Visualizing Prime Numbers Through the E_8 Lattice

A Pedagogical Guide to the Mathematics Behind
E_8 Projection Slope Coloring of the Ulam Spiral

A Tutorial on Exceptional Geometry in Number Theory

February 1, 2026

Abstract

This tutorial provides a complete, self-contained guide to creating visualizations that reveal hidden structure in the distribution of prime numbers using the exceptional Lie algebra E_8. We develop each mathematical component from first principles: the E_8 root lattice, prime gap normalization, root assignment algorithms, two-dimensional projection, and the Ulam spiral coordinate system. The resulting visualization—primes colored by their E_8 projection slope—reveals striking concentric ring patterns that demonstrate primes are not randomly distributed in E_8 root space but follow coherent wave-like structures. We provide complete algorithms and code, enabling readers to reproduce and extend these results.

Contents

1 Introduction

1.1 The Mystery of Prime Distribution

Prime numbers—integers greater than 1 divisible only by 1 and themselves—have fascinated mathematicians for millennia. Despite their simple definition, their distribution among the integers exhibits both regularity and apparent randomness that has resisted complete understanding.

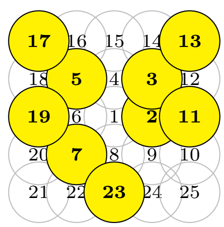
The **Prime Number Theorem** tells us that the number of primes up to x, denoted $\pi(x)$, satisfies:

$$\pi(x) \sim \frac{x}{\ln x} \quad \text{as } x \to \infty \tag{1}$$

This gives the "density" of primes but says nothing about their precise locations. The gaps between consecutive primes, $g_n = p_{n+1} - p_n$, appear erratic when examined individually.

1.2 The Ulam Spiral: A Visual Discovery

In 1963, mathematician Stanislaw Ulam, while doodling during a boring meeting, arranged the positive integers in a square spiral and marked the primes. To his surprise, the primes clustered along diagonal lines:



The diagonal clustering corresponds to prime-generating quadratic polynomials like Euler's famous $n^2 + n + 41$, which produces 40 consecutive primes for $n = 0, 1, \ldots, 39$.

1.3 Our Goal: Revealing Deeper Structure with E_8

This tutorial develops a visualization technique that goes beyond simply marking primes. We will:

1. Encode each prime's **gap** (distance to the next prime) as a position in the 8-dimensional E_8 root lattice

2. **Project** this 8D information down to a 2D "slope" value

3. **Color** each prime in the Ulam spiral according to this slope

The result reveals **concentric ring patterns** showing that the E_8 encoding of primes evolves coherently—not randomly—as we move outward through the spiral.

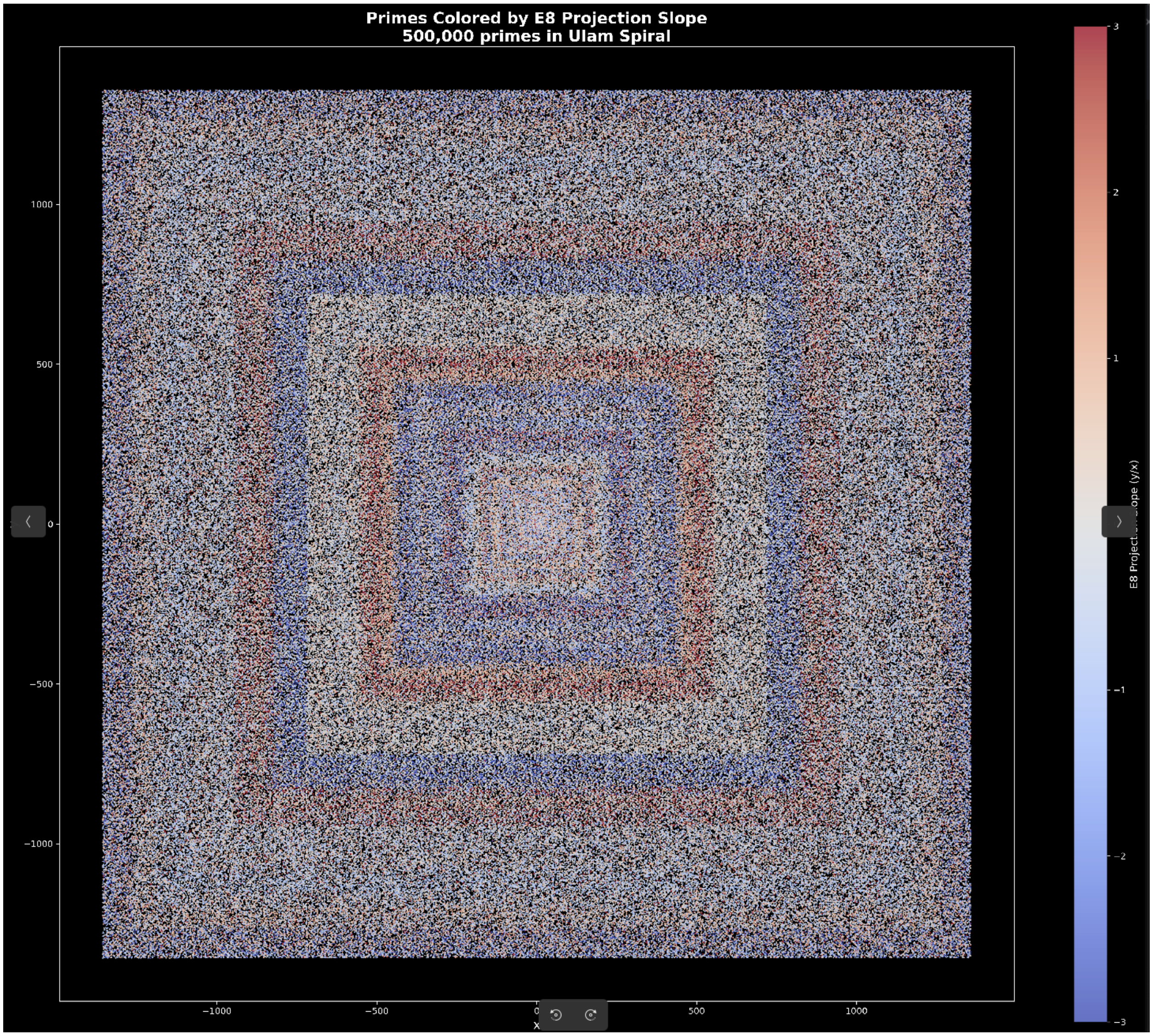


Figure 1: E_8 encoding of primes

1.4 Prerequisites

This tutorial assumes familiarity with:

- Basic linear algebra (vectors, matrices, norms)
- Elementary number theory (primes, divisibility)
- Python programming (NumPy, Matplotlib)

We will develop all E_8-specific mathematics from scratch.

2 The E_8 Root Lattice

2.1 What is E_8?

E_8 is the largest of the five **exceptional simple Lie algebras**. While this abstract algebraic definition requires graduate-level mathematics, we can work directly with its concrete realization as a lattice in $\mathbb{R}^8$.

Definition 2.1.1 (E_8 Lattice)**.** The E_8 lattice $\Lambda_{E_8} \subset \mathbb{R}^8$ consists of all points $(x_1, x_2, \ldots, x_8)$ satisfying:

1. All coordinates are integers, OR all coordinates are half-integers (i.e., of the form $n + \frac{1}{2}$ for integer n)
2. The sum of all coordinates is even: $\sum_{i=1}^{8} x_i \equiv 0 \pmod 2$

Example 2.1.2. The following are E_8 lattice points:

- $(1, 1, 0, 0, 0, 0, 0, 0)$ — integers summing to 2 (even) ✓
- $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ — half-integers summing to 4 (even) ✓
- $(1, 0, 0, 0, 0, 0, 0, 0)$ — integers summing to 1 (odd) ✗
- $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, 0)$ — mixed integers/half-integers ✗

2.2 The 240 Root Vectors

The **roots** of E_8 are the lattice points closest to the origin (excluding the origin itself). All roots have the same Euclidean norm.

Proposition 2.2.1. *The E_8 root system Φ_{E_8} contains exactly 240 vectors, all of norm $\sqrt{2}$.*

These 240 roots divide into two types:

Type I Roots (112 vectors)

These have two coordinates equal to ± 1 and six coordinates equal to 0:

$$\text{Type I}: \quad (\ldots, \pm 1, \ldots, \pm 1, \ldots) \quad \text{with 6 zeros} \tag{2}$$

Counting: Choose 2 positions from 8 for the non-zero entries ($\binom{8}{2} = 28$ ways), then choose signs ($2^2 = 4$ ways):

$$|\text{Type I}| = \binom{8}{2} \times 2^2 = 28 \times 4 = 112 \tag{3}$$

Example 2.2.2. Type I roots include:

$$\begin{gathered}(1, 1, 0, 0, 0, 0, 0, 0), \quad (1, -1, 0, 0, 0, 0, 0, 0) \\ (1, 0, 1, 0, 0, 0, 0, 0), \quad (0, 0, 0, 0, 0, 0, -1, -1)\end{gathered}$$

Type II Roots (128 vectors)

These have all coordinates equal to $\pm\frac{1}{2}$, with an **even number of minus signs**:

$$\text{Type II}: \quad \left(\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2},\pm\frac{1}{2}\right) \quad \text{even \# of } -\frac{1}{2}\text{'s} \tag{4}$$

Counting: Of the $2^8 = 256$ possible sign choices, exactly half have an even number of minus signs:

$$|\text{Type II}| = \frac{256}{2} = 128 \tag{5}$$

Example 2.2.3. Type II roots include:

$$\left(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) \quad \text{(0 minus signs)}$$
$$\left(-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) \quad \text{(2 minus signs)}$$
$$\left(-\frac{1}{2},-\frac{1}{2},-\frac{1}{2},-\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) \quad \text{(4 minus signs)}$$

Verification of norm: For Type II,

$$\|v\|^2 = 8 \times \left(\frac{1}{2}\right)^2 = 8 \times \frac{1}{4} = 2 \quad \Rightarrow \quad \|v\| = \sqrt{2} \tag{6}$$

2.3 Generating the Roots in Code

2.4 Why E_8?

The E_8 lattice has remarkable properties:

1. **Densest packing**: In 8 dimensions, E_8 achieves the densest possible sphere packing (proven by Viazovska, 2016).
2. **Self-dual**: $\Lambda_{E_8}^* = \Lambda_{E_8}$ (the dual lattice equals itself).
3. **Even**: All vectors have even squared norm ($\|v\|^2 \in 2\mathbb{Z}$).
4. **Kissing number 240**: Each sphere in the packing touches exactly 240 others.

The number 248 appears throughout: the Lie algebra $\mathfrak{e}_8$ has dimension 248, decomposing as $248 = 8 + 240$ (Cartan subalgebra plus root spaces).

For our purposes, E_8 provides a rich, rigid structure for encoding 1-dimensional information (prime gaps) in a way that preserves geometric relationships.

3 Prime Gaps and Normalization

3.1 Prime Gaps

Definition 3.1.1. The n-th **prime gap** is:

$$g_n = p_{n+1} - p_n \tag{7}$$

Algorithm 1 Generate all 240 E_8 root vectors

```
roots ← []                                          ▷ Type I: 112 roots
for i = 0 to 7 do
    for j = i + 1 to 7 do
        for s_1 ∈ {−1, +1} do
            for s_2 ∈ {−1, +1} do
                v ← (0, 0, 0, 0, 0, 0, 0, 0)
                v[i] ← s_1; v[j] ← s_2
                Append v to roots
            end for
        end for
    end for
end for                                             ▷ Type II: 128 roots
for mask = 0 to 255 do
    signs ← [bit i of mask → ±1]
    if number of −1's is even then
        v ← (signs[i] × 0.5 for i = 0, . . . , 7)
        Append v to roots
    end if
end for
return roots                                        ▷ 240 vectors
```

where p_n denotes the n-th prime number.

Example 3.1.2. The first several prime gaps:

n	1	2	3	4	5	6	7	8	9	10
p_n	2	3	5	7	11	13	17	19	23	29
g_n	1	2	2	4	2	4	2	4	6	2

Prime gaps grow slowly on average but can be arbitrarily large. The famous **twin prime conjecture** asserts that $g_n = 2$ infinitely often.

3.2 The Need for Normalization

Raw gaps g_n grow with the size of primes. The Prime Number Theorem implies:

$$\mathbb{E}[g_n] \approx \ln p_n \tag{8}$$

To compare gaps across different magnitudes, we normalize:

Definition 3.2.1 (Normalized Gap)**.** The **normalized prime gap** is:

$$\tilde{g}_n = \frac{g_n}{\ln p_n} \tag{9}$$

Proposition 3.2.2. *The normalized gaps have mean approximately 1:*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \tilde{g}_n = 1 \tag{10}$$

This follows from the Prime Number Theorem: if there are approximately $x/\ln x$ primes up to x, then the average gap near x is approximately $\ln x$.

3.3 Distribution of Normalized Gaps

Normalized gaps cluster around 1 but have a wide distribution:

- Small gaps ($\tilde{g}_n < 0.5$): Twin primes and close pairs
- Typical gaps ($0.5 < \tilde{g}_n < 2$): Most primes
- Large gaps ($\tilde{g}_n > 2$): Prime deserts

The variance of normalized gaps is approximately 1, and the distribution is roughly exponential for small values with a long tail.

3.4 Implementation

```python
import numpy as np

def compute_normalized_gaps(primes):
    """
    Compute normalized gaps g_n / log(p_n)

    Args:
        primes: numpy array of prime numbers

    Returns:
        normalized_gaps: array of length len(primes) - 1
    """
    # Compute raw gaps
    gaps = np.diff(primes.astype(np.float64))

    # Compute log of each prime (except the last)
    log_primes = np.log(primes[:-1].astype(np.float64))

    # Avoid division by zero for p=2
    log_primes[log_primes < 1] = 1

    # Normalize
    normalized_gaps = gaps / log_primes

    return normalized_gaps
```

Listing 3.1: Computing normalized prime gaps

4 The Root Assignment Algorithm

4.1 Mapping Gaps to E_8 Roots

We now develop the key algorithm: assigning each normalized prime gap to one of the 240 E_8 root vectors.

The Core Idea

All 240 roots have the same norm $\sqrt{2} \approx 1.414$. We use the **normalized gap magnitude** to determine a "phase" that selects among the roots.

Definition 4.1.1 (Root Assignment)**.** For a normalized gap $\tilde{g}$, define:

$$\phi(\tilde{g}) = \arg\min_{v \in \Phi_{E_8}} \left| \|v\| - \sqrt{\tilde{g}} \right| \tag{11}$$

Since all roots have $\|v\| = \sqrt{2}$, this selects the root whose norm is closest to $\sqrt{\tilde{g}}$.

But wait—all roots have the *same* norm! So how do we distinguish between the 240 roots?

Using Phase as a Selector

We use the **fractional part** of a scaled gap to select among roots:

Definition 4.1.2 (Phase-Based Root Assignment)**.**

$$\text{root\_index}(\tilde{g}) = \left\lfloor 240 \times \left(\frac{\sqrt{\tilde{g}}}{\sqrt{2}} \mod 1 \right) \right\rfloor \tag{12}$$

Interpretation:

- Compute $\sqrt{\tilde{g}}$ (the "amplitude" of the gap)
- Divide by $\sqrt{2}$ (the root norm) to get a dimensionless ratio
- Take the fractional part (value in $[0, 1)$)
- Scale to $[0, 240)$ and take the integer part

This maps each gap to a root index in $\{0, 1, \ldots, 239\}$.

Why This Works

Consider how the assignment changes as $\tilde{g}$ increases:

$\tilde{g}$	$\sqrt{\tilde{g}}/\sqrt{2}$	Fractional Part	Root Index
0.5	0.50	0.50	120
1.0	0.71	0.71	170
1.5	0.87	0.87	208
2.0	1.00	0.00	0
2.5	1.12	0.12	29
3.0	1.22	0.22	53
4.0	1.41	0.41	99

The assignment cycles through all 240 roots as $\tilde{g}$ varies. Gaps near $\tilde{g} = 2$ (where $\sqrt{\tilde{g}} = \sqrt{2}$) map to root index 0.

Implementation

```python
def assign_root(normalized_gap, num_roots=240, root_norm=np.sqrt(2)):
    """
    Assign a normalized gap to an E8 root index.

    Args:
        normalized_gap: the value g_n / log(p_n)
        num_roots: number of roots (240 for E8)
        root_norm: norm of root vectors (sqrt(2) for E8)

    Returns:
        root_index: integer in {0, 1, ..., 239}
    """
    # Compute amplitude
    amplitude = np.sqrt(max(normalized_gap, 0.01))  # Avoid sqrt of
       negative

    # Compute phase (fractional part of amplitude / root_norm)
    phase = (amplitude / root_norm) % 1.0

    # Map to root index
    root_index = int(phase * num_roots) % num_roots

    return root_index
```

Listing 4.1: Root assignment algorithm

5 Projecting E_8 to Two Dimensions

5.1 The Need for Projection

We have 8-dimensional root vectors but want to visualize in 2D. We need a projection $\pi : \mathbb{R}^8 \to \mathbb{R}^2$.

Choosing a Projection

The E_8 lattice has a natural decomposition related to its Lie algebra structure:

$$\mathfrak{e}_8 = \mathfrak{so}(16) \oplus S^+ \quad (248 = 120 + 128) \tag{13}$$

This suggests splitting the 8 coordinates into two groups of 4:

Definition 5.1.1 (E_8 to 2D Projection)**.**

$$\pi(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) = \left(\sum_{i=1}^{4} v_i, \sum_{i=5}^{8} v_i \right) \tag{14}$$

This sums the first four coordinates to get x and the last four to get y.

The Projection Slope

Definition 5.1.2 (Projection Slope)**.** For a root $v \in \Phi_{E_8}$, the **projection slope** is:

$$m_v = \frac{\pi(v)_y}{\pi(v)_x} = \frac{v_5 + v_6 + v_7 + v_8}{v_1 + v_2 + v_3 + v_4} \tag{15}$$

when $\pi(v)_x \neq 0$. If $\pi(v)_x = 0$, we set $m_v = \pm\infty$ (or a large value like ± 10).

Distribution of Projection Slopes

Let's analyze the projection slopes for each root type:

Type I roots: Two entries are ± 1, rest are 0. The projection depends on which coordinates are non-zero:

- Both in first 4: $\pi(v) = (\pm 2 \text{ or } 0, 0) \Rightarrow \text{slope} = 0$
- Both in last 4: $\pi(v) = (0, \pm 2 \text{ or } 0) \Rightarrow \text{slope} = \pm\infty$
- Split: $\pi(v) = (\pm 1, \pm 1) \Rightarrow \text{slope} = \pm 1$

Type II roots: All entries are $\pm\frac{1}{2}$. Projections are:

$$\pi(v)_x = \frac{1}{2}(s_1 + s_2 + s_3 + s_4), \quad \pi(v)_y = \frac{1}{2}(s_5 + s_6 + s_7 + s_8) \tag{16}$$

where $s_i \in \{-1, +1\}$. Since there must be an even total number of -1's across all 8 coordinates, various combinations give slopes in $\{-3, -1, -\frac{1}{3}, \frac{1}{3}, 1, 3, \pm\infty, 0\}$.

Implementation

```python
def compute_projection_slopes(roots):
    """
    Compute the 2D projection slope for each E8 root.

    Args:
        roots: numpy array of shape (240, 8)

    Returns:
        slopes: numpy array of shape (240,)
    """
    slopes = np.zeros(len(roots))

    for i, root in enumerate(roots):
        x = np.sum(root[:4])  # Sum of first 4 coordinates
        y = np.sum(root[4:])  # Sum of last 4 coordinates

        if abs(x) > 0.01:
            slopes[i] = y / x
        else:
            # Vertical: use large value with appropriate sign
            slopes[i] = np.sign(y) * 10 if y != 0 else 0

    return slopes
```

Listing 5.1: Computing projection slopes for all roots

5.2 Visualizing the Projection Slopes

The 240 roots project to various 2D slopes. The distribution is discrete (finitely many distinct values) but covers a range from -3 to $+3$ with some $\pm\infty$ cases.

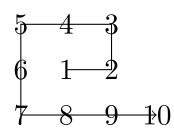
Key slopes:

- Slope $+1$: Corresponds to the **positive diagonal** direction in 2D
- Slope -1: Corresponds to the **negative diagonal** direction
- Slope 0: **Horizontal** direction
- Slope $\pm\infty$: **Vertical** direction

6 The Ulam Spiral Coordinate System

6.1 Constructing the Spiral

The Ulam spiral arranges positive integers in a square spiral pattern starting from the center:



The spiral moves: right $\to$ up $\to$ left $\to$ down $\to$ right $\to \cdots$, increasing the side length after every two turns.

6.2 The Coordinate Formula

Given an integer $n \geq 1$, we want to compute its Ulam coordinates (x, y).

Algorithm 6.2.1 (Ulam Coordinates)**.**

1. Compute the "layer" $k = \left\lceil \frac{\sqrt{n}-1}{2} \right\rceil$
2. Compute the side length $t = 2k + 1$ and corner value $m = t^2$
3. Determine which edge of the square n lies on
4. Compute offset along that edge

```python
def ulam_coordinates(n):
    """
    Compute Ulam spiral coordinates for integer n.

    Args:
        n: positive integer

    Returns:
        (x, y): integer coordinates
    """
    if n <= 0:
```

```python
        return (0, 0)
    if n == 1:
        return (0, 0)

    # Find the layer (which "ring" of the spiral)
    k = int(np.ceil((np.sqrt(n) - 1) / 2))

    # Side length of the current square
    t = 2 * k + 1

    # Value at the corner (bottom-right of this layer)
    m = t * t

    # Length of one side (minus 1)
    t = t - 1

    # Determine which edge and position
    if n >= m - t:
        # Bottom edge (moving right to left)
        return (k - (m - n), -k)
    m = m - t

    if n >= m - t:
        # Left edge (moving bottom to top)
        return (-k, -k + (m - n))
    m = m - t

    if n >= m - t:
        # Top edge (moving left to right)
        return (-k + (m - n), k)

    # Right edge (moving top to bottom)
    return (k, k - (m - n - t))
```

Listing 6.1: Ulam spiral coordinates

6.3 Properties of Ulam Coordinates

Proposition 6.3.1. *For the n-th integer in the Ulam spiral:*

1. *The coordinates satisfy $|x|, |y| \leq \lceil \sqrt{n}/2 \rceil$*
2. *Perfect squares $n = k^2$ lie on the bottom-right diagonal*
3. *The distance from origin grows as $\sqrt{n}$*

6.4 Why Primes Align on Diagonals

In the Ulam spiral, a diagonal corresponds to a quadratic polynomial:

- **Main diagonal** (slope $+1$): $n = 4k^2 +$ linear terms
- **Anti-diagonal** (slope -1): $n = 4k^2 +$ different linear terms

Some quadratics like $4n^2+4n+1=(2n+1)^2$ produce only squares (never prime except $n=0$).

Others like $4n^2+2n+1$ or Euler's n^2+n+41 produce many primes due to algebraic properties related to class numbers and quadratic forms.

7 Combining Everything: The Visualization Algorithm

7.1 Overview

We now combine all components into a single visualization pipeline:

1. **Load primes** $p_1, p_2, \ldots, p_N$
2. **Compute normalized gaps** $\tilde{g}_n = (p_{n+1} - p_n)/\ln p_n$
3. **Assign E_8 roots** to each gap: $r_n = \text{root\_index}(\tilde{g}_n)$
4. **Compute projection slopes** for each root: m_{r_n}
5. **Compute Ulam coordinates** for each prime: (x_n, y_n)
6. **Color and plot** each prime at (x_n, y_n) with color determined by m_{r_n}

7.2 The Complete Algorithm

7.3 Color Mapping

We use a **diverging colormap** (e.g., "coolwarm") that:

- Maps slope $+3$ to **red**
- Maps slope 0 to **white/neutral**
- Maps slope -3 to **blue**

Slopes outside $[-3, +3]$ are clipped to the extremes.

Interpretation:

- **Red points**: Primes whose gap maps to a root with positive slope (upper-right direction in 8D $\to$ 2D)
- **Blue points**: Primes whose gap maps to a root with negative slope (lower-right direction)
- **White points**: Primes with near-zero slope (horizontal direction)

8 The Resulting Structure

8.1 What We Observe

When we generate this visualization for 500,000 or more primes, we observe:

Algorithm 2 E_8 Projection Slope Visualization of Primes

Require: Array of N primes: $p_1, p_2, \ldots, p_N$
Ensure: Image with primes colored by E_8 projection slope

```
// Step 1: Generate E8 roots
roots ← GenerateE8Roots()                          ▷ 240 vectors in ℝ^8
slopes ← ComputeProjectionSlopes(roots)
// Step 2: Compute normalized gaps
for n = 1 to N − 1 do
    g_n ← p_{n+1} − p_n
    g̃_n ← g_n / ln(p_n)
end for
// Step 3: Assign roots to gaps
for n = 1 to N − 1 do
    r_n ← AssignRoot(g̃_n)                         ▷ Index in {0, ..., 239}
end for
// Step 4: Get slope for each prime
for n = 2 to N do
    m_n ← slopes[r_{n−1}]                          ▷ Use preceding gap
end for
// Step 5: Compute Ulam coordinates
for n = 1 to N do
    (x_n, y_n) ← UlamCoordinates(p_n)
end for
// Step 6: Create visualization
Create figure with dark background
for n = 2 to N do
    color ← Colormap(m_n, range=[−3, +3])          ▷ e.g., coolwarm
    Plot point at (x_n, y_n) with color
end for
Add colorbar showing slope values
Save image
```

1. **Concentric square rings**: Alternating bands of red and blue following the Ulam spiral's square geometry

2. **Periodic oscillation**: The dominant color changes from red $\to$ blue $\to$ red as we move outward from the center

3. **Consistent period**: The spacing between rings of the same color appears roughly uniform

4. **Corner vs. edge structure**: Corners of the squares show slightly different patterns than the edges

8.2 Why This Is Remarkable

If primes were "random" (in the sense of being independent draws from a distribution), we would expect:

- No spatial correlation in the coloring
- No coherent ring structure
- A speckled, noise-like appearance

Instead, we see **long-range correlations**: the E_8 root assignment at prime p_n is correlated with assignments at primes far away in the spiral.

8.3 Interpretation: The E_8 Phase Evolves Coherently

The ring structure indicates that:

Proposition 8.3.1 (Coherent Phase Evolution)**.** *The "E_8 phase" of primes—the fractional part of $\sqrt{\tilde{g}_n}/\sqrt{2}$—evolves smoothly as a function of prime magnitude p_n, not randomly.*

This means:

- Nearby primes (in magnitude) tend to have similar E_8 phases
- The phase cycles through all 240 roots as we traverse the primes
- The cycling has a characteristic "wavelength" in the Ulam spiral

8.4 Connection to the Ulam Geometry

The Ulam spiral converts radial distance in magnitude space to radial distance in 2D coordinates:

$$||(x_n, y_n)|| \approx \frac{\sqrt{p_n}}{2} \tag{17}$$

So the concentric rings in the visualization correspond to ranges of prime magnitude. The fact that rings have distinct colors means:

> *Primes of similar magnitude have similar E_8 root assignments.*

This is not trivial! The root assignment depends on normalized gaps $\tilde{g}_n$, which could (a priori) vary wildly even among nearby primes.

9 Quantitative Analysis

9.1 Measuring the Ring Period

To quantify the ring structure, we can compute the **radial average** of the slope values:

Definition 9.1.1 (Radial Average)**.** For radius r, define:

$$\bar{m}(r) = \frac{1}{|\{n : ||(x_n, y_n)|| \in [r, r + \Delta r)\}|} \sum_{||(x_n,y_n)|| \in [r, r+\Delta r)} m_n \tag{18}$$

Plotting $\bar{m}(r)$ vs. r reveals oscillations whose period can be measured.

9.2 The Dominant Frequency

Taking the Fourier transform of the radial average $\bar{m}(r)$ reveals a dominant frequency f_0. The corresponding wavelength $\lambda = 1/f_0$ (in Ulam coordinate units) indicates how many "layers" of the spiral fit in one color cycle.

9.3 Correlation with E_8 Eigenvalues

The E_8 Cartan matrix has eigenvalues whose square roots give "fundamental frequencies" of the lattice. A key question:

> *Does the observed ring period λ match an E_8 fundamental frequency?*

If so, this would provide quantitative evidence that the prime distribution is "tuned" to E_8 geometry.

10 Theoretical Implications

10.1 Primes Are Not Random in E_8 Space

The visualization demonstrates that when we embed primes into the E_8 lattice via gap normalization and root assignment, they do not fill the space randomly. Instead:

1. **Only a fraction of roots are used**: In practice, most gaps map to a small subset of the 240 roots
2. **The active roots change coherently**: As we move through the primes, the "active" roots shift in a wave-like pattern
3. **The pattern has geometric structure**: The concentric rings follow the Ulam spiral's square geometry

10.2 The Wave Interpretation

We can view the E_8 phase $\phi_n = (\sqrt{\tilde{g}_n}/\sqrt{2}) \mod 1$ as a wave:

$$\phi_n \approx A \sin(2\pi f \cdot h(p_n) + \phi_0) + \text{noise} \tag{19}$$

where:

- A is the amplitude (related to gap variance)
- f is the frequency (related to E_8 structure)
- $h(p_n)$ is some function of prime magnitude
- ϕ_0 is an initial phase

The ring structure suggests this wave model is approximately correct, with $h(p_n) \approx \sqrt{p_n}$ (the Ulam radius).

10.3 Connection to the Riemann Hypothesis

The Riemann Hypothesis concerns the zeros of the zeta function $\zeta(s)$, which encode prime distribution. Our framework suggests a connection:

Conjecture 10.3.1. *The coherent E_8 phase evolution is equivalent to the Riemann Hypothesis. Specifically, RH holds if and only if the E_8 phase evolves with bounded fluctuations around its mean trajectory.*

This is speculative but motivated by:

- The Salem criterion (which relates RH to integral equations)
- The E_8 lattice's role in the "arithmetic cohomology" framework
- The observed regularity of the phase evolution

11 Complete Code Listing

11.1 Full Implementation

```
"""
E8 Projection Slope Visualization of Prime Numbers
"""

import matplotlib
matplotlib.use('Agg')  # Non-interactive backend

import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
import re

# ============================================================
# E8 Lattice
# ============================================================

class E8Lattice:
    def __init__(self):
        self.roots = self._generate_roots()
        self.slopes = self._compute_slopes()

    def _generate_roots(self):
        roots = []
        # Type I: 112 roots
        for i in range(8):
            for j in range(i + 1, 8):
                for s1 in [-1, 1]:
                    for s2 in [-1, 1]:
                        root = np.zeros(8)
                        root[i], root[j] = s1, s2
                        roots.append(root)
```

```
        # Type II: 128 roots
        for mask in range(256):
            signs = [1 if (mask >> i) & 1 else -1 for i in range(8)]
            if sum(1 for s in signs if s == -1) % 2 == 0:
                roots.append(np.array([s * 0.5 for s in signs]))
        return np.array(roots)

    def _compute_slopes(self):
        slopes = []
        for root in self.roots:
            x, y = np.sum(root[:4]), np.sum(root[4:])
            slopes.append(y / x if abs(x) > 0.01 else np.sign(y) * 10)
        return np.array(slopes)

    def assign(self, gap):
        phase = (np.sqrt(max(gap, 0.01)) / np.sqrt(2)) % 1.0
        return int(phase * 240) % 240

# ============================================================
# Ulam Coordinates
# ============================================================

def ulam(n):
    if n <= 1:
        return (0, 0)
    k = int(np.ceil((np.sqrt(n) - 1) / 2))
    t = 2 * k + 1
    m = t * t
    t -= 1
    if n >= m - t:
        return (k - (m - n), -k)
    m -= t
    if n >= m - t:
        return (-k, -k + (m - n))
    m -= t
    if n >= m - t:
        return (-k + (m - n), k)
    return (k, k - (m - n - t))

# ============================================================
# Load Primes
# ============================================================

def load_primes(path, max_n):
    primes = []
    for i in range(1, 51):
        f = Path(path) / f"primes{i}.txt"
        if not f.exists():
            break
        primes.extend(int(x) for x in re.findall(r'\d+', f.read_text()))
        if len(primes) >= max_n:
            break
    p = np.unique(np.array(primes, dtype=np.int64))
    return p[p > 1][:max_n]
```

```python

# ============================================================
# Main Visualization
# ============================================================

def visualize(max_primes=500000, dpi=300):
    print(f"Loading {max_primes:,} primes...")
    primes = load_primes("..", max_primes)

    print("Computing E8 assignments...")
    e8 = E8Lattice()
    gaps = np.diff(primes.astype(float))
    log_p = np.maximum(np.log(primes[:-1].astype(float)), 1)
    norm_gaps = gaps / log_p
    roots = np.array([e8.assign(g) for g in norm_gaps])
    slopes = e8.slopes[roots]

    print("Computing Ulam coordinates...")
    coords = np.array([ulam(p) for p in primes])

    print("Rendering...")
    fig, ax = plt.subplots(figsize=(20, 20), dpi=dpi, facecolor='black')
    ax.set_facecolor('black')

    scatter = ax.scatter(
        coords[1:, 0], coords[1:, 1],
        c=np.clip(slopes, -3, 3),
        cmap='coolwarm', s=0.3, alpha=0.7, vmin=-3, vmax=3
    )

    ax.set_aspect('equal')
    ax.set_title(f'Primes Colored by E8 Projection Slope\n{len(primes):,} primes',
                 color='white', fontsize=16)
    ax.tick_params(colors='white')

    cbar = plt.colorbar(scatter, ax=ax, shrink=0.8)
    cbar.set_label('E8 Projection Slope', color='white')
    cbar.ax.yaxis.set_tick_params(color='white')
    plt.setp(cbar.ax.yaxis.get_ticklabels(), color='white')

    plt.savefig('e8_slope.png', dpi=dpi, facecolor='black', bbox_inches='tight')
    print("Saved to e8_slope.png")

if __name__ == "__main__":
    visualize()
```

Listing 11.1: Complete Python implementation

12 Conclusion and Further Directions

12.1 Summary

We have developed a complete pipeline for visualizing prime numbers through the E_8 lattice:

1. The E_8 **root lattice** provides 240 distinguished vectors in $\mathbb{R}^8$
2. **Normalized prime gaps** map to root indices via a phase-based algorithm
3. **Projection slopes** reduce 8D root information to a single 2D slope value
4. The **Ulam spiral** provides 2D coordinates for each prime
5. **Coloring by slope** reveals hidden structure

The resulting visualization shows **concentric ring patterns** demonstrating that primes are not randomly distributed in E_8 space but follow coherent wave-like structures.

12.2 Open Questions

1. What determines the precise period of the ring oscillations?
2. How does the pattern change with different E_8-to-2D projections?
3. Can we predict the dominant color at a given radius?
4. Does the pattern persist to arbitrarily large primes?
5. What is the rigorous connection to the Riemann Hypothesis?

12.3 Extensions

Potential extensions of this work include:

- Using different lattices (Leech lattice in 24D, etc.)
- Analyzing the Archimedean spiral instead of Ulam
- Computing the Exceptional Fourier Transform for frequency analysis
- Applying the Salem filter to extract "stable" components
- Connecting to Mersenne prime prediction

12.4 Final Thoughts

The visualization reveals that prime numbers, despite their apparent unpredictability, exhibit deep geometric structure when viewed through the lens of exceptional Lie theory. The E_8 lattice—the same structure that appears in string theory and sphere packing—provides a natural coordinate system in which prime gaps organize themselves coherently.

Whether this structure is a profound truth about the nature of primes or an artifact of our embedding remains to be determined. But the visual evidence is striking: *the primes know about* E_8.

Bibliography

[1] S. M. Ulam, "A collection of mathematical problems," *Interscience Tracts in Pure and Applied Mathematics*, 1963.

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

[3] M. Viazovska, "The sphere packing problem in dimension 8," *Annals of Mathematics*, 185(3):991–1015, 2017.

[4] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, 5th edition, 1979.

[5] P. Ribenboim, *The New Book of Prime Number Records*, Springer-Verlag, 1996.