.

Proposition 3.8 (Jordan trace values)**.** *The Jordan traces for the 48 F_4 roots and their multiplicities are:*

$J(\beta)$	-2	-1	0	$+1$	$+2$
Long roots	*4*	*0*	*8*	*0*	*4*
Short type A	*0*	*4*	*0*	*4*	*0*
Short type B	*1*	*0*	*6*	*0*	*1*
Short type C	*0*	*4*	*0*	*4*	*0*
Total	***5***	***8***	***14***	***8***	***5***

The idempotent-type roots ($|J| = 1$) number 16, and the nilpotent-type roots ($J = 0$) number 14.

Proof. Direct enumeration. For long roots $\pm e_i \pm e_j$: $J = (\pm1) + (\pm1) + 0 + 0$ when $i, j \le 2$, etc. For type B short roots $\frac{1}{2}(\pm1)^4$ with even negatives: $J = \frac{1}{2}(4), \frac{1}{2}(0), \frac{1}{2}(-4)$ for 0, 2, 4 negatives respectively, giving $J \in \{+2, 0, -2\}$. The remaining cases follow similarly. □

3.4 Idempotent Classification

The Jordan trace classifies F_4 roots by their algebraic type in $J_3(\mathbb{O})$:

Definition 3.9 (Root classification)**.** An F_4 root β is classified as:

- **Nilpotent**: $|J(\beta)| < 0.01$,
- **Primitive**: $0.01 \le |J(\beta)| < 0.5$,
- **Idempotent**: $||J(\beta)| - 1| < 0.1$,
- **Regular**: otherwise.

3.5 F4 Character Weights

Definition 3.10 (F4 character). For an F_4 root β_j with Weyl height $h(\beta_j) = \sum_{i=1}^{4} |(\beta_j)_i|$, define:

$$\chi(\beta_j) = \begin{cases} 2(1 + 0.1 \cdot h(\beta_j)) & \text{if } \beta_j \text{ is long,} \\ 1 + 0.1 \cdot h(\beta_j) & \text{if } \beta_j \text{ is short.} \end{cases} \tag{14}$$

Remark 3.11. The factor of 2 for long roots reflects their larger contribution to the adjoint representation of F_4 (dim = 52). The Weyl height modulation $0.1 \cdot h$ provides a mild position-dependent weighting within the Weyl chamber. For long roots with $h = 2$ (e.g., $e_1 + e_2$), the character is $2(1 + 0.2) = 2.4$. For short type A roots with $h = 1$ (e.g., e_1), it is 1.1.

3.6 Plasma Colormap for F4 Grid

The F4 crystalline grid visualization maps the Jordan trace $J \in [-2, +2]$ to an RGB color via the `plasma` colormap [10]:

$$\text{color}(J) = \text{plasma}\left(\frac{\text{clamp}(J, -2, 2) + 2}{4}\right), \tag{15}$$

where plasma(t) for $t \in [0, 1]$ interpolates from deep indigo ($t = 0$, $J = -2$) through magenta to bright yellow ($t = 1$, $J = +2$). We use a hardcoded 256-entry RGB lookup table generated from `matplotlib.cm.plasma`.

4 The F4 Exceptional Fourier Transform

4.1 Definition

The F4-EFT decomposes the prime gap signal into 48 spectral components indexed by F_4 roots.

Definition 4.1 (F4-EFT). Given N consecutive normalized gaps $\tilde{g}_0, \ldots, \tilde{g}_{N-1}$ with E_8 root assignments $\iota_0, \ldots, \iota_{N-1}$, the *F4 Exceptional Fourier Transform* is the 48-component complex vector:

$$\hat{E}_{F_4}(j) = \sum_{\substack{n=0 \\ f(\iota_n)=j}}^{N-1} (\tilde{g}_n - 1) \cdot \chi(\beta_j) \cdot \exp\left(i \cdot \frac{2\pi \|\beta_j\|}{\sqrt{2}} \cdot \frac{n}{N}\right), \quad j = 0, \ldots, 47, \tag{16}$$

where $f(\iota_n)$ is the F_4 index of E_8 root ι_n (under the mapping of definition 3.2), and χ is the F_4 character (14).

Remark 4.2. Key features of the F4-EFT:

- The factor $(\tilde{g}_n - 1)$ subtracts the expected mean gap under the PNT, so the EFT measures deviations from uniformity.
- The phase $\theta_j(n) = 2\pi\|\beta_j\|/\sqrt{2} \cdot n/N$ encodes a time-dependent sweep modulated by the F_4 root norm. For long roots ($\|\beta_j\| = \sqrt{2}$), the phase completes one full cycle over N gaps. For short roots ($\|\beta_j\| = 1$), the phase completes $1/\sqrt{2} \approx 0.707$ cycles.
- Each gap contributes only to the bin of its specific F_4 root $f(\iota_n)$, not to all 48 components.

4.2 Power Spectrum and Jordan Decomposition

The *F4 power spectrum* is:

$$\mathcal{P}(j) = |\hat{E}_{F_4}(j)|^2, \quad j = 0, \ldots, 47. \tag{17}$$

The *Jordan decomposition* groups the spectrum into 12 bins by Jordan trace value:

$$\hat{D}_k = \sum_{j:J(\beta_j)\in B_k} \hat{E}_{F_4}(j), \quad k = 0, \ldots, 11, \tag{18}$$

where $B_0 = [-3, -2.5), B_1 = [-2.5, -2), \ldots, B_{11} = [2.5, 3)$.

4.3 Phase-Lock Analysis

The F4-EFT phase structure is characterized by four metrics:

Definition 4.3 (Phase-lock diagnostics)**.**

$$\text{Phase coherence:} \quad C = |\frac{1}{48}\sum_{j=0}^{47} e^{i \arg \hat{E}_{F_4}(j)}|, \tag{19}$$

$$\text{Power entropy:} \quad H = -\sum_{j=0}^{47} p_j \ln p_j \Big/ \ln 48, \quad p_j = \mathcal{P}(j)/\textstyle\sum_k \mathcal{P}(k), \tag{20}$$

$$\text{Long/short ratio:} \quad R = \frac{\sum_{j\in\mathcal{L}} \mathcal{P}(j)}{\sum_{j\in\mathcal{S}} \mathcal{P}(j)}, \tag{21}$$

$$\text{Jordan peak trace:} \quad J^* = \arg\max_k |\hat{D}_k|^2. \tag{22}$$

The system is *phase-locked* if $C > 0.3$ and has *Jordan structure* if $\max_k |\hat{D}_k|^2 > 2 \cdot \text{mean}_k |\hat{D}_k|^2$.

4.4 The Salem–Jordan Kernel

Definition 4.4 (Salem–Jordan kernel)**.** For temperature parameter $\tau = 0.5$, the kernel applied to spectral magnitude $m \geq 0$ is:

$$K_J(m, \tau) = \frac{52 - 4m^2}{52} \cdot \frac{1}{e^{m/\tau} + 1}. \tag{23}$$

The Fermi–Dirac factor provides exponential suppression for large magnitudes. The character factor $(52 - 4m^2)/52$ approximates the F_4 adjoint trace near the identity: $\chi_{F_4}(e^{x/\tau}) \approx 52 - 4|x|^2$.

4.5 Crystalline Vertex Extraction

Definition 4.5 (Crystalline vertex score)**.** For gap index n mapping to F_4 root $j = f(\iota_n)$:

$$\sigma(n) = \begin{cases} 2 \cdot \mathcal{P}(j) & \text{if } ||J(\beta_j)| - 1| < 0.2 \quad \text{(idempotent boost)}, \\ \mathcal{P}(j) & \text{otherwise.} \end{cases} \tag{24}$$

The top V gap indices by $\sigma(n)$ are the *crystalline vertices*.

Remark 4.6. The idempotent boost doubles the score for roots with $|J| \approx 1$ (i.e., idempotent elements of $J_3(\mathbb{O})$). By proposition 3.8, 16 of the 48 F_4 roots qualify. Selection uses a min-heap of size V for $O(N \log V)$ extraction.

5 E8 Lattice Decoding

5.1 Gap Embedding into $\mathbb{R}^8$

Given 8 consecutive primes $p_n, p_{n+1}, \ldots, p_{n+7}$ and the preceding prime p_{n-1}, we compute 8 gaps $g_i = p_{n+i} - p_{n+i-1}$ for $i = 0, \ldots, 7$. The embedding is:

$$v = \frac{\sqrt{2}}{\sqrt{8} \cdot \sigma_g} \cdot (g - \bar{g}), \tag{25}$$

where $\bar{g} = \frac{1}{8}\sum_i g_i$ is the mean and $\sigma_g = \sqrt{\frac{1}{8}\sum_i (g_i - \bar{g})^2}$ is the standard deviation. This normalizes v to have typical norm $\approx \sqrt{2}$, the E_8 root length.

5.2 Closest Vector Decoding

The E_8 lattice is $\Lambda_{E_8} = D_8 \cup (D_8 + c)$, where:

- $D_8 = \{x \in \mathbb{Z}^8 : \sum x_i \equiv 0 \pmod 2\}$ (checkerboard lattice).
- $c = (\frac{1}{2}, \ldots, \frac{1}{2})$.

Algorithm 1 E8 Closest Vector Decoding

```
Require: Input vector $v \in \mathbb{R}^8$
Ensure: Nearest $E_8$ lattice point $\lambda^*$ and distance $d^*$

 1: Integer sublattice: $z \leftarrow \mathrm{round}(v)$
 2: if $\sum z_i$ is odd then
 3:     $k \leftarrow \arg\max_i |v_i - z_i|$                ▷ Largest rounding error
 4:     $z_k \leftarrow z_k + \mathrm{sgn}(v_k - z_k)$        ▷ Flip parity
 5: end if
 6: $d_1 \leftarrow \|v - z\|$

 7: Half-integer sublattice: $w \leftarrow \mathrm{round}(v - c)$
 8: if $\sum w_i$ is odd then
 9:     $k \leftarrow \arg\max_i |(v_i - \frac{1}{2}) - w_i|$
10:     $w_k \leftarrow w_k + \mathrm{sgn}((v_k - \frac{1}{2}) - w_k)$
11: end if
12: $y \leftarrow w + c$; $d_2 \leftarrow \|v - y\|$

13: return $(z, d_1)$ if $d_1 \le d_2$, else $(y, d_2)$
```

The packing radius of E_8 is $R_{\text{pack}} = 1/\sqrt{2} \approx 0.7071$. Blocks with decoding error $d^* < R_{\text{pack}}$ are *correctable*.

5.3 Bit Extraction via $E_8/2E_8$

The quotient $E_8/2E_8 \cong \mathbb{F}_2^4$ is isomorphic to the extended Hamming code $\mathcal{H}_8$ [3, 5]. From the lattice point $\lambda^* = (y_1, \ldots, y_8)$ (translated to integers if half-integer), we extract 4 bits:

$$m_1 = (y_1 + y_2 + y_3 + y_4) \bmod 2, \tag{26}$$
$$m_2 = (y_1 + y_2 + y_5 + y_6) \bmod 2, \tag{27}$$
$$m_3 = (y_1 + y_3 + y_5 + y_7) \bmod 2, \tag{28}$$
$$m_4 = y_1 \bmod 2. \tag{29}$$

These correspond to the generator matrix of the $[8, 4, 4]$ extended Hamming code. Each block of 8 primes yields 4 bits. Pairs of consecutive nibbles $(m_1 m_2 m_3 m_4)$ can be assembled into bytes in either high–low or low–high order.

6 Multi-Method Decoder

We implement twelve independent methods for extracting structured information from the prime gap sequence, organized in five groups. Each method produces a byte stream that is evaluated for non-random structure.

6.1 Group 1: E8 Lattice Decoding

M1 Hamming 4-bit (high–low pairing): The standard $E_8/2E_8$ extraction of section 5.3, with consecutive nibbles assembled as $(n_{2k} \ll 4) \mid n_{2k+1}$. Rate: $\frac{1}{2}$ byte per block of 8 primes.

M2 Hamming 4-bit (low–high pairing): Same extraction, opposite nibble ordering: $(n_{2k+1} \ll 4) \mid n_{2k}$.

M3 Hamming 4-bit (low-error only): Restricts to correctable blocks ($d^* < 1/\sqrt{2}$).

M4 Hamming raw bits: The 4-bit stream packed into bytes (MSB first), without nibble pairing.

M5 E8 sign bits: For each block, record $\text{bit}_k = [y_k \geq 0]$ for $k = 1, \ldots, 8$. Rate: 1 byte per block.

M6 Sublattice flag: 1 bit per block indicating integer (0) or half-integer (1) sublattice. Rate: $\frac{1}{8}$ byte per block.

M7 E8 parity bits: $\text{bit}_k = y_k \bmod 2$ for each coordinate. Rate: 1 byte per block.

6.2 Group 2: E8 Root Assignment

M8 E8 root index mod 256: For each gap n, output $\iota(\tilde{g}_n) \bmod 256$ as a byte. Rate: 1 byte per gap.

M9 E8 projection slope (quantized): Map the projection slope $s \in [-10, 10]$ to $[0, 255]$ via $\lfloor 255 \cdot (s + 10)/20 \rfloor$.

6.3 Group 3: Raw Gap Properties

M10 Gap value mod 256: Direct gap size g_n mod 256.

M11 Gap/2 mod 256: $(g_n/2)$ mod 256 (gaps are even for $p > 2$).

M12 Normalized gap (quantized): Map $\tilde{g}_n \in [0, 4]$ to $[0, 255]$.

6.4 Group 4: F4 Sub-harmonic

M13 F4 root index: For F4-mapped gaps, the F4 root index (0–47) as a byte.

M14 F4 root index mod 26: Map to letters A–Z via $65 + (j \bmod 26)$.

M15 Jordan trace classification bits: 2 bits per F4 gap: bit $0 = [J > 0]$, bit $1 = [|J| > 1]$.

6.5 Group 5: Crystalline Vertices

M16 Vertex gap values: Gap sizes g_n at the V crystalline vertex positions, mod 256.

M17 Vertex gaps mod 26: Same, mapped to A–Z.

M18 Vertex spacing: Differences between consecutive sorted vertex positions, mod 256.

M19 Vertex prime values mod 256: The prime p_{n+1} at each vertex position, mod 256.

6.6 Statistical Evaluation

Each method's output byte stream is evaluated by:

Definition 6.1 (Decoder metrics)**.**

- **Shannon entropy**: $H = -\sum_{b=0}^{255} p_b \log_2 p_b$ bits per byte (max 8.0 for uniform).
- **Chi-square p-value**: Test byte frequencies against uniform $U(0, 255)$ with 255 degrees of freedom, using normal approximation $z = (\chi^2 - 255)/\sqrt{510}$.
- **Longest printable run**: Maximum contiguous substring of bytes in the printable ASCII range $[32, 126]$.

A stream showing non-random structure would exhibit: low entropy ($H \ll 8$), small p-value ($p \ll 0.05$), and/or long printable runs ($\gg 10$ characters forming readable text).

7 Computational Implementation

7.1 Prime Data

Primes are loaded from external files `primes1.txt` through `primes50.txt` (generated by `primesieve`[1]). Each file contains whitespace-separated integers with a header line. The loader skips the header (detected by the presence of alphabetic characters), parses all integers > 1, and stops at the requested maximum.

[1]`https://primesieve.org/`

7.2 Self-Contained C Implementation

The F4 crystalline grid is implemented as a single C source file (`f4_crystalline_grid.c`, ∼600 lines) that produces a PPM image with no external dependencies beyond `libm` and OpenMP. The full pipeline:

Algorithm 2 F4 Crystalline Grid (C/OpenMP)

```
Require: -max-primes $N$, -n-vertices $V$, -size $S$
Ensure: PPM image file
 1: Generate 240 $E_8$ roots, 48 $F_4$ roots                         ▷ $O(1)$
 2: Build $E_8 \to F_4$ mapping (cosine similarity)                 ▷ $O(240 \times 48)$
 3: Load primes from files                                          ▷ $O(N)$
 4: for each gap $n = 0, \ldots, N-2$ (OpenMP parallel) do
 5:     $g_n \leftarrow p_{n+1} - p_n$; $\tilde{g}_n \leftarrow g_n / \max(\log p_n, 1)$
 6:     $\iota_n \leftarrow \lfloor 240 \cdot (\sqrt{\max(\tilde{g}_n, 0.01)}/\sqrt{2} \bmod 1) \rfloor$
 7:     $j_n \leftarrow f(\iota_n)$ (F4 index); $J_n \leftarrow J(\beta_{j_n})$ (Jordan trace)
 8: end for
 9: Compute F4-EFT spectrum (48 complex components)                 ▷ Serial, $O(N)$
10: Compute vertex scores $\sigma(n)$ with idempotent boost          ▷ OpenMP
11: Select top $V$ by min-heap                                      ▷ $O(N \log V)$
12: Compute Ulam coordinates $u(p_n)$ for all $n$                    ▷ OpenMP
13: Allocate $S \times S \times 3$ pixel buffer (black)
14: Plot F4 primes: map $J_n \in [-2, 2]$ to plasma LUT              ▷ OpenMP
15: Draw white circles with yellow edges at vertex positions
16: Write P6 PPM
```

Remark 7.1 (Memory estimate)**.** For $N = 2 \times 10^6$ primes: primes (16 MB), gaps + normalized gaps (32 MB), E8/F4 assignments (8 MB), scores (16 MB), Ulam coordinates (16 MB), pixel buffer at $6000^2 \times 3$ (108 MB). Total: ≈ 200 MB. For $N = 5 \times 10^7$: ≈ 2 GB. For $N = 10^8$: ≈ 4 GB.

7.3 Python F4 Analysis Pipeline

The Python implementation (`e8_f4_prime_analysis.py`) provides the reference pipeline using NumPy, with supporting modules:

- `f4_lattice.py`: F_4 root generation, $E_8 \to F_4$ mapping, Cartan matrix, character precomputation.
- `jordan_algebra.py`: Octonion multiplication table, Albert algebra $J_3(\mathbb{O})$, Jordan trace, Cayley plane projection.
- `f4_eft.py`: F4-EFT computation, phase-lock analysis, crystalline vertex extraction.
- `salem_jordan.py`: Salem–Jordan kernel, Fermi–Dirac filter, adaptive τ selection, spectral filtering.

7.4 Multi-Method Decoder Implementation

The decoder (`e8_multi_decoder.py`) imports from both the F4 analysis pipeline and the E8 lattice decoder (`e8_prime_decoder.py`). It runs all 19 base methods plus first-N filtered variants (first 100 and first 1000 bytes of each), totaling 53+ method variants evaluated in a single pass.

7.5 Build System

Listing 1: Makefile targets

```
CC = gcc
CFLAGS = -O3 -march=native -Wall -fopenmp

f4_crystalline_grid: f4_crystalline_grid.c
        $(CC) $(CFLAGS) -o $@ $< -lm

run-crystal:
        ./f4_crystalline_grid --max-primes 2000000 --n-vertices 38

run-decoder:
        python3 e8_multi_decoder.py --max-primes 2000000
```

The PPM output is converted to PNG via ImageMagick: `convert grid.ppm grid.png`.

8 Results

8.1 F4 Crystalline Grid ($N = 2{,}000{,}000$ primes)

Running `f4_crystalline_grid -max-primes 2000000 -n-vertices 38` produces the following:

Table 1: F4 Crystalline Grid: summary statistics ($N = 2 \times 10^6$)

Quantity	Value
Primes loaded	2,000,000
Prime range	2 to 32,452,843
Total gaps	1,999,999
E_8 roots	240
F_4 roots	48
E_8 roots passing F_4 threshold	240 (100%)
Gaps mapping to F_4	1,999,999 (100.0%)
Crystalline vertices selected	38
Ulam coordinate range	$x \in [-2848, 2848]$, $y \in [-2848, 2848]$
Image size	6000 × 6000 pixels
PPM file size	103 MB
PNG file size (after conversion)	2.4 MB

The 100% F4 mapping fraction confirms theorem 3.4: every E_8 root projects to an F_4 root with cosine similarity 1.0.

8.2 Multi-Method Decoder Results ($N = 100{,}000$ primes)

The decoder evaluated 53 method variants. table 2 shows the methods ranked by longest printable ASCII run, and table 3 ranks by lowest entropy.

Table 2: Top methods by longest printable ASCII run

#	Method	Bytes	MaxRun	Entropy
14	F4 root index mod 26 (letter)	85,502	85,502	3.541
13	F4 root index (F4 gaps only)	85,502	46	3.693
12	Normalized gap (quantized)	99,999	20	6.701
9	E8 projection slope (quantized)	99,999	11	2.131
8	E8 root index mod 256	99,999	11	7.266
3	Hamming 4-bit (low-error)	1,335	9	7.845
2	Hamming 4-bit (low–high)	6,249	9	7.950
1	Hamming 4-bit (high–low)	6,249	8	7.950
15	Jordan trace class. bits	21,375	8	3.911

Table 3: Top methods by lowest entropy (most structure)

#	Method	Bytes	Entropy	χ^2 p
17	Vertex gaps mod 26 (letters)	500	0.118	0.000
16	Vertex gap values mod 256	500	0.128	0.000
9	E8 projection slope	99,999	2.131	0.000
7	E8 parity bits	12,499	3.196	0.000
14	F4 root index mod 26	85,502	3.541	0.000
13	F4 root index (F4 only)	85,502	3.693	0.000
15	Jordan trace class. bits	21,375	3.911	0.000
10	Gap value mod 256	99,999	3.943	0.000

8.3 Analysis of Decoder Results

Hamming methods (M1–M4). Entropy ≈ 7.95 bits/byte, indistinguishable from random. The chi-square p-value is < 0.01, indicating a slight deviation from uniformity in the nibble distribution, but the entropy is so close to the maximum of 8.0 that no meaningful structure is present. The longest printable runs (8–9 characters) are consistent with random expectation for streams of $\sim$6,000 bytes.

Projection slope (M9). Entropy 2.13 bits/byte reflects the fact that the slope takes only a small number of distinct values ($\{-10, -2, -1, 0, 1, 2, 10\}$ for the 240 roots), so the quantized byte values cluster around a few values (notably `0x72`, `0x7F`, `0x8C`, `0xFF`, `0x00`). This is a consequence of the E8 root geometry, not of prime structure.

F4 root index (M13–M14). Entropy 3.5–3.7 bits/byte. Since there are only 48 F_4 root indices, the maximum possible entropy is $\log_2 48 \approx 5.6$ bits; the observed value indicates non-uniform distribution across the 48 roots. When mapped to letters A–Z (mod 26), the entire 85,502-byte stream is “printable” by construction, but the resulting text is not readable English.

Vertex gap values (M16–M17). Entropy 0.12–0.13 bits/byte—the lowest of all methods. This is because crystalline vertices are selected for maximal F4 power spectrum, and the corresponding gaps cluster around a few values. With only 500 bytes, the low entropy reflects the vertex selection criterion rather than an encoded message.

Parity bits (M7). Entropy 3.20 bits/byte. The low entropy arises because most E_8 lattice point coordinates are small integers (0 or ±1), so parity is highly non-uniform.

8.4 Performance

Table 4: Runtime for $N = 2 \times 10^6$ primes on Intel Core Ultra 9 275HX

Program	**Time**	**Output**
`f4_crystalline_grid` (C/OpenMP)	< 5 s	103 MB PPM
ImageMagick `convert` PPM $\to$ PNG	~ 2 s	2.4 MB PNG
`e8_multi_decoder.py` (10^5 primes)	3.6 s	53 methods

9 Octonion Arithmetic

The Jordan algebra module implements full octonion arithmetic as the algebraic foundation for the F_4-Albert algebra connection.

9.1 Multiplication Table

The octonions $\mathbb{O}$ are an 8-dimensional non-associative algebra with basis $\{1, e_1, \ldots, e_7\}$ and multiplication: $e_i \cdot e_j = f_{ijk}\, e_k$, where f_{ijk} are the structure constants given by the Cayley–Dickson construction. The multiplication table is:

$$\mathrm{MULT\_TABLE}[i][j] = k, \qquad \mathrm{MULT\_SIGNS}[i][j] = \pm 1, \tag{30}$$

such that $e_i \cdot e_j = \mathrm{MULT\_SIGNS}[i][j] \cdot e_{\mathrm{MULT\_TABLE}[i][j]}$.

The full 8×8 sign matrix is:

$$\mathrm{SIGNS} = \begin{pmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & - & + \\ + & - & - & + & + & + & - & - \\ + & + & - & - & + & - & + & - \\ + & - & - & - & - & + & + & + \\ + & + & - & + & - & - & + & - \\ + & + & + & - & - & - & - & + \\ + & - & + & + & - & + & - & - \end{pmatrix} \tag{31}$$

Operations implemented:

- Addition, subtraction (componentwise).
- Multiplication: $\mathbb{O} \times \mathbb{O} \to \mathbb{O}$ via $(a \cdot b)_k = \sum_{i,j} f_{ijk}\, a_i\, b_j$ ($O(64)$ multiplications).
- Conjugation: $\bar{a} = a_0 - \sum_{k=1}^{7} a_k e_k$.
- Norm: $\|a\| = \sqrt{a_0^2 + \cdots + a_7^2}$.

9.2 Albert Algebra Product

The Albert algebra $J_3(\mathbb{O})$ supports the Jordan product $X \circ Y = (XY + YX)/2$, computed as:

$$(X \circ Y)_{ii} = X_{ii}Y_{ii} + \mathrm{Re}(X_{jk}\overline{Y_{jk}}) + \mathrm{Re}(X_{ki}\overline{Y_{ki}}), \tag{32}$$

where (i, j, k) is a cyclic permutation of $(1, 2, 3)$.

9.3 Cayley Plane Projection

The Cayley plane $\mathbb{O}P^2$ is a 16-dimensional manifold on which F_4 acts. For visualization, we project F_4 roots to polar coordinates (r, θ) via:

$$z = b_1 + i\, b_2, \quad w = b_3 + i\, b_4, \tag{33}$$

$$r = \sqrt{|z|^2 + |w|^2}, \quad \theta = \arctan\left(\frac{\mathrm{Im}(z) + \mathrm{Im}(w)}{\mathrm{Re}(z) + \mathrm{Re}(w)}\right). \tag{34}$$

10 Discussion

10.1 What the Patterns Are

The ring structure in the E8 slope visualization arises from the interplay of two effects:

1. **Gap periodicity in root space**: The map $\tilde{g} \mapsto \varphi(\tilde{g})$ wraps the normalized gap around the circle $[0, 1)$ via a square-root scaling. As one moves outward in the Ulam spiral, the typical gap size grows (logarithmically), causing the phase to advance and the assigned slope to cycle.
2. **Ulam spiral geometry**: Primes at similar distances from the center tend to have similar magnitudes and hence similar normalized gaps, producing coherent coloring in annular regions.

The 100% F4 mapping (theorem 3.4) means that the F4 crystalline grid visualizes *all* prime gaps, not a filtered subset. The crystalline vertices select gaps where the F4-EFT power spectrum is concentrated, boosted by the idempotent condition on the Jordan trace.

10.2 What the Patterns Are Not

- The E8 root assignment is a *deterministic function of the gap size alone.* It encodes no deeper number-theoretic structure beyond the normalized gap distribution.
- The ring structure would appear for *any* sequence of positive reals with similar statistical properties to prime gaps (e.g., a Poisson process with rate $1/\log n$).
- The crystalline vertices are not "special primes" in any number-theoretic sense.
- The multi-method decoder (section 6) finds no method producing readable text. All methods either show near-maximum entropy (random) or low entropy explained by the algebraic structure of the mapping (small number of root types), not by an encoded message.

10.3 The Complete F4 Coverage Phenomenon

The result of theorem 3.4—that all 240 E_8 roots project to F_4 with similarity 1.0—deserves comment. This is *not* a generic feature of projections from E_8 to rank-4 sublattices. It holds specifically because:

1. The coordinate-aligned projection $(r_1, \ldots, r_8) \mapsto (r_1, \ldots, r_4)$ sends each E_8 root type to an exact F_4 root type (long to long, short to short).
2. The fallback to coordinates $(r_5, \ldots, r_8)$ catches the roots concentrated entirely in the fiber, which again project exactly to F_4 roots.

A random 4-dimensional projection of E_8 would typically yield far fewer exact F_4 matches.

10.4 Reproducibility

All source code is available at:

`https://github.com/johnjanik/HodgedeRham`

The repository contains:

- `f4_crystalline_grid.c`: Self-contained C/OpenMP renderer (PPM output, ~600 lines, no dependencies beyond `libm`).
- `e8_multi_decoder.py`: Multi-method decoder (12 base methods $\times$ filtered variants $= 53$ total).
- `e8_f4_prime_analysis.py`: Python F4 analysis pipeline.
- `f4_lattice.py`, `jordan_algebra.py`, `f4_eft.py`, `salem_jordan.py`: Supporting modules.
- `e8_prime_decoder.py`: E8 lattice decoder with statistical tests.
- `Makefile`: Build and run targets.

Build with `gcc -O3 -march=native -fopenmp -lm`.

11 Conclusion

We have presented a complete, reproducible pipeline for analyzing prime gap structure through the lens of exceptional Lie lattices. The E8 root assignment map transforms the scalar sequence of normalized prime gaps into a structured coloring of the Ulam spiral, and the F4 restriction, Jordan trace, and Salem–Jordan filter extract a discrete "crystalline grid" from the continuous ring pattern.

The key structural result (theorem 3.4) is that all 240 E_8 roots project exactly to F_4 roots under the natural coordinate-aligned embedding, yielding 100% F4 coverage of all prime gaps.

The multi-method decoder finds no evidence of an encoded message in the prime gap data: all 53 extraction methods either produce near-random output (entropy ≈ 8 bits/byte) or low-entropy output fully explained by the finite cardinality of root types.

The mathematical objects involved—E_8, F_4, the Albert algebra $J_3(\mathbb{O})$, the Salem kernel—are among the most beautiful structures in mathematics. Whether the observed patterns reflect deep connections to prime distribution or are primarily a consequence of projecting gap statistics onto a rich algebraic framework remains an open question.

A Complete E8 Root Enumeration

For reproducibility, we specify the exact enumeration order of the 240 E_8 roots used in all computations.

Type I (roots 0–111): Iterate $i = 0, \ldots, 7$ and $j = i + 1, \ldots, 7$. For each pair (i, j), enumerate the four sign patterns $(s_1, s_2) \in \{(-1, -1), (-1, +1), (+1, -1), (+1, +1)\}$. The root is $s_1 e_i + s_2 e_j$.

Type II (roots 112–239): Iterate mask $= 0, \ldots, 255$. For each mask, let $r_k = \frac{1}{2}$ if bit k is set, $-\frac{1}{2}$ otherwise ($k = 0, \ldots, 7$). Keep only masks where the number of unset bits (negative signs) is even. This yields 128 roots in mask order.

B Complete F4 Root Enumeration

Long roots (indices 0–23): Iterate $i = 0, \ldots, 3$ and $j = i + 1, \ldots, 3$. For each pair, enumerate four sign patterns.

Short type A (indices 24–31): Iterate $i = 0, \ldots, 3$, signs $s \in \{-1, +1\}$.

Short type B (indices 32–39): Iterate mask $= 0, \ldots, 15$, keep even negative count.

Short type C (indices 40–47): Iterate mask $= 0, \ldots, 15$, keep odd negative count.

C Plasma Colormap Table

The 256-entry plasma LUT is generated from `matplotlib 3.8`:

```python
import matplotlib.cm as cm
for i in range(256):
    r, g, b, _ = cm.plasma(i / 255.0)
    print(f"␣␣{{{int(r*255)},␣{int(g*255)},␣{int(b*255)}}},")
```

The full table is embedded in `f4_crystalline_grid.c` and maps $t \in [0,1]$ (derived from $J \in [-2,2]$ via $t = (J+2)/4$) to an RGB triple. The endpoints: $t = 0$ maps to deep indigo (12, 7, 134); $t = 1$ maps to bright yellow (239, 248, 33).

D Decoder Method Cross-Reference

M#	Method	Source function	Bits/gap
1	Hamming high–low	`extract_bits` + pair	0.5
2	Hamming low–high	`extract_bits` + pair	0.5
3	Hamming low-error	`extract_bits` + filter	0.5
4	Hamming raw bits	`extract_bits`	0.5
5	Sign bits	`e8_decode` signs	1.0
6	Sublattice flag	`e8_decode` type	0.125
7	Parity bits	`e8_decode` parities	1.0
8	Root index mod 256	`assign_root`	1.0
9	Projection slope	`projected_slopes`	1.0
10	Gap mod 256	raw gap	1.0
11	Gap/2 mod 256	raw gap /2	1.0
12	Normalized gap	$\tilde{g}$ quantized	1.0
13	F4 root index	`project_e8_to_f4`	1.0
14	F4 mod 26 (letter)	F4 index mod26	1.0
15	Jordan trace bits	$J > 0$, $\|J\| > 1$	0.25
16	Vertex gaps	crystalline selection	var.
17	Vertex mod 26	crystalline + mod 26	var.
18	Vertex spacing	Δ(vertex positions)	var.
19	Vertex primes	p at vertices	var.

References

[1] J. F. Adams. *Lectures on Exceptional Lie Groups.* University of Chicago Press, 1996.

[2] J. C. Baez. The octonions. *Bull. Amer. Math. Soc.* (N.S.), 39(2):145–205, 2002.

[3] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer, 3rd edition, 1999.

[4] H. Cramér. On the order of magnitude of the difference between consecutive prime numbers. *Acta Arith.*, 2(1):23–46, 1936.

[5] W. Ebeling. *Lattices and Codes.* Springer, 3rd edition, 2013.

[6] H. Freudenthal. Beziehungen der E_7 und E_8 zur Oktavenebene, I–XI. *Indag. Math.*, 16–25, 1954–1963.

[7] D. A. Goldston, J. Pintz, and C. Y. Yıldırım. Primes in tuples I. *Ann. Math.*, 170(2):819–862, 2009.

[8] J. E. Humphreys. *Introduction to Lie Algebras and Representation Theory.* Springer, 1972.

[9] P. Jordan, J. von Neumann, and E. Wigner. On an algebraic generalization of the quantum mechanical formalism. *Ann. Math.*, 35(1):29–64, 1934.

[10] J. D. Hunter. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.*, 9(3):90–95, 2007.

[11] J. Maynard. Small gaps between primes. *Ann. Math.*, 181(1):383–413, 2015.

[12] K. McCrimmon. *A Taste of Jordan Algebras.* Springer, 2004.

[13] OpenMP Architecture Review Board. *OpenMP Application Programming Interface, Version 5.0*, 2018.

[14] R. Salem. Power series with integral coefficients. *Duke Math. J.*, 12(2):153–172, 1945.

[15] T. A. Springer. Jordan algebras and algebraic groups. *Ergebnisse der Mathematik*, vol. 75. Springer, 1998.

[16] S. M. Ulam. A visual display of some properties of the distribution of primes. *Notices Amer. Math. Soc.*, 10:197, 1963.

[17] M. S. Viazovska. The sphere packing problem in dimension 8. *Ann. Math.*, 185(3):991–1015, 2017.

[18] Y. Zhang. Bounded gaps between primes. *Ann. Math.*, 179(3):1121–1174, 2014.