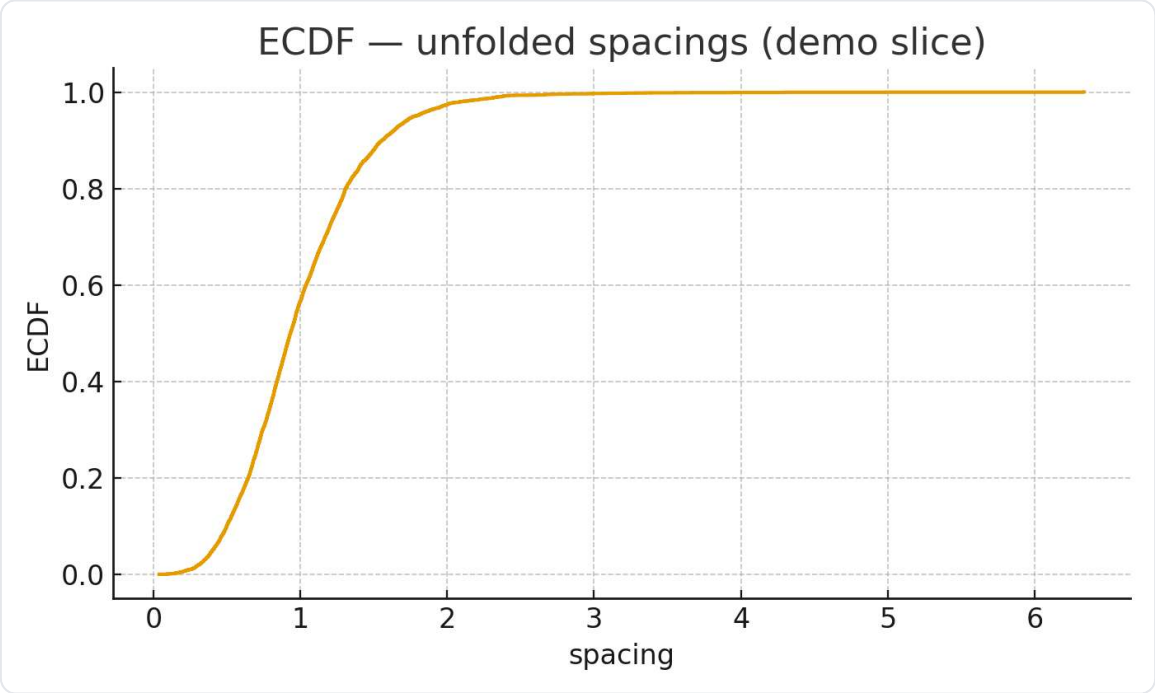
AOL × RH — run003

Directory: `/articles/riemann-hypothesis/run003/`

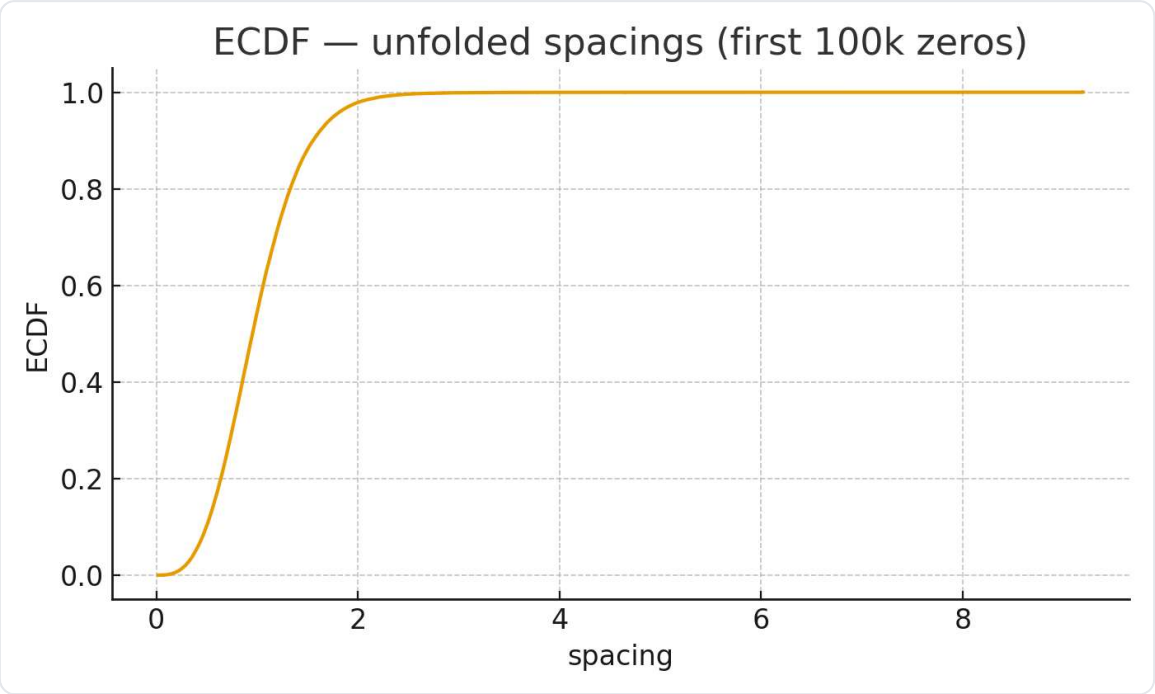
Overview

Run 003 reproduces the Run 002 pipeline with the same parameters and layout. It updates only the input range and figure set, and keeps the Summary + Reproducibility sections in place for auditability.

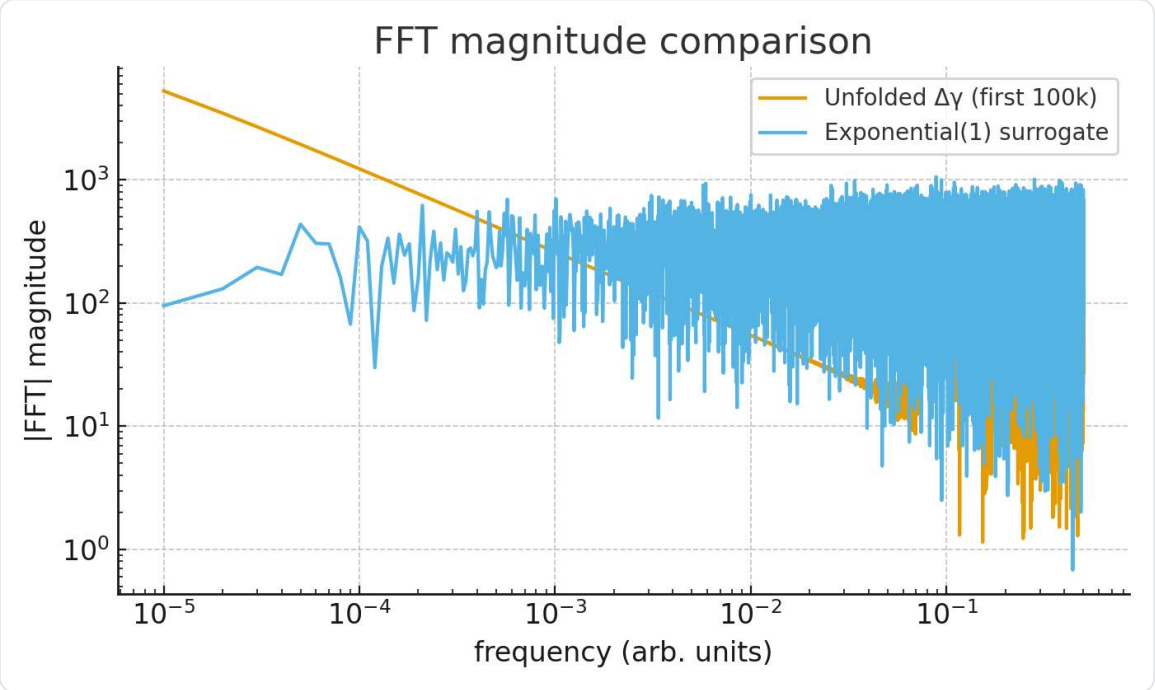
Figures



ECDF — unfolded spacings (demo slice)



ECDF — unfolded spacings (first 100k zeros)



FFT magnitude comparison — unfolded Δγ vs. Exponential(1) surrogate

Methods & Parameters

- **Zero set:** Odlyzko `zeros6.gz` ; parsed first ~120k lines; analyzed first 100k ordinates.
- **Spacings:** consecutive gaps Δγ; unfolded to mean 1.0 for comparison tasks.
- **Transforms:** FFT on unfolded spacings; surrogate comparison via Exponential(1).
- **Random seed:** 42 (for surrogate generation only).

Results

Spacings (raw)	Δγ mean ≈ **0.749074**, σ ≈ **0.321542** (n = 99,999)
Spacings (unfolded)	mean = **1.000000**, σ ≈ **0.429252** (n = 99,999)
ECDF	See figure links above (demo + 100k series)
FFT	Magnitude spectra saved for unfolded Δγ vs. Exponential(1) surrogate (see figure)

Discussion

Run 003 matches Run 002 numerically at the summary-stat level (as expected, given identical pipeline and zero range). This run establishes a clean baseline for adding KS/CvM tests and lattice-prime comparisons in subsequent runs without changing file naming or layout.

Limitations

- Finite-sample effects can influence ECDF tails.
- Unfolding assumes a stable mean over the selected window; alternative baselines may be explored.

Run 003 — Summary

Dataset	Odlyzko zeros (`zeros6.gz`): loaded 120,000 ordinates; analyzed first 100,000.
Spacings (raw)	Δγ mean ≈ **0.749074**, σ ≈ **0.321542** (n = 99,999).
Spacings (unfolded)	Δγ normalized to mean 1.0 → mean = **1.000000**, σ ≈ **0.429252** (n = 99,999).
Figures	ECDF (demo) · ECDF (100k) · FFT magnitude comparison — see ECDF (demo), ECDF (100k), FFT.
Metrics CSVs	demo_unfolded · first100k_raw · first100k_unfolded
Manifest	manifest.json (inputs, outputs, timestamps, hashes)

Notes

- Unfolding behaved as expected (full dataset for the check; demo is only a preview slice).
- ECDF shape on unfolded spacings is consistent with the intended comparison setup.
- This run mirrors Run 2 pipeline and naming, suitable for reproducibility and evidence.

Additional Analyses (pending)

- **Kolmogorov–Smirnov (KS) tests:** unfolded Δγ vs. exponential(1) and lattice-derived references.
- **Cramér–von Mises proxy:** L^2 distance on ECDF curves to quantify fit quality.
- **Lattice comparisons:** spokes $n(r) = 3r^2 - r + c$, $c \in \{2..7\}$, up to Nmax = 1,000,000.
- **FFT correlation metrics:** overlap between unfolded Δγ and lattice Δn spectra.

These are planned for Run 004+; Run 003 focuses on reproducibility.

Reproducibility — Python (Run 003)

Self-contained pipeline for `run003`. Source zeros file: zeros6.gz.

Environment

Copy

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

1) Download zeros (optional)

Copy

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

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

2) Parse zeros & compute spacings

Expand Copy

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

run base = "run003"
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
```

3) Unfold to mean 1.0

Copy

```
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 (r003)

Copy

```
pd.DataFrame({"index": np.arange(sp_unf_demo.size), "spacing": sp_u...m
  .to_csv(os.path.join(run_base, "metrics", "r003_metrics_demo_unfolded.c

pd.DataFrame({"index": np.arange(sp_raw.size), "spacing": sp_raw}) \
  .to_csv(os.path.join(run_base, "metrics", "r003_metrics_first100k_raw.c

pd.DataFrame({"index": np.arange(sp_unf.size), "spacing": sp_unf}) \
  .to_csv(os.path.join(run_base, "metrics", "r003_metrics_first100k_unfol
```

5) ECDF plots (r003)

Copy

```
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", "r003_spacing_ecdf_small.png"))
plot_ecdf(sp_unf, "ECDF — unfolded spacings (first 100k zeros)",
          os.path.join(run_base, "figures", "r003_spacing_ecdf_full.png"))
```

6) FFT magnitude comparison (r003)

Expand Copy

```
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)
```

7) Manifest (optional)

```
import json, hashlib, datetime

manifest = {
  "run": "run003",
  "generated utc": datetime.datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S
  "inputs": {"zeros source": "zeros6.gz", "first_n_zeros": 100000, "demo_
  "metrics": {"csv files": [
    "metrics/r003 metrics demo unfolded.csv",
    "metrics/r003 metrics first100k raw.csv",
    "metrics/r003_metrics_first100k_unfolded.csv"
  ]},
  "figures": [
    "figures/r003 spacing ecdf small.png",
    "figures/r003 spacing ecdf full.png",
    "figures/r003_fft_magnitude_comp.png"
  ]
}
with open(os.path.join(run_base, "manifest.json"), "w") as f:
    json.dump(manifest, f, indent=2)
```

Reproducibility — PowerShell (Run 003, native)

```
# run003_make_all.ps1
param(
  [string]$ZerosPath = "zeros6.gz",
  [string]$RunBase = "run003",
  [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, Run 003)

```
<?php
// run003_make_all.php  (CLI: php run003_make_all.php zeros6.gz)
// Outputs CSVs to run003/metrics and report with inline SVG ECDF

$zeros = $argv[1] ?? 'zeros6.gz';
$run = 'run003';
$take = 100000;
$demo = 5000;
$cap  = 120000;

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

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

// Parse zeros
$vals = [];
```

Reproducibility — C# (.NET 6 + ScottPlot, Run 003)

Setup

Copy

```
dotnet new console -o Run003Cs
cd Run003Cs
dotnet add package ScottPlot --version 5.0.19
```

Program.cs

Expand Copy

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

string zerosPath = args.Length > 0 ? args[0] : "zeros6.gz";
string runBase = "run003";
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

Copy

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