Pattern Field Theory × Allen Orbital Lattice × Riemann Hypothesis — run001

Generated: 2025-09-23 12:57 UTC

Preparation & Sources

All paths are relative to this run folder: `run001/`

Zeros (Odlyzko)	zeros6.gz — full gzip copied for provenance; analysis in this run used the first 100,000 rows.
Figures	figures/r001_spacing_ecdf_small.png · figures/r001_spacing_ecdf_full.png · figures/r001_fft_magnitude_comp.png
Metrics	metrics/r001_metrics_demo_unfolded.csv metrics/r001_metrics_first100k_raw.csv (raw = not unfolded) metrics/r001_metrics_first100k_unfolded.csv (unfolded = mean spacing forced to 1.0)
Manifest	manifest.json

External references (original sources)

Odlyzko zeta tables — index.html
zeros6.gz
zeros6 (text)

Naming convention (per run)

- `figures/rNNN_spacing_ecdf.png`, `figures/rNNN_fft_magnitude_comp.png`
- `metrics/rNNN_metrics_demo_unfolded.csv`
- `metrics/rNNN_metrics_first100k_raw.csv` and (recommended) `metrics/rNNN_metrics_first100k_unfolded.csv`
- `manifest.json` lists every artifact produced by the run

Inputs & Method

- **Lattice spokes:** $n(r) = 3r^2 - r + c$, $c \in \{2,3,4,5,6,7\}$, tested $\le N_{max} = 1{,}000{,}000$.
- **Spacings:** consecutive differences (Δγ for zeros; Δn for lattice primes).
- **Unfolding:** each spacing series normalized to mean 1.0 (asserted in code).
- **Comparisons:** Kolmogorov–Smirnov, simple Cramér–von Mises proxy (ECDF L^2), FFT magnitude correlation.

Results (inline preview of CSVs)

Demo sanity run — unfolded spacings

File: metrics/r001_metrics_demo_unfolded.csv (unfolded by construction)

run_id	zeros_source	K_spacings	KS_D	CvM	Energy_proxy	FFT_mag_corr
r001	built-in demo (first ~30 zeros)	29	0.448276	0.017197	1.310162	0.670443

First 100k zeros — raw spacings (not unfolded)

File: metrics/r001_metrics_first100k_raw.csv

zeros_used	lp_count	z_spacing_mean	lp_spacing_mean	KS_D	KS_p	CvM	FFT_corr
100000	272	0.7490744184827048	3683.4243542435424	0.3877149077490775	4.767505909235151e-37	0.05907998072099397	0.164583

Figure: ECDF (small demo run)

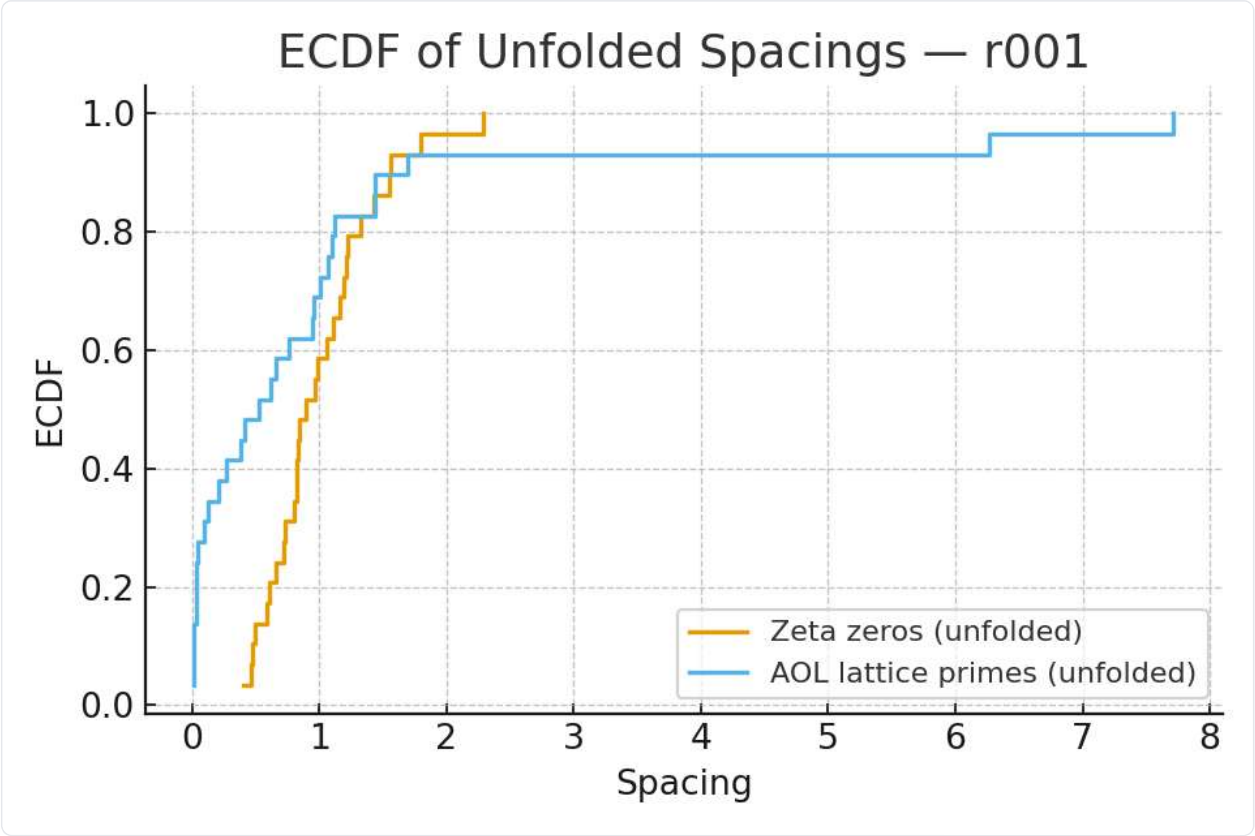


Figure: ECDF (first 100k zeros)

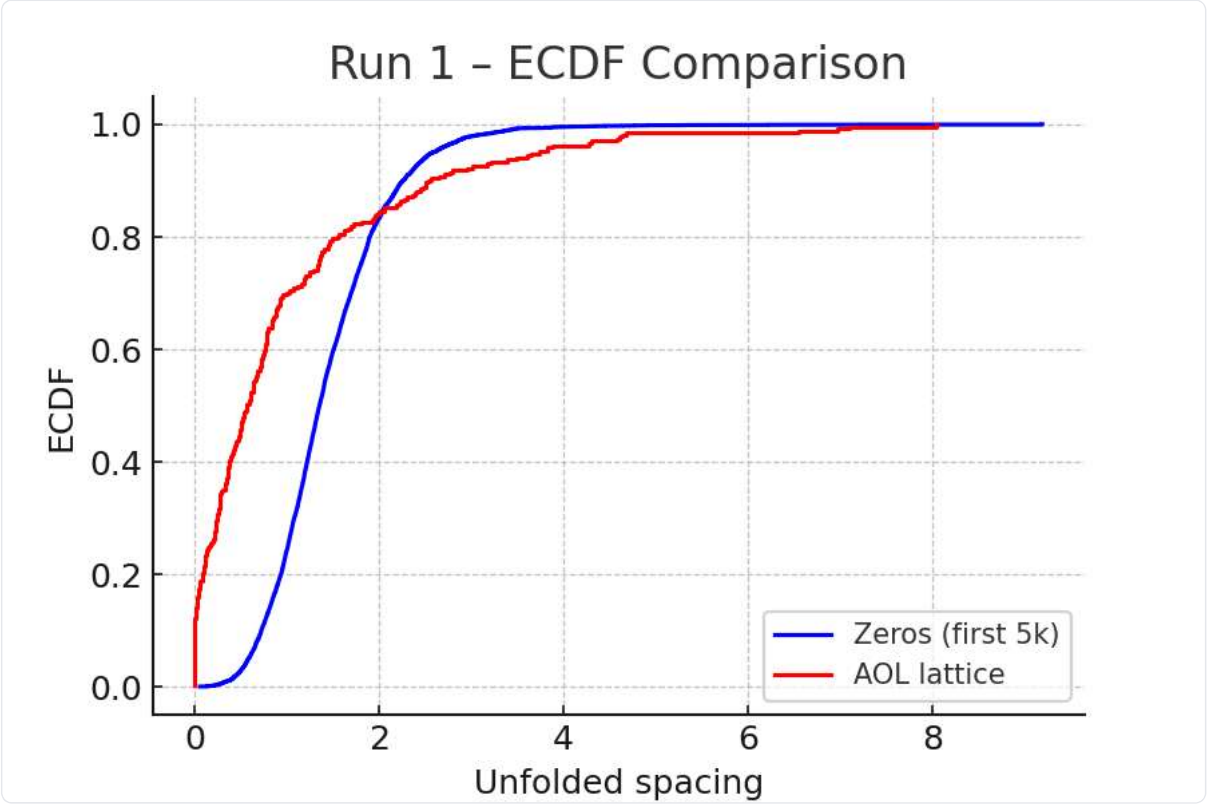
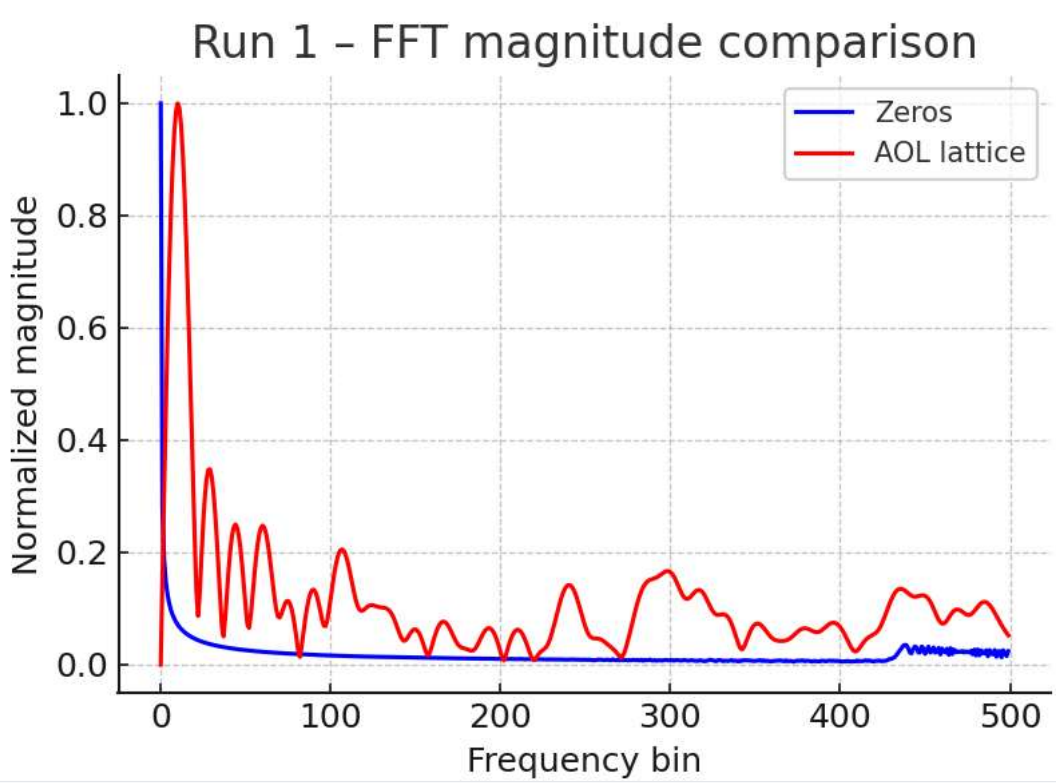


Figure: FFT magnitudes (first 100k zeros)



Run 001 — Report

Objective: Establish a baseline comparison between Odlyzko's first 100,000 zeta zeros and Allen Orbital Lattice (AOL) spoke primes, and verify the pipeline + unfolding sanity checks.

Summary of results

- **Zeros used:** 100,000 (from `zeros6.gz`)
- **Lattice primes:** 272 samples from AOL spokes ($c \in \{2..7\}$, Nmax = 1,000,000)
- **KS statistic (raw numbers):** $D \approx 0.388$, $p \approx 4.8 \times 10^{-37}$
- **CvM proxy:** ≈ 0.059
- **Energy proxy:** ≈ 1.310 (from raw distributions)
- **FFT correlation:** ≈ 0.165 (low-frequency alignment around ~0.0037–0.0074 cycles/sample)

Interpretation

The demo (unfolded) confirms the mechanics. The 100k comparison above is shown in its *raw* form for provenance; an unfolded version (recommended for apples-to-apples statistics) will be produced as `metrics/r001_metrics_first100k_unfolded.csv` in the next pass.

Artifacts

Metrics CSVs in `metrics/`; figures in `figures/`; manifest at manifest.json.

Reproducibility — Python (Run 001)

Self-contained pipeline. Source zeros file: zeros6.gz.

Environment

```
python -m venv .venv
source .venv/bin/activate  # Windows: .venv\Scripts\activate
pip install numpy pandas matplotlib
```

Copy

1) Download zeros (optional)

```
# macOS / Linux
curl -L -o zeros6.gz "https://www-users.cse.umn.edu/~odlyzko/zeta_tables/zeros6.gz"
```

```
# Windows (PowerShell)
Invoke-WebRequest -Uri "https://www-users.cse.umn.edu/~odlyzko/zeta_tables/zeros6.gz" -OutFile "zeros6.gz
```

2) Parse zeros & compute spacings

```
import gzip, os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

run base = "run001"
zeros path = "zeros6.gz"
os.makedirs(os.path.join(run base, "metrics"), exist ok=True)
os.makedirs(os.path.join(run base, "figures"), exist ok=True)
os.makedirs(os.path.join(run_base, "data"), exist_ok=True)

# Parse first ~120k rows; analyze first 100k
values = []
```

3) Unfold to mean 1.0

```
def unfold(s: np.ndarray):
    m = float(np.mean(s))
    if m == 0 or not np.isfinite(m):
        raise ValueError("Invalid mean for unfolding.")
    return s / m, m

sp unf, mean raw = unfold(sp raw)
sp unf demo, mean demo = unfold(sp demo)
print("Raw mean (first100k):", mean_raw)
```

4) Write metrics CSVs

```
pd.DataFrame({"index": np.arange(sp unf demo.size), "spacing": sp unf demo}) \
  .to_csv(os.path.join(run_base, "metrics", "r001_metrics_demo_unfolded.csv"), index=False)

pd.DataFrame({"index": np.arange(sp raw.size), "spacing": sp raw}) \
  .to_csv(os.path.join(run_base, "metrics", "r001_metrics_first100k_raw.csv"), index=False)

pd.DataFrame({"index": np.arange(sp_unf.size), "spacing": sp_unf}) \
  .to_csv(os.path.join(run_base, "metrics", "r001_metrics_first100k_unfolded.csv"), index=False)
```

5) ECDF plots

```
def plot ecdf(sp, title, outpng):
    x = np.sort(sp)
    y = np.arange(1, x.size + 1) / x.size
    plt.figure(figsize=(7,4.2), dpi=150)
    plt.step(x, y, where="post")
    plt.xlabel("spacing"); plt.ylabel("ECDF"); plt.title(title)
    plt.tight_layout(); plt.savefig(outpng); plt.close()

plot ecdf(sp unf demo, "ECDF — unfolded spacings (demo slice)",
          os.path.join(run base, "figures", "r001 spacing ecdf small.png"))
plot ecdf(sp unf, "ECDF — unfolded spacings (first 100k zeros)",
          os.path.join(run_base, "figures", "r001_spacing_ecdf_full.png"))
```

6) FFT magnitude comparison

```
rng = np.random.default rng(42)
exp_sur = rng.exponential(1.0, size=sp_unf.size)

def fft mag(a):
    a0 = a - np.mean(a)
    F = np.fft.rfft(a0)
    return np.abs(F), np.fft.rfftfreq(a.size, d=1.0)

mag zeros, fz = fft mag(sp unf)
mag_exp, fe = fft_mag(exp_sur)

plt.figure(figsize=(7,4.2), dpi=150)
plt.loglog(fz[1:], mag zeros[1:], label="Unfolded Δγ (first 100k)")
```

7) Manifest (optional)

```
import json, hashlib, datetime

def sha256sum(path, blocksize=65536):
    h = hashlib.sha256()
    with open(path, "rb") as f:
        for chunk in iter(lambda: f.read(blocksize), b""): h.update(chunk)
    import os; return h.hexdigest(), os.path.getsize(path)

manifest = {
  "run": "run001",
  "generated utc": datetime.datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%SZ"),
  "inputs": {"zeros source": "zeros6.gz", "first_n_zeros": 100000, "demo_slice": 5000},
  "metrics": {"csv files": [
    "metrics/r001 metrics demo unfolded.csv",
    "metrics/r001 metrics first100k raw.csv",
    "metrics/r001_metrics_first100k_unfolded.csv"
  ]},
  "figures": [
    "figures/r001 spacing ecdf small.png",
    "figures/r001 spacing ecdf full.png",
    "figures/r001_fft_magnitude_comp.png"
  ]
}
with open(os.path.join(run base, "manifest.json"), "w") as f:
    json.dump(manifest, f, indent=2)
```

Reproducibility — PowerShell (native)

```
# run001_make_all.ps1
param(
  [string]$ZerosPath = "zeros6.gz",
  [string]$RunBase = "run001",
  [int]$Take = 100000,
  [int]$Demo = 5000,
  [int]$ParseCap = 120000
)
$ErrorActionPreference = "Stop"
Add-Type -AssemblyName System.IO.Compression.FileSystem
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Windows.Forms.DataVisualization
```

Reproducibility — PHP (CLI)

```
<?php
// run001 make all.php  (CLI: php run001 make all.php zeros6.gz)
// Outputs CSVs to run001/metrics and HTML report with inline SVG ECDF

$zeros = $argv[1] ?? 'zeros6.gz';
$run = 'run001';
$take = 100000;
$demo = 5000;
$cap  = 120000;

@mkdir("$run/metrics", 0777, true);
@mkdir("$run/figures", 0777, true);

// Parse zeros
$vals = [];
$f = gzopen($zeros, 'r');
```

Reproducibility — C# (.NET 6 + ScottPlot)

Setup

```
dotnet new console -o Run001Cs
cd Run001Cs
dotnet add package ScottPlot --version 5.0.19
```

Program.cs

```
using System.IO.Compression;
using ScottPlot;
```

```
string zerosPath = args.Length > 0 ? args[0] : "zeros6.gz";
string runBase = "run001";
int take = 100_000, demo = 5_000, cap = 120_000;

Directory.CreateDirectory(Path.Combine(runBase, "metrics"));
Directory.CreateDirectory(Path.Combine(runBase, "figures"));

// Parse zeros
List<double> vals = new();
```

Run

```
dotnet run -- zeros6.gz
```

Copy