py-mathx-lab Documentation

Release 2.1.0

Walter Weinmann

Jan 14, 2026

GUIDE

CHAPTER

ONE

MATHEMATICAL EXPERIMENTATION

Mathematical experimentation is the practice of using examples, computation, and visualization to discover structure, to generate conjectures, find counterexamples, estimate quantities, and build intuition that later supports (or refutes) formal arguments.

It is *not* "proof by computer output". Instead, experiments are a disciplined way to ask better questions and to stress-test ideas-especially now that modern computers make it easy to explore large search spaces, high-precision numerics, and rich visualizations.

This repository, **py-mathx-lab**, is a small "lab notebook" of such experiments: compact, reproducible, and readable.

1.1 What counts as an "experiment" in mathematics?

An experiment is a finite procedure that produces evidence about a mathematical claim or object. Typical outcomes:

- **Conjecture generation**: patterns suggest statements that might be true (or false).
- **Counterexample search**: systematic exploration tries to break a hypothesis early.
- **Quantitative exploration**: estimate constants, rates, limits, or distributions.
- **Model checking**: validate (or invalidate) approximations and heuristics.
- **Visualization**: reveal structure that is hard to see symbolically.

The modern viewpoint-where computation is a genuine part of mathematical discovery-is widely discussed under the name *experimental mathematics* ([Bailey and Borwein, 2005, Bailey *et al.*, 2004, Borwein, 2005]). Some authors emphasize "plausible reasoning" supported by computation, paired with careful verification and eventual proof ([Borwein, 2009, Borwein and Bailey, 2008]). Other texts take a more problem-driven, exploratory style aimed at students and engineers ([Li *et al.*, 2003]), or present computation as a tool to formulate concrete research problems ([Arnold, 2015]).

1.2 Why computers change the game

Computers do not replace mathematical thinking-but they expand what is feasible to *inspect*:

- **Scale**: enumerate millions of cases (to find the first failure or build confidence).
- **Precision**: compute with high-precision floats or exact rationals/integers to avoid roundoff illusions.
- **Multiple lenses**: combine numerics, exact arithmetic, symbolic manipulation, and plotting.
- **Search**: automate discovery (parameter sweeps, optimization, random sampling, heuristics).
- **Reproducibility**: re-run the same pipeline with fixed seeds and pinned dependencies.

Used well, computation turns "I wonder if…" into "here is evidence, here are edge cases, and here is what we should prove next".

1.3 Experiments vs. proofs

A proof is the end of the story; an experiment is often the beginning.

Experiments are excellent for:

- **disproving** statements (one counterexample ends it),
- identifying **what is actually true** (after a naive conjecture fails),
- suggesting **lemmas** and **invariants** that make a proof possible.

But experiments can also mislead. Common failure modes:

- floating point error and catastrophic cancellation,
- plotting artifacts,
- "pattern matching" based on too few samples,
- unintentional selection bias ("I tried the cases that worked").

The goal is to use experiments to *reduce uncertainty*, not to hide it.

1.4 Targets for py-mathx-lab

py-mathx-lab aims to be a practical, long-lived collection of experiments with shared conventions:

1. **Reproducible runs**
   Each experiment is runnable as a module, writes results to a single output directory, and (when relevant) uses a fixed seed.
2. **Readable code**
   The code should be short, well-typed, and structured so readers can modify it.
3. **Useful artifacts**
   Each experiment should generate at least one of:
   - a figure
   - a table / summary statistics
   - a counterexample / witness object
   - a short narrative explaining what was learned
4. **Clear boundaries**
   An experiment write-up should state:
   - what is being tested,
   - what counts as "success" or "failure",
   - what might invalidate the result (precision limits, domain constraints, runtime limits).
5. **Traceability**
   Each page includes references to books/papers that motivated the work or explain the background.

1.5 Repository conventions for experiments

An experiment page should usually include:

- **Goal** (one paragraph)
- **How to run** (a command that works from the repo root)
- **Parameters** (including defaults and ranges, if swept)
- **Results** (figures/tables and a short interpretation)

- **Notes / pitfalls** (numerical caveats, surprising behavior)
- **References** (bibliography keys)

In code, prefer:

- deterministic outputs (fixed seeds, stable sorting),
- explicit configuration objects / CLI arguments,
- sanity checks (dimension checks, bounds checks, invariants),
- cross-checks (e.g., float vs. exact, two independent formulas).

1.6 Examples of good "experiment themes"

The experiment format supports many domains, for example:

- **Numerical analysis**: error landscapes, stability regions, conditioning, Monte Carlo integration.
- **Number theory**: continued fractions and convergents, integer sequences, modular patterns, primality heuristics.
- **Geometry/topology**: random point clouds, curvature approximations, combinatorial invariants.
- **Optimization/probability**: stochastic search behavior, concentration phenomena, empirical distributions.

A concrete example of a rich, experiment-friendly topic is continued fractions, where it is natural to compute and plot convergents, partial quotient statistics, and periodicity phenomena ([Borwein *et al.*, 2014]).

Next steps: start with *Getting started*, browse the *Valid Tags* directory, and then browse the *Experiments Gallery* gallery.

CHAPTER

TWO

GETTING STARTED

This project uses an **uv-only** workflow and a small Makefile wrapper to run everything consistently.

2.1 Prerequisites

You need:

- **Python 3.14**
- **uv** on your PATH
- **GNU Make**

2.1.1 Windows notes

- Install GNU Make (e.g., via Chocolatey).
- You can run everything from `cmd.exe`, PowerShell, or from WSL.

2.2 Verify tools

From the repository root:

```
make uv-check
make python-check
make python-info
```

2.3 First-time setup

Create a virtual environment and install dependencies:

```
make venv
make install-dev
make install-docs
```

2.4 Run the full development chain

```
make final
```

`make final` runs:

- formatting (Ruff)
- linting (Ruff)
- type checking (mypy)

- tests (pytest)
- documentation (sphinx)

Documentation can be built separately via:

```
make docs
```

2.5 Run an experiment

Example (E001):

```
make run EXP=e001 ARGS="--seed 1"
```

See the experiment overview here: *Experiments Gallery*.

2.6 Build documentation locally

```
make docs
```

Output:

- `docs/_build/html`

2.7 Troubleshooting

2.7.1 `error: Failed to spawn: sphinx-build`

Sphinx is not installed in the uv environment.

Fix:

```
make install-docs
make docs
```

2.7.2 `make python-check` fails

The repo enforces Python **3.14**. Install Python 3.14, then recreate the environment:

```
make clean-venv
make venv
make install-dev
make install-docs
```

CHAPTER

THREE

DEVELOPMENT

This page describes the development workflow and the conventions used in this repository.

3.1 Workflow overview

Use the Makefile targets for everything. They wrap `uv run ...` so you do not need to activate a virtual environment manually.

Typical day-to-day:

```
make final
```

Documentation-only build:

```
make docs
```

Clean caches and build artifacts:

```
make clean
```

Remove the virtual environment (full reset):

```
make clean-venv
```

3.2 Makefile workflow

The Makefile is the **single entry point** for development tasks (env setup, quality checks, docs builds, and running experiments).

- Prefer `make final` before pushing.
- Prefer `make docs` to validate documentation changes.
- Use `make run EXP=<id>` to execute an experiment and write artifacts to `out/<id>/`.

3.2.1 Dependency groups

This summary is included from the Makefile documentation:

`uv` installs dependencies from your `pyproject.toml`. This project uses "extras":

- **default**: runtime dependencies (needed to run the package)
- **dev**: developer tools (ruff, mypy, pytest, …)
- **docs**: documentation tools (sphinx, theme, myst, bibtex, …)

3.2.2 What the Makefile uses

- Most developer commands run via:

```
uv run --extra dev ...
```

- Documentation build runs via:

```
uv run --extra docs ...
```

3.2.3 Why `docs-deps` uses `--all-extras`

`make docs-deps` runs:

```
uv sync --all-extras
```

because the docs pipeline also runs a small helper under the **dev** extra before invoking Sphinx.

3.3 Virtual environment behavior

3.3.1 Where is the venv?

The venv is always created at:

- `.venv/`

3.3.2 Minimum Python version

The Makefile checks `PYTHON_MIN` (default is `3.14`). If your Python is older, `make python-check` fails.

3.3.3 Why `UV_LINK_MODE=copy` exists

On Windows, hardlinks can fail or warn on multi-drive setups (e.g. repo on `D:` but cache on `C:`). So the Makefile defaults to:

- `UV_LINK_MODE=copy`

You *can* override it, but the default is chosen to reduce Windows pain.

3.3.4 Common "reset" if your venv is broken

```
make clean-venv
make install-dev
```

3.4 Formatting (ruff)

Formatting is purely about **how code looks**, not what it does.

3.4.1 Targets

Target	What it does	When to use it
`make format`	formats selected repo paths	normal formatting
`make format-check`	checks formatting only (no changes)	CI-like check
`make fmt`	"broad auto-fix": ruff fixes + formats everything	when you want the repo cleaned up quickly

3.4.2 Typical usage

Before committing:

```
make fmt
```

If CI says "formatting changed":

```
make format
```

3.5 Linting (ruff)

Linting looks for **potential bugs and bad patterns**, for example:

- unused imports
- variables shadowing names
- common correctness issues ruff knows how to detect

3.5.1 Targets

Target	What it does	Does it edit files?
`make lint`	ruff check-only	no
`make lint-fix`	ruff with auto-fix	yes

3.5.2 Typical usage

- Use `make lint` when you only want to see problems.
- Use `make lint-fix` when you want ruff to fix safe issues automatically.

3.6 Typing (mypy)

Typing checks help catch issues like:

- calling functions with wrong argument types
- returning the wrong types
- forgetting to handle `None`

3.6.1 Target

```
make mypy
```

It runs:

- `mypy mathxlab tests experiments`

3.6.2 Common tip for juniors

If mypy errors look scary, start with the first error in the output. Many later errors are "follow-up noise" caused by an earlier wrong type.

3.7 Tests (pytest): fast / slow / perf

This repo separates tests using **pytest markers**:

- **fast tests**: `not slow and not perf`
- **slow tests**: `slow and not perf`
- **perf tests**: `perf`

Markers are set in tests like:

```
import pytest

@pytest.mark.slow
def test_big_case():
    ...
```

3.7.1 Coverage and threshold

Tests collect coverage for the library packages (not for experiment scripts). Coverage fails the target if it drops below **80%**.

3.7.2 Stable temp paths

pytest is invoked with stable temp directories:

- `temp_pytest_cache`
- `temp_pytest`

They are cleaned by `make clean`.

3.7.3 `pytest` (fast tests)

```
make pytest
```

What it does:

- runs only **fast** tests (`not slow and not perf`)
- runs with coverage
- fails if coverage $< 80\%$

This is the main "developer loop" test target.

3.7.4 `pytest-xdist` (fast tests in parallel)

```
make pytest-xdist
```

Runs fast tests using xdist:

- `-n auto --dist=load`

Use this when the test suite gets bigger and you want faster local feedback.

3.7.5 `pytest-slow` (two-phase: fast then slow)

```
make pytest-slow
```

What it does (important detail):

1. deletes `.coverage`
2. runs **fast tests** first *in best-effort mode* (failures in this phase do **not** fail the target — this is intentional in the Makefile)
3. runs **slow tests** with:
   - xdist by default (`PYTEST_XDIST_SLOW=-n auto --dist=load`)
   - `--cov-append` to combine coverage from both phases
   - coverage threshold (80%)

> **Warning**
>
> If you want "fast tests must pass", run `make pytest` separately. `make pytest-slow` is designed to always get through the slow suite even if fast tests are currently failing.

3.8 Performance tests (pytest-perf): what they are and how to use them

Performance tests are still **pytest tests**, but marked with `@pytest.mark.perf`. They exist to catch **accidental slowdowns** (e.g. a function becomes 10× slower).

3.8.1 Why the Makefile forces thread counts to 1

Math/scientific libraries sometimes use multiple CPU threads automatically (BLAS/OpenMP/etc.). That makes timings noisy and hard to compare.

So perf targets set:

- `OMP_NUM_THREADS=1`
- `MKL_NUM_THREADS=1`
- `OPENBLAS_NUM_THREADS=1`
- `NUMEXPR_NUM_THREADS=1`

This makes timings **more reproducible** across machines and runs.

3.8.2 `pytest-perf` (run perf-marked tests)

```
make pytest-perf
```

Runs only:

- `-m "perf"`

and shows progress output.

Use this when:

- you changed a performance-sensitive function
- you want to ensure you didn't introduce an obvious regression

3.8.3 `pytest-perf-baseline` (update accepted baseline numbers)

```
make pytest-perf-baseline
```

Same as `pytest-perf`, but also passes:

- `--perf-update-baseline`

Use this **only** when:

- you intentionally changed performance (e.g. algorithm changed)
- and you want to accept the new timings as the baseline

> **Warning**
>
> Baseline updates should be done on a reasonably idle machine. If you run baseline updates while your system is busy, you may "bake in" slow/noisy numbers.

3.9 Performance suite (non-pytest): snapshots and comparisons

In addition to `pytest-perf`, the Makefile also provides a small "perf runner" pipeline:

3.9.1 `perf` (dev snapshot)

```
make perf
```

Runs:

- `python mathxlab/tools/run_perf.py --mode dev --overwrite`

This is typically used during development to write/update a **performance snapshot**.

3.9.2 `perf-release` (release snapshot)

```
make perf-release
```

Runs:

- `python mathxlab/tools/run_perf.py --mode release --overwrite`

This is usually for "release-ish" measurements (more stable, fewer surprises).

3.9.3 `perf-compare` (compare two snapshots)

```
make perf-compare A=v0.1.0 B=v0.2.0
```

Runs:

- `python mathxlab/tools/compare_perf.py --a … --b …`

Use this when you want a readable report of "what got faster/slower" between two saved snapshots.

> **Tip**
>
> If you're not sure what valid values for `A` and `B` are, run `make perf` once and look at the output written by the script. It usually prints the snapshot identifiers/paths it created.

3.9.4 Run logs

Experiments are Python modules like:

- `mathxlab/experiments/e001.py`
- `mathxlab/experiments/e002.py`
- …

3.9.5 Run one experiment (recommended: via Make)

```
make run EXP=e001
```

This:

- creates `out/e001/` (if missing)
- creates `out/e001/logs/`
- writes a log file:
  - `out/e001/logs/run_e001.log`
- runs the experiment with:
  - `python -m mathxlab.experiments.e001 --out out/e001 -v`

3.9.6 Forward CLI arguments

```
make run EXP=e001 ARGS="--seed 123 --n 200000"
```

> **Note**
>
> On Windows, always quote `ARGS="..."` if it contains spaces.

3.9.7 Run all experiments

```
make out
```

This finds all `mathxlab/experiments/e???.py` files and runs them sequentially.

3.9.8 Full reference

For the complete target-by-target reference and troubleshooting, see makefile.

3.10 Formatting, linting, typing, tests

- Formatting: **Ruff formatter**
- Linting: **Ruff**
- Typing: **mypy**
- Tests: **pytest**

3.10.1 CI formatting behavior

In CI, formatting runs in check mode (`ruff format --check`). Locally it formats in place.

3.11 Experiment authoring guidelines

When adding a new experiment:

1. Add a new module under `mathxlab/experiments/`, e.g. `e002_...py`.
2. Prefer deterministic outputs:
   - `--seed` argument if randomness is involved
   - write results to a single `--out` directory
3. Keep the experiment runnable as a module:
   - `python -m mathxlab.experiments.e002`
4. Update the docs:
   - add a short entry to *Experiments Gallery*
   - optionally add a dedicated page under `docs/experiments/` later

3.11.1 Report contract for algorithmic experiments (Phase 2 and later)

For experiments involving algorithms (primality tests, factorization, explicit bounds), the `out/e###/report.md` file is part of the experiment's *scientific contract*.

It must state (when applicable):

- **Deterministic vs probabilistic.** Always label this explicitly.
- **Probability of error** for probabilistic methods (bases / repetitions).
- **Correctness cross-checks** against a trusted reference for CI-safe ranges.
- **Known counterexamples / failure modes** (e.g., Carmichael numbers for Fermat).
- **Finite-range behavior** vs asymptotics (do not oversell small N).

Use this drop-in template for report sections (copy/paste into `report.md` or generate it in code):

3.12 Algorithmic guarantees

- **Method:** (name the algorithm)
- **Status:** `DETERMINISTIC` | `PROBABILISTIC`

3.12.1 If probabilistic

- **Randomness / bases:** (list bases used or how randomness was sampled)
- **Conservative error statement:**
  - For **Miller-Rabin**, a common bound is: `P(false prime) <= 4^{-k}` after `k` independent random bases.
  - If you use a **fixed base set** (engineering choice), state the intended input range.

3.13 Correctness cross-check

State how you validated correctness for a CI-safe range.

- **Reference:** (e.g., sieve ground truth, deterministic trial division)
- **Checked range:** (e.g., `n <= 1_000_000`)
- **Result:** mismatches = `0` (or list the smallest mismatch as a witness)

3.14 Known counterexamples / failure modes (when applicable)

Use this section when the method is known to fail on structured inputs.

- **Fermat test:** Carmichael numbers pass for all coprime bases (smallest: `561`).
- **Fermat base-2 pseudoprime:** `341 = 11 * 31` passes `2^(n-1) mod n = 1` but is composite.
- **Miller-Rabin:** specific bases can be fooled by strong pseudoprimes (state the bases used).
- **Pollard rho:** may stall for unlucky seeds; retries and parameter changes are expected.

3.15 Runtime knobs (CI-safe)

List the knobs that keep runtime bounded in CI.

- `n_max`: (upper bound)
- `sample_size`: (how many candidates were tested)
- `max_rounds` / `max_retries`: (for randomized algorithms)
- `seed`: (if randomness is involved)

3.16 Finite-range behavior (for asymptotics / explicit bounds)

When referencing an asymptotic statement (e.g., PNT), explicitly separate:

- **Theory statement:** what is true as `x -> infinity`.
- **Finite range used here:** `x in [A, B]`.
- **Where it becomes meaningful:** state a measurable criterion (e.g., relative error $< 5\%$) and the smallest `x` where that holds.

3.17 Documentation

Docs are built with Sphinx + MyST.

Build locally:

```
make install-docs
make docs
```

Deployed website:

- GitHub Pages from the `docs` workflow

3.18 Contributing (high-level)

- Create a feature branch.
- Open a PR against `main`.
- CI must pass before merge.
- Keep PRs small and well-scoped.

CHAPTER

FOUR

VALID TAGS

This page defines the allowed tags for experiments in **py-mathx-lab**. Tags are used in the *Experiments Gallery* and individual experiment pages to categorize content.

4.1 Primary Tags (Domains)

These represent the broad mathematical area of the experiment.

Tag	Description
`analysis`	Calculus, real/complex analysis, limits, and approximation.
`aps`	Arithmetic Progressions and related theorems.
`conjecture-generation`	Patterns suggest statements that might be true (or false).
`counterexample-search`	Systematic exploration tries to break a hypothesis early.
`dirichlet-characters`	Dirichlet characters modulo q, orthogonality, and primitive characters.
`l-functions`	Dirichlet L-functions and related analytic objects controlling prime distribution.
`model-checking`	Validate (or invalidate) approximations and heuristics.
`number-theory`	Properties of integers, divisibility, and prime numbers.
`prime-races`	Prime number races (e.g., Chebyshev bias) comparing counts in residue classes.
`quantitative-exploratic`	Estimate constants, rates, limits, or distributions.
`visualization`	Reveal structure that is hard to see symbolically.

4.2 Secondary Tags (Topics & Methods)

These provide more specific detail about the techniques or subtopics involved.

Tag	Description
`arithmetic-functions`	Classical functions on integers (e.g., ϕ, μ, σ, τ) and their relations.
`bounds`	Explicit mathematical bounds (e.g., on prime-counting functions).
`carmichael-lambda`	Carmichael's lambda function (n), the exponent of (Z/nZ)*.
`carmichael`	Carmichael numbers (absolute Fermat pseudoprimes).
`classification`	Grouping objects into classes based on shared properties.
`critical-line`	Zeta/L-function values on Re(s)=1/2 and related numerics.
`dirichlet-convolution`	Dirichlet convolution of arithmetic functions and identity checks.
`dirichlet-series`	Dirichlet generating functions and related analytic tools.
`divisor-function`	Divisor functions such as (n)=d(n) and (n), including record behavior.
`explicit-formula`	Explicit formulas connecting primes and zeros ((x), (x), etc.).
`exploration`	Open-ended search for patterns or properties.
`factorization`	Integer factorization methods and hardness.
`fermat`	Fermat numbers and Fermat primes.

continues on next page

Table 1 – continued from previous page

Tag	Description
`gaps`	Studies of the distribution of gaps between primes.
`generating-functions`	Ordinary/exponential generating functions (esp. partitions).
`gram-points`	Gram points and Gram's law heuristics for zeta zeros.
`hardy-z`	Hardy's Z-function and Riemann–Siegel theta function.
`heuristics`	Probabilistic or empirical models (e.g., Cramér's model for primes).
`li-x`	Logarithmic integral li(x) and its relation to $\pi(x)$.
`liouville`	Liouville function (n) and related parity questions for Ω(n).
`lucas-lehmer`	Lucas–Lehmer primality test for Mersenne numbers.
`mangoldt`	von Mangoldt function Λ(n) and related Chebyshev functions (x), (x).
`mersenne`	Mersenne numbers $M_p = 2^p - 1$ and Mersenne primes.
`miller-rabin`	Miller–Rabin primality test and its counterexamples.
`mobius`	Möbius function $\mu(n)$, Möbius inversion, squarefree indicators.
`multiplicative`	Multiplicative arithmetic functions; Dirichlet convolution viewpoints.
`numerics`	Heavy use of floating-point or high-precision computation.
`omega`	Prime-factor counting functions (n) and Ω(n) (distinct vs with multiplicity).
`open-problems`	Related to famous unproven conjectures.
`optimization`	Finding maxima, minima, or best-fit parameters.
`partition`	Partition function p(n), identities, and asymptotics.
`perfect`	Related specifically to perfect, abundant, or deficient numbers.
`pnt`	Prime Number Theorem and related asymptotics.
`primes`	Prime number distribution, density, and related theorems.
`primorial`	Primorials, Euclid numbers, primorial primes.
`pseudoprime`	Pseudoprimes, primality-test failures.
`pseudoprimes`	Collective study of various types of pseudoprimes.
`riemann-zeta`	Riemann zeta function (s): series, Euler product, analytic continuation, zeros.
`search`	Systematic search through a large state space.
`semiprime`	Semiprimes, RSA-type composites.
`sieving`	Sieve methods (Eratosthenes, Sundaram, Atkin, etc.).
`sigma`	Related to the sum-of-divisors function $\sigma(n)$.
`summatory`	Summatory functions (e.g., Mertens $M(x)$, summatory totient $\Phi(x)$).
`taylor`	Related to Taylor series and their approximations.
`totient`	Euler's totient function $\phi(n)$, totient equations, summatory behavior.
`twin`	Twin primes and the twin prime conjecture.
`klauber`	Klauber triangle and prime patterns between consecutive squares.
`sacks`	Sacks spiral ($r = \sqrt{n},\ \theta = 2\pi\sqrt{n}$) and related prime patterns.
`hex-spiral`	Integer spirals on a hexagonal lattice (hex-grid prime maps).
`ulam`	Ulam spiral and related prime patterns in 2D.
`wieferich`	Wieferich primes and related congruences.
`wilson`	Wilson's theorem and Wilson primes.
`zeta-zeros`	Nontrivial zeros, zero-counting $N(T)$, root bracketing.

4.3 Usage

When adding a new experiment:

1. Choose at least one **Primary Tag** (Domain or Type).
2. Choose one or more **Secondary Tags** (Topics & Methods).
3. Add them to the `**Tags:**` line in your `.md` file.
4. Update the *Experiments Gallery* using the corresponding CSS classes (`tag-primary` for primary tags, `tag-secondary` for secondary tags).

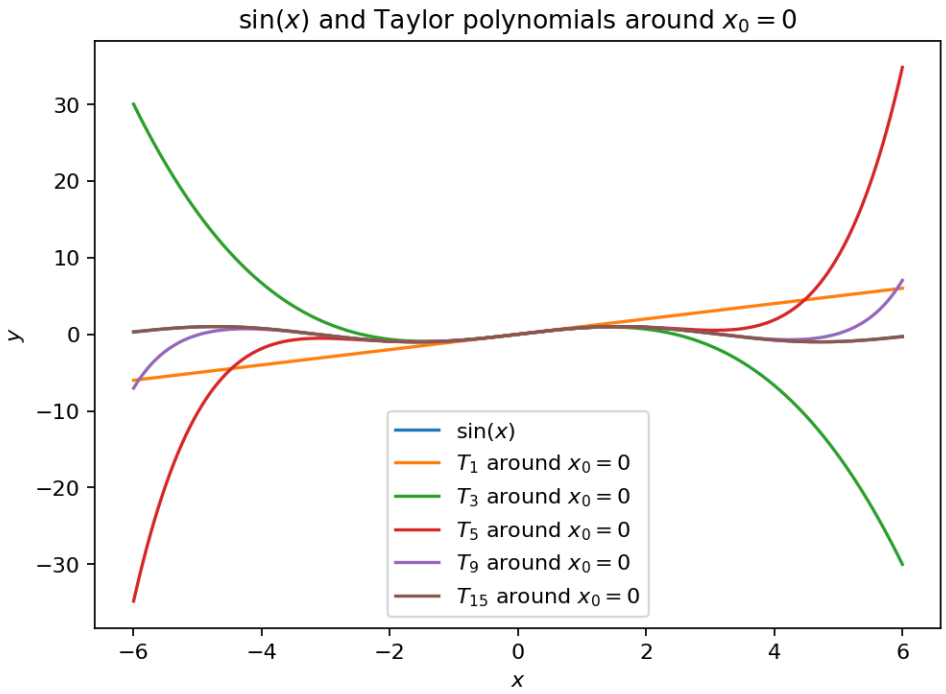
CHAPTER

FIVE

EXPERIMENTS GALLERY

A compact, image-first overview of the experiments in **py-mathx-lab**.

5.1 E001: Taylor Error Landscapes



Tags: `analysis`, `quantitative-exploration`, `visualization`, `numerics`, `taylor` See: *Valid Tags*.

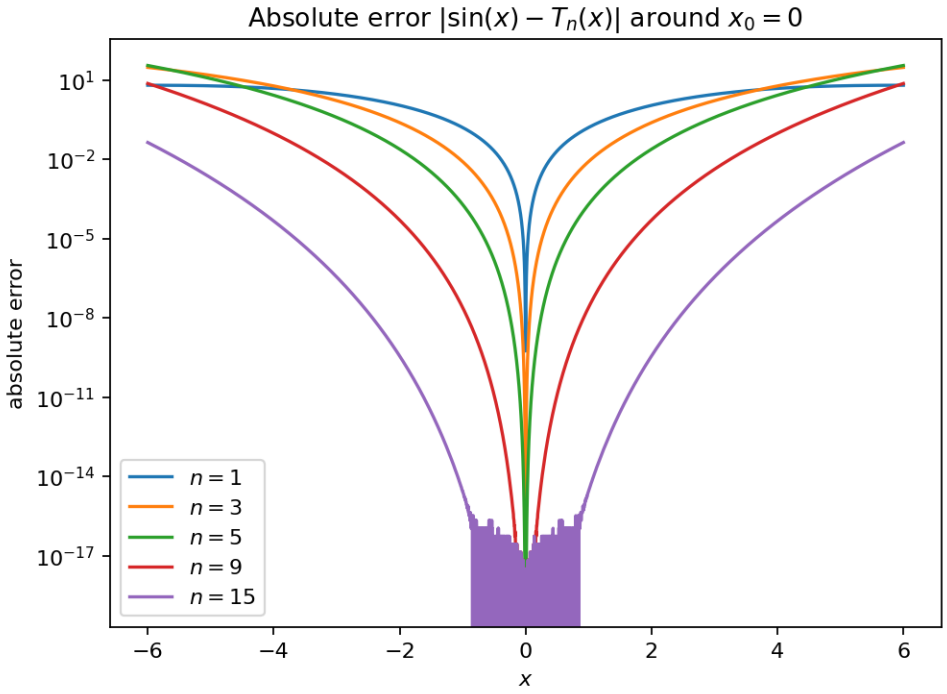
5.1.1 Goal

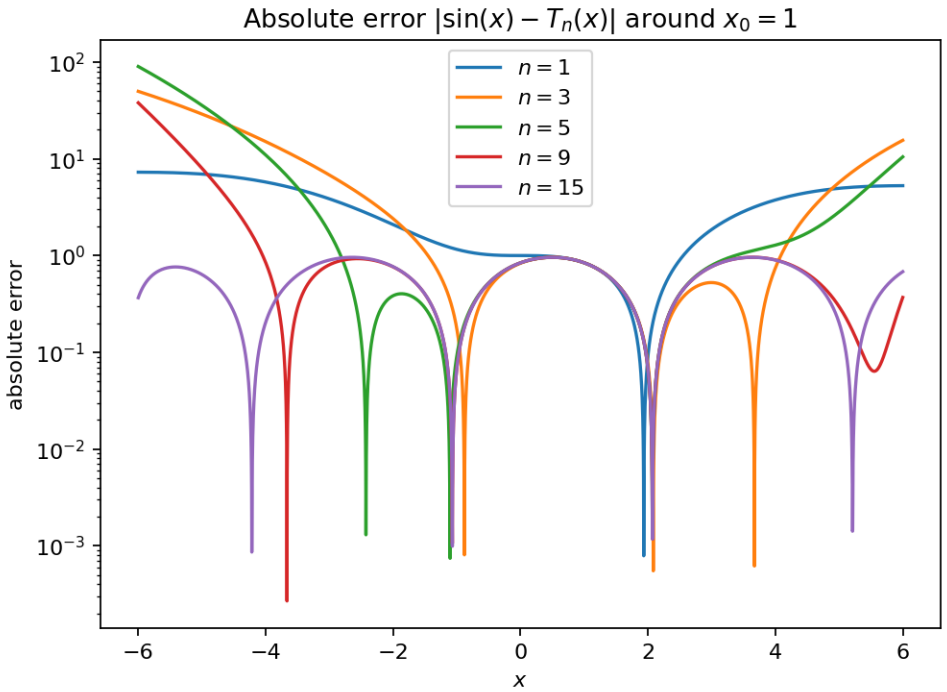
Build intuition for Taylor truncation error by visualizing the absolute error landscape $|\sin(x) - T_n(x; x_0)|$ over a domain while varying:

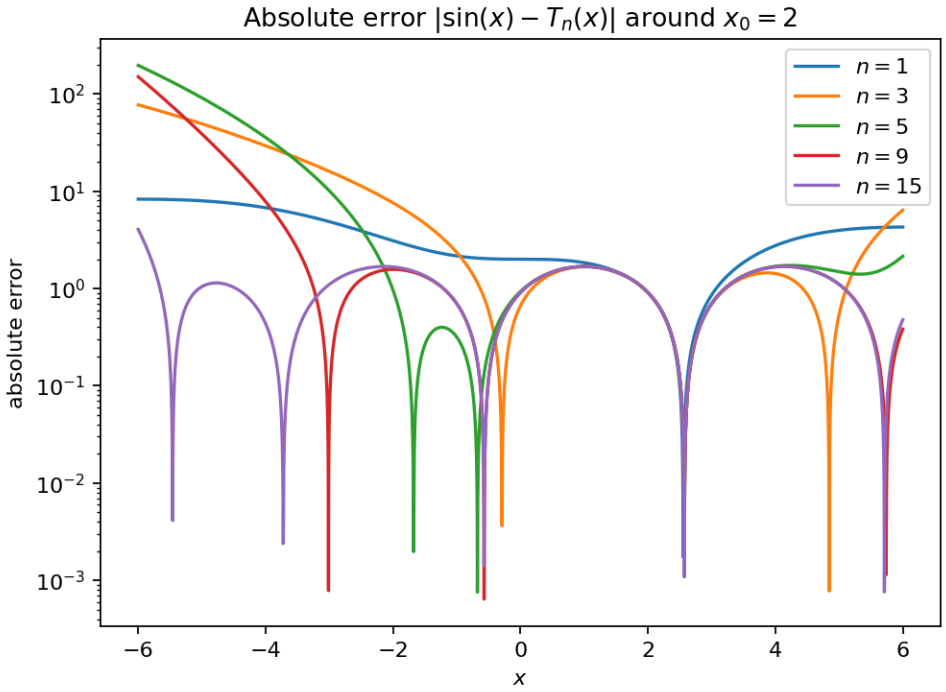
- the polynomial degree n,
- the expansion center x_0.

5.1.2 Background (quick refresher)

If you want a short mathematical recap first, read: *Taylor series refresher*.


Absolute error |sin(x) − T_n(x)| around x_0 = 0
absolute error
x
n = 1
n = 3
n = 5
n = 9
n = 15


Absolute error |sin(x) − T_n(x)| around x_0 = 1
absolute error
x
n = 1
n = 3
n = 5
n = 9
n = 15



5.1.3 Research question

How does the approximation error of Taylor polynomials for $\sin(x)$ depend on the polynomial degree n and the expansion center x_0, over a fixed domain of x?

Concretely: for a chosen grid of (n, x_0), what does the error landscape $E_{n,x_0}(x) = |\sin(x) - T_n(x; x_0)|$ look like, and where do numerical artifacts start to dominate the truncation error?

5.1.4 Why this qualifies as a mathematical experiment

This page is not just a worked example or a derivation — it is an *experiment* in the sense of **experimental mathematics**: a finite, reproducible procedure that produces **evidence** about how a mathematical object behaves.

For E001, the object is the family of Taylor polynomials $T_n(x; x_0)$ for $\sin(x)$, and the observable is the error function $E_{n,x_0}(x) = |\sin(x) - T_n(x; x_0)|$. The experiment qualifies because it:

- **Explores a parameter space:** it varies degree n and center x_0 and inspects how the entire error landscape changes.
- **Generates testable conjectures:** e.g. "there is a widening low-error region around x_0 as n increases" and "improvement is not uniform across a fixed interval".
- **Searches for failure modes / edge cases:** large $|x - x_0|$, high degrees, and wide domains can expose numerical artifacts that *look* like truncation error but are actually floating-point limitations.
- **Produces artifacts that can be checked independently:** plots and parameter snapshots make it easy to compare runs, reproduce the same conditions, and verify that an observed pattern is not accidental.

The goal is to turn "Taylor series should be good near x_0" into *structured evidence* about **where** and **how** the approximation is good (or bad), which then informs what one would try to prove or bound formally.

- How does the truncation error $|\sin(x) - T_n(x; x_0)|$ behave as we move away from the center x_0?
- How many terms are needed to achieve a specific precision (e.g., 10^{-6}) over a fixed interval?
- Does increasing the degree n always improve the result everywhere in the domain?

5.1.5 Experiment design

- **Target function:** $\sin(x)$
- **Evaluation:** $x \in [-2\pi, 2\pi]$ (default)
- **Parameters:**
  - `degrees`: $\{1, 3, 5, 7, 9\}$
  - `centers`: $\{0, \pi/2, \pi\}$
- **Outputs:**
  - Plot of $f(x)$ vs. $T_n(x)$
  - Semi-log plot of $|f(x) - T_n(x)|$

5.1.6 How to run

```
make run EXP=e001 ARGS="--seed 1"
```

Artifacts are written under `out/e001/` (figures, parameters, and a short `report.md`).

5.1.7 What to expect

Qualitatively (and this is what the plots should confirm):

- near x_0, the higher degree reduces the error quickly,
- away from x_0, the approximation can degrade even for higher degrees,
- changing x_0 shifts the "low-error region".

5.1.8 Results

After running the experiment, include (or regenerate) the figures in the documentation. The canonical output location is `out/e001/`. For publishing, copy one representative "hero" image into `docs/_static/experiments/` (see "Gallery images" below).

5.1.9 Notes / pitfalls

- Use **log-scale** for absolute error plots to see the full dynamic range.
- Be careful interpreting relative error near zeros of $\sin(x)$.
- Huge domains and high degrees can expose floating-point artifacts that are not truncation error.

5.1.10 Extensions

- **Alternative functions:** Repeat the experiment for $exp(x)$ or $1/(1-x)$.
- **Relative error:** Plot $|(f - T_n)/f|$ instead of absolute error.
- **Automatic degree selection:** Find the minimal n such that error $< \epsilon$ on a given interval.

5.1.11 Gallery images (recommended)

To keep the experiment gallery attractive and stable:

1. run the experiment locally,
2. pick one representative output figure,
3. copy it into the repo under:

   `docs/_static/experiments/e001_hero.png`

This allows the docs to show thumbnails without depending on generated `out/` artifacts.

5.1.12 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e001_taylor_error_landscapes
```

Seed: `1`

Parameters

- Domain: `[x_min, x_max] = [-6.0, 6.0]`
- num_points: `4000`
- degrees: `1, 3, 5, 9, 15`
- centers: `0, 1, 2`

5.1.13 What this run produces

- `figures/fig_01_sin_and_taylor_center_*.png` — overlay of sin(x) and Taylor polynomials for one center
- `figures/fig_02_error_landscape_center_*.png` — absolute error curves for each center, overlaid by degree

Notes

- Taylor approximations are *local*: accuracy is highest near the expansion center and generally degrades away from it.
- If you increase degrees or the domain size, floating-point roundoff may become visible.

params.json (snapshot)


```
{
  "centers": [
    0.0,
    1.0,
    2.0
  ],
  "degrees": [
    1,
    3,
    5,
    9,
    15
  ],
  "num_points": 4000,
  "x_max": 6.0,
  "x_min": -6.0
}
```


5.1.14 References

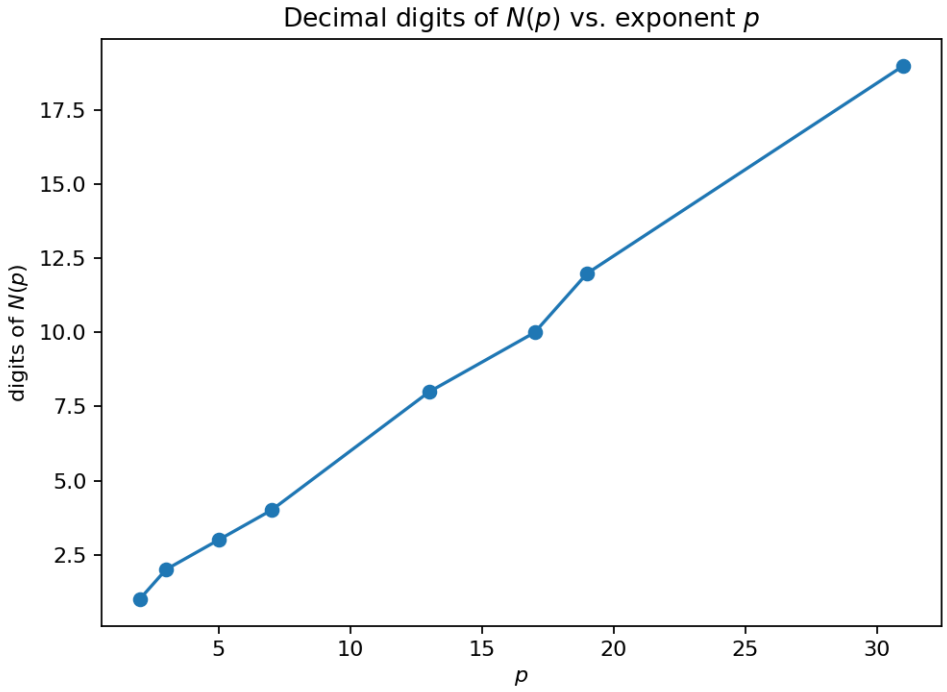
See ../references.

[Bailey and Borwein, 2005, Borwein, 2005, Borwein and Bailey, 2008]

5.1.15 Related experiments

- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E085: Dirichlet eta acceleration for (s)* (E085: Dirichlet eta acceleration for (s))
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)

5.2 E002: Even Perfect Numbers — Generator and Growth



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `numerics` See: *Valid Tags*.

5.2.1 Highlights

- Generate even perfect numbers from known Mersenne exponents p.
- Plot digits and bit-length growth vs. p.
- Test the approximation $\log_{10}(N(p)) \approx 2p \log_{10}(2)$.

5.2.2 Goal

Generate **even perfect numbers** from known Mersenne prime exponents p and visualize how fast the numbers grow. Measure growth via **digit count**, **bit length**, and simple logarithmic approximations.

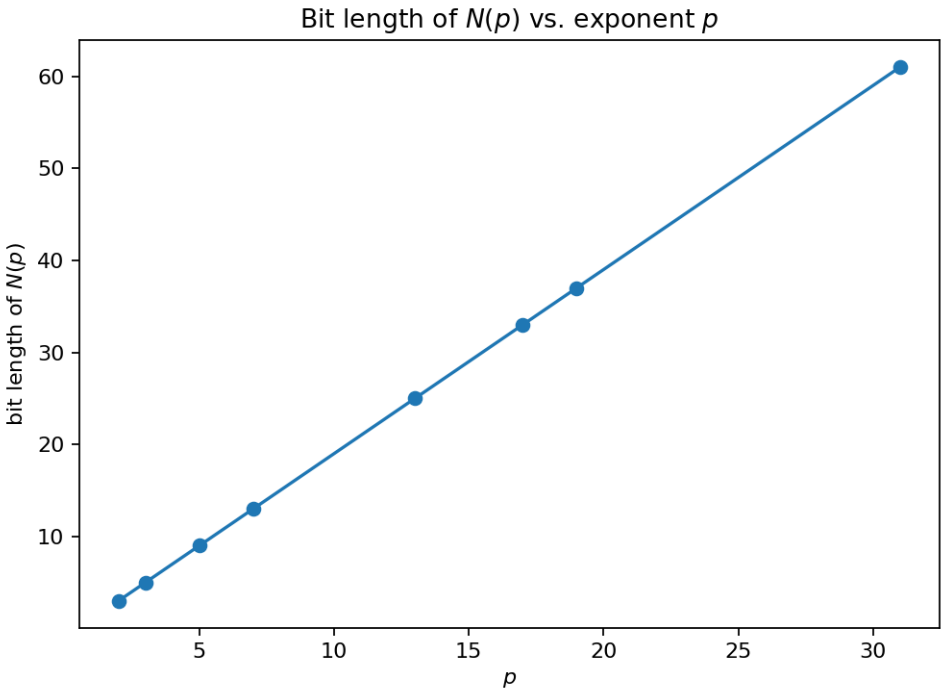
5.2.3 Research question

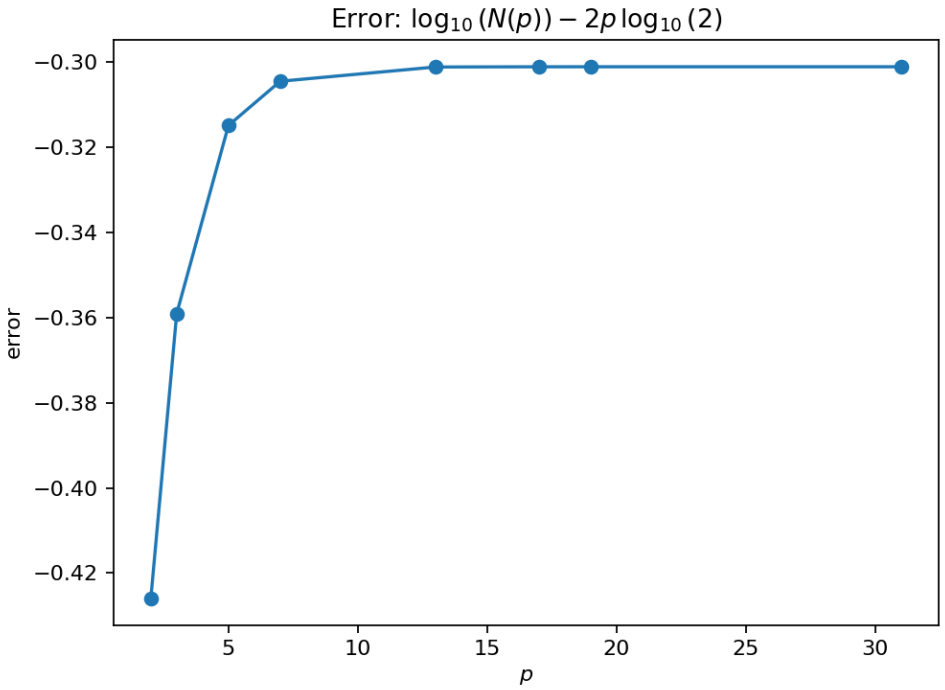
For

$$N(p) = 2^{p-1}(2^p - 1),$$

how do:

- digits(N),


Bit length of $N(p)$ vs. exponent p
bit length of $N(p)$
60
50
40
30
20
10
5
10
15
20
25
30
p


Error: $\log_{10}(N(p)) - 2p\log_{10}(2)$
error
−0.30
−0.32
−0.34
−0.36
−0.38
−0.40
−0.42
5
10
15
20
25
30
p

- bit_length(N),
- $\log_{10}(N)$

scale with the exponent p?

How accurate is the approximation

$$\log_{10}(N(p)) \approx 2p \log_{10}(2)?$$

5.2.4 Why this qualifies as a mathematical experiment

The Euclid–Euler theorem tells us exactly what even perfect numbers look like, but it does not directly convey how quickly the objects become astronomically large. This experiment uses computation and visualization to build quantitative intuition and test simple asymptotic approximations.

5.2.5 Experiment design

Inputs

- A curated list of known Mersenne prime exponents, e.g. $p \in \{2, 3, 5, 7, 13, 17, 19, 31, \dots\}$.

Observables

For each exponent p:

- $N(p)$ as a Python integer
- digits($N(p)$)
- bit_length($N(p)$)
- approximation error:

$$\Delta(p) = \log_{10}(N(p)) - 2p \log_{10}(2)$$

Plots

- p vs. digits
- p vs. bit length
- p vs. $\Delta(p)$

5.2.6 How to run

From the repo root:

```
make run EXP=e002
```

or:

```
uv run python -m mathxlab.experiments.e002
```

5.2.7 Notes / pitfalls

- Avoid converting huge integers to decimal strings repeatedly in tight loops; compute digits using logs where possible.
- Use `int.bit_length()` for stable bit length (fast and exact).
- Keep the exponent list modest for fast docs builds; this is a growth experiment, not a search for new Mersenne primes.

5.2.8 Extensions

- Compare growth to 2^{2p} and quantify the relative gap.
- Add a "human scale" axis: compare digits($N(p)$) to common benchmarks (atoms in the observable universe, etc.).
- Pull the exponent list from a small data file so the experiment can be updated without code changes.

5.2.9 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e002 ARGS="--seed 1"
```

Seed: `1`

5.2.10 Exponents

Mersenne prime exponents used in this run:

`2, 3, 5, 7, 13, 17, 19, 31`

5.2.11 Outputs

- `figures/fig_01_digits_vs_p.png`
- `figures/fig_02_bits_vs_p.png`
- `figures/fig_03_log10_error_vs_p.png`
- `params.json`

Notes

- Growth is extreme: both digits and bit length scale roughly linearly in `p`.
- The log-approximation error stays bounded and illustrates why `N(p)` behaves like `2^(2p)` up to a small correction.

params.json (snapshot)


```
{
  "exponents": [
    2,
    3,
    5,
    7,
    13,
    17,
    19,
    31
  ]
}
```

5.2.12 References

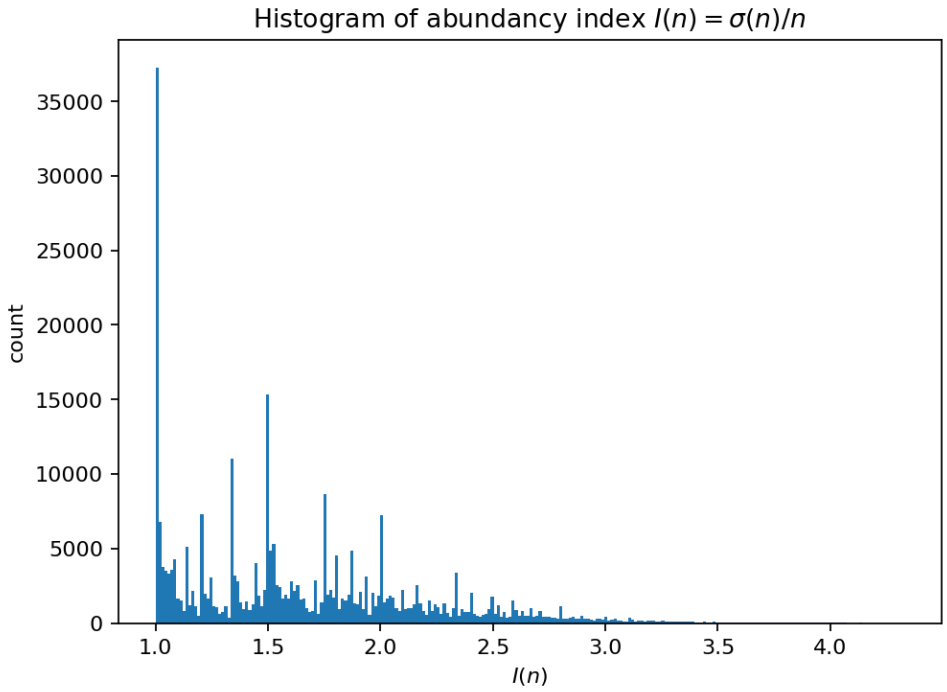
See ../references.

[Caldwell, n.d., Voight, 1998, OEIS Foundation Inc., 2025]

5.2.13 Related experiments

- *E005: Odd Perfect Numbers — Constraint Filter Pipeline* (Odd Perfect Numbers — Constraint Filter Pipeline)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E097: (n)/n landscape: deficient, perfect, abundant* (E097: (n)/n landscape: deficient, perfect, abundant)

5.3 E003: Abundancy Index Landscape



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `numerics` See: *Valid Tags*.

5.3.1 Highlights

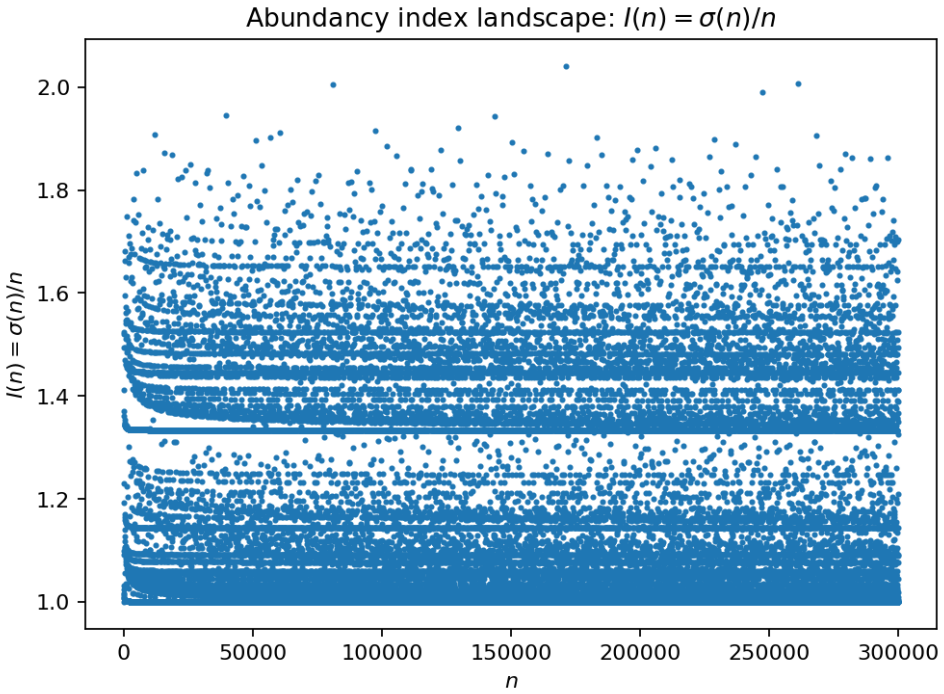
- Compute $\sigma(1..N)$ via a divisor-sum sieve.
- Visualize the distribution of $I(n) = \sigma(n)/n$.
- Highlight perfect numbers as the razor-thin level set $I(n) = 2$.

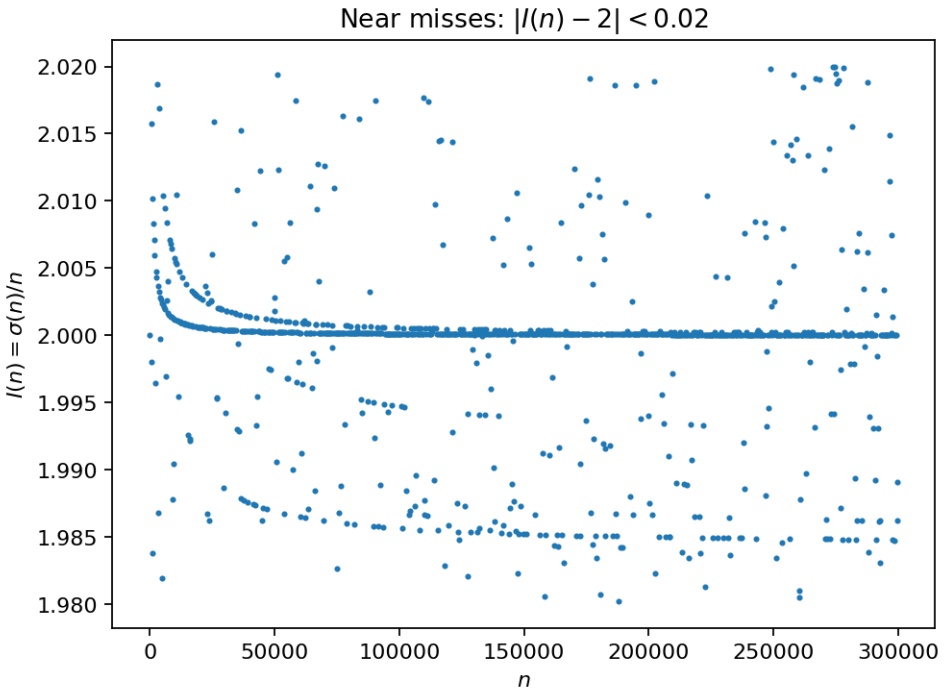
5.3.2 Goal

Visualize how **rare** perfect numbers are by plotting the distribution of the **abundancy index**

$$I(n) = \frac{\sigma(n)}{n}$$

for integers $n \leq N$. Perfect numbers satisfy $I(n) = 2$.


Abundancy index landscape: $I(n) = \sigma(n)/n$
$I(n) = \sigma(n)/n$
2.0
1.8
1.6
1.4
1.2
1.0
0
50000
100000
150000
200000
250000
300000
n


Near misses: $|I(n) - 2| < 0.02$
$I(n) = \sigma(n)/n$
2.020
2.015
2.010
2.005
2.000
1.995
1.990
1.985
1.980
0
50000
100000
150000
200000
250000
300000
n

5.3.3 Research question

For a given bound N:

- What does the empirical distribution of $I(n)$ look like?
- How frequently do we see values near 2?
- Where do the perfect numbers appear in the landscape?

5.3.4 Why this qualifies as a mathematical experiment

The divisor-sum function $\sigma(n)$ is highly structured but "spiky" and hard to intuit symbolically. Computational sweeps reveal qualitative structure (clusters, gaps, tails) and put perfect numbers into context.

5.3.5 Experiment design

Method: compute $\sigma(1)$, ..., $\sigma(N)$ via a divisor-sum sieve

Use the identity "each divisor contributes to its multiples":

- initialize an array `sigma[0..N]` with zeros
- for each `d = 1..N`:
  - add `d` to `sigma[k]` for all multiples `k = d, 2d, 3d, ...`

This runs in about $O(N \log N)$ time and avoids per-number factorization.

Observables

- $I(n) = \sigma(n)/n$
- distance to perfection: $|I(n) - 2|$
- classification:
  - deficient: $I(n) < 2$
  - perfect: $I(n) = 2$
  - abundant: $I(n) > 2$

Plots

- histogram of $I(n)$ (or of $I(n) - 1$)
- scatter plot of n vs. $I(n)$ (optionally with log-scale on n)
- highlight perfect numbers

5.3.6 How to run

```
make run EXP=e003
```

or:

```
uv run python -m mathxlab.experiments.e003
```

5.3.7 Notes / pitfalls

- $I(n)$ is rational. For classification, compare integers using $\sigma(n)$ and $2n$ rather than floats.
- For plots, floats are fine, but compute the "perfect" condition as `sigma[n] == 2*n`.
- Start with $N \leq 1\,000\,000$ to keep runtime and memory reasonable.

5.3.8 Extensions

- Repeat for different ranges and overlay histograms.
- Plot the top-k values of $I(n)$ and compare to known extremal families.
- Explore the "near misses" set (feeds directly into E006).

5.3.9 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e003
```

Parameters

- N: `300000`
- scatter stride: `10`
- histogram bins: `250`
- near band: `0.02`

5.3.10 Outputs

- `figures/fig_01_hist_abundancy.png`
- `figures/fig_02_scatter_abundancy.png`
- `figures/fig_03_near_2.png`
- `params.json`

5.3.11 Findings

- Perfect numbers found ($\leq$ N): `4`

Notes

- Compare perfection using integers: (n) == 2n.
- For plotting, floats are acceptable, but classification should not depend on float rounding.

params.json (snapshot)

```
{
  "bins": 250,
  "n_max": 300000,
  "near_band": 0.02,
  "stride_scatter": 10
}
```

5.3.12 References

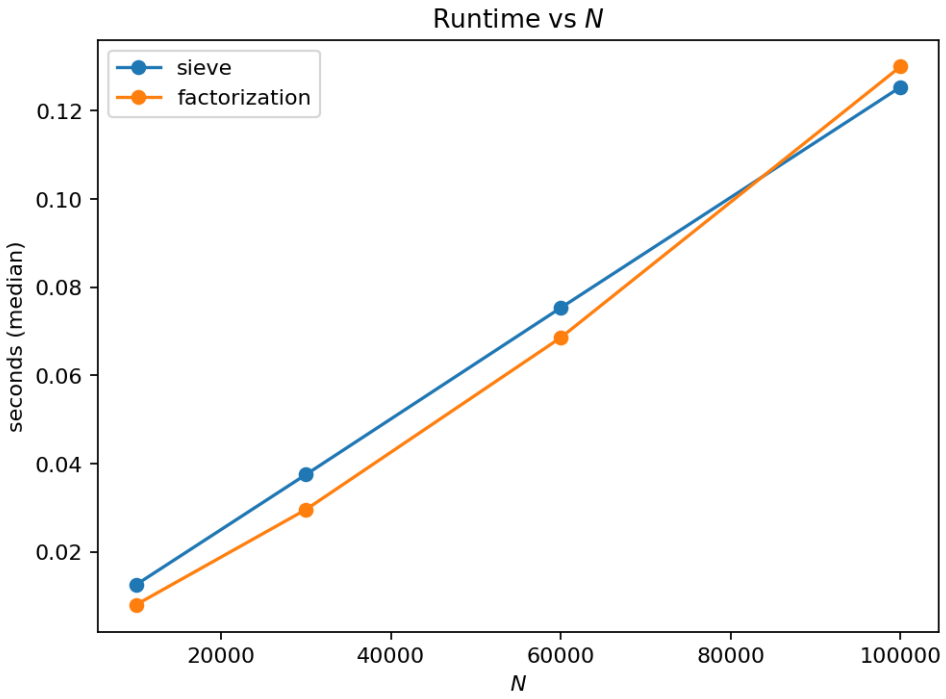
See ../references.

[Voight, 1998, Weisstein, 2003, OEIS Foundation Inc., 2025]

5.3.13 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E059: Abundancy index landscape* (Abundancy index landscape)
- *E094: (n) vs. Ω(n): Erdős–Kac normalization* (E094: (n) vs. Ω(n): Erdős–Kac normalization)
- *E097: (n)/n landscape: deficient, perfect, abundant* (E097: (n)/n landscape: deficient, perfect, abundant)

5.4 E004: Computing $\sigma(n)$ at Scale — Sieve vs. Factorization



Tags: `number-theory`, `quantitative-exploration`, `numerics`, `optimization` See: *Valid Tags*.

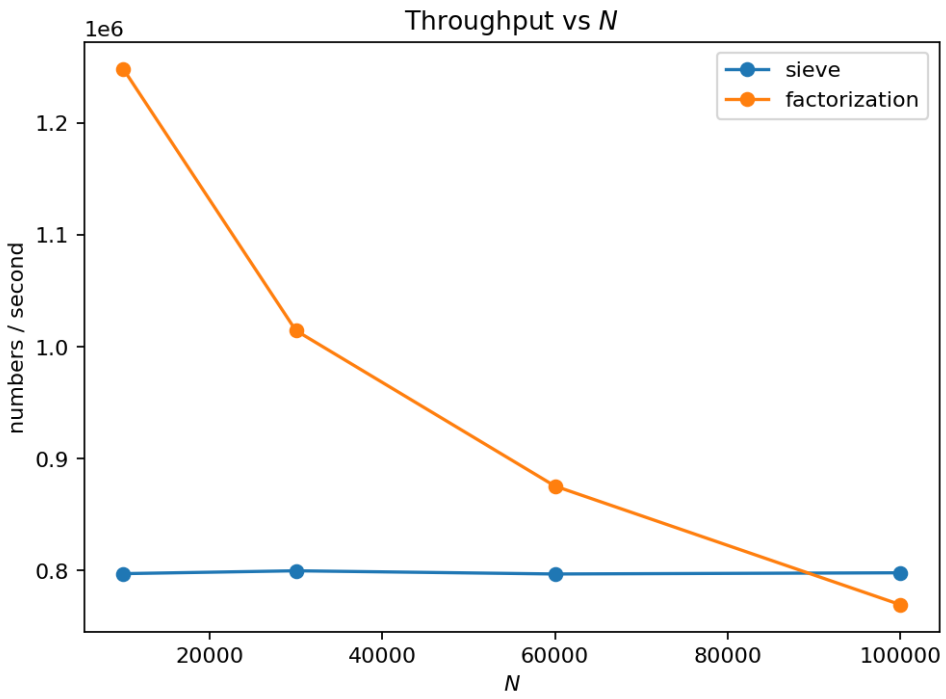
5.4.1 Highlights

- Benchmark bulk sieve vs. per-number factorization.
- Find runtime crossover points as N grows.
- Validate correctness on sampled inputs.

5.4.2 Goal

Benchmark two practical ways to compute the sum-of-divisors function:

1. **Sieve method:** compute all $\sigma(1..N)$ in one pass.
2. **Factorization method:** compute $\sigma(n)$ per-number from the prime factorization.



5.4.3 Research question

For a range of bounds N:

- which approach is faster?
- where is the crossover point?
- what are the memory tradeoffs?

5.4.4 Why this qualifies as a mathematical experiment

Both methods are mathematically equivalent, but performance depends on constants, caching, and implementation details. This is a quantitative exploration of algorithmic behavior grounded in number-theory structure.

5.4.5 Experiment design

Method A: divisor-sum sieve (bulk computation)

Compute `sigma[1..N]` by adding each divisor to its multiples (as in E003).

Method B: per-number factorization

Factor each n (e.g. using a precomputed prime list up to $\sqrt{N}$) and compute:

If

$$n = \prod_{i=1}^{k} p_i^{a_i},$$

then

$$\sigma(n) = \prod_{i=1}^{k} \frac{p_i^{a_i+1} - 1}{p_i - 1}.$$

Measurements

- wall-clock runtime vs. N (multiple trials, median)
- peak memory (rough estimate acceptable for v1)
- correctness cross-check: random sample where both methods agree

Outputs

- plot: runtime vs. N for both methods
- short table: N, runtime(A), runtime(B), speedup

5.4.6 How to run

```
make run EXP=e004
```

or:

```
uv run python -m mathxlab.experiments.e004
```

5.4.7 Notes / pitfalls

- Factorization becomes expensive quickly; keep the factorization method limited to moderate N.
- Use integer arithmetic for correctness checks (`sigma[n] == 2*n` etc.).
- Report both runtime and *effective throughput* (numbers processed per second).

5.4.8 Extensions

- Add a third method using smallest-prime-factor (SPF) sieve for fast factorization.
- Compare pure Python vs. `numpy` arrays for Method A.
- Turn the benchmark into a reusable utility for later experiments.

5.4.9 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e004
```

Parameters

- n_values: `10000, 30000, 60000, 100000`
- trials: `3`

5.4.10 Results (median runtime)

N	sieve [s]	factorization [s]	speedup (sieve/fact)
10000	0.0103	0.0056	1.83
30000	0.0303	0.0205	1.48
60000	0.0605	0.0476	1.27
100000	0.1020	0.0877	1.16

Notes

- Sieve computes all (1..N) at once; factorization recomputes structure per number.
- Factorization is capped to moderate N to keep runtime reasonable in pure Python.

params.json (snapshot)


```
{
  "max_n_factor": 100000,
  "n_values": [
    10000,
    30000,
    60000,
    100000
  ],
  "trials": 3
}
```


5.4.11 References

See ../references.

[Voight, 1998, Weisstein, 2003]

5.4.12 Related experiments

- *E098: Maximizers of (n)/n^ across* (E098: Maximizers of (n)/n^ across)
- *E017: Sieve memory blow-up vs. segmented sieve* (Sieve memory blow-up vs. segmented sieve)
- *E051: Semiprimes: balanced vs. unbalanced factorization timing* (Semiprimes: balanced vs. unbalanced factorization timing)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)

5.5 E005: Odd Perfect Numbers — Constraint Filter Pipeline

Tags: `number-theory`, `counterexample-search`, `visualization`, `search` See: *Valid Tags*.

5.5.1 Highlights

- Apply necessary constraints as staged filters on odd candidates.
- Plot survival curves (remaining candidates per constraint stage).
- Make the "open problem" constraints tangible at finite scales.

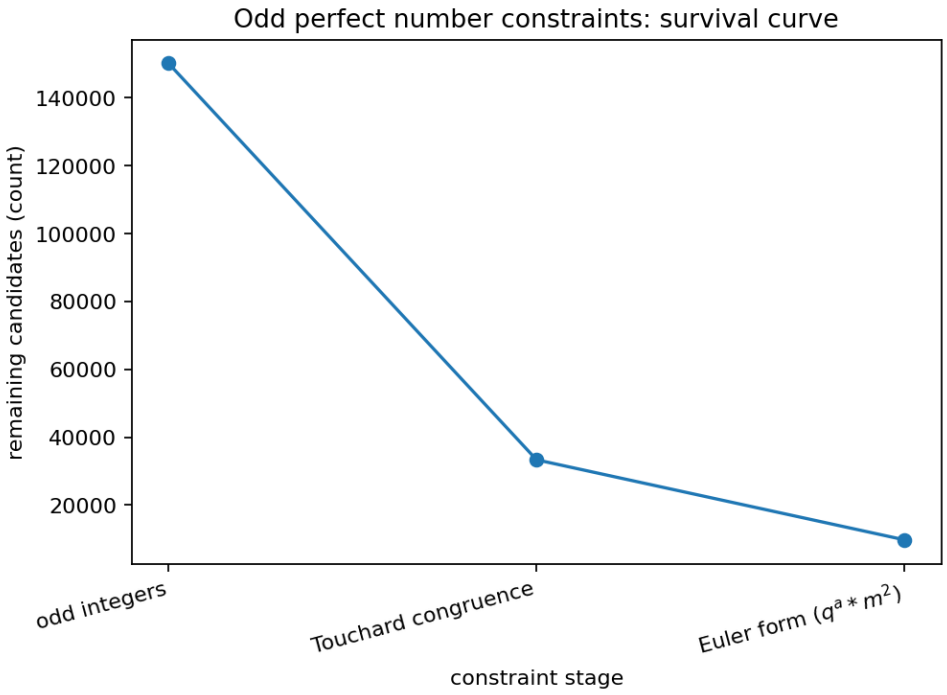
5.5.2 Goal

Demonstrate *why brute-force search for odd perfect numbers is unrealistic* by applying known **necessary conditions** as a step-by-step filter pipeline and visualizing how the candidate pool collapses.

5.5.3 Research question

Given odd integers up to a bound N:

- how many survive each constraint stage?
- which constraints are most "eliminating" in practice at small scales?



- what does a survival curve suggest about scalability?

5.5.4 Why this qualifies as a mathematical experiment

Odd perfect numbers are an open problem. Theorems provide constraints rather than a classification. Computation makes these constraints tangible by measuring their elimination power on finite candidate sets.

5.5.5 Experiment design

Candidate set

Start with odd integers $1 < n \leq N$.

Constraint stages (v1)

Use a conservative, well-known set of *necessary* conditions that can be checked mechanically:

1. **Euler form (shape constraint):** any odd perfect number must be of the form

$$n = q^{\alpha} m^2$$

   where q is prime, $\gcd(q, m) = 1$, and

$$q \equiv \alpha \equiv 1 \pmod 4.$$

2. **Congruence filter (Touchard-type):** odd perfect numbers satisfy strong congruence restrictions (implemented as one or two simple congruence checks supported by the references).
3. **Small-prime structure filters:** apply a few lightweight necessary conditions (e.g., reject candidates divisible by some specific small patterns if justified by the reference set).

For each stage, record "remaining candidates".

Output

- table: stage name, remaining count, elimination percentage
- plot: survival curve (stage index vs. remaining candidates)

5.5.6 How to run

```
make run EXP=e005
```

or:

```
uv run python -m mathxlab.experiments.e005
```

5.5.7 Notes / pitfalls

- Be explicit about what is *proved* vs. what is a heuristic. Only implement conditions that are truly necessary.
- At small N, some deep constraints won't show their full strength; the goal is the *pipeline idea*, not a record bound.
- Euler-form testing requires factorization; keep N small enough for trial division (v1).

5.5.8 Extensions

- Add stronger bounds/constraints from the modern literature and compare elimination power.
- Replace trial division with an SPF sieve to scale the Euler-form test.
- Report not just counts but also the distribution of remaining prime-factor patterns.

5.5.9 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e005
```

Parameters

- N: `300000`

5.5.10 Survival table

Stage	Remaining
odd integers	149999
Touchard congruence	33333
Euler form ($q^a * m^2$)	9774

Notes

- These are necessary conditions only; surviving candidates are *not* perfect by implication.
- Euler-form testing requires factorization; SPF makes it feasible for moderate N.

params.json (snapshot)

```
{
  "n_max": 300000,
  "stride_plot": 1
}
```

5.5.11 References

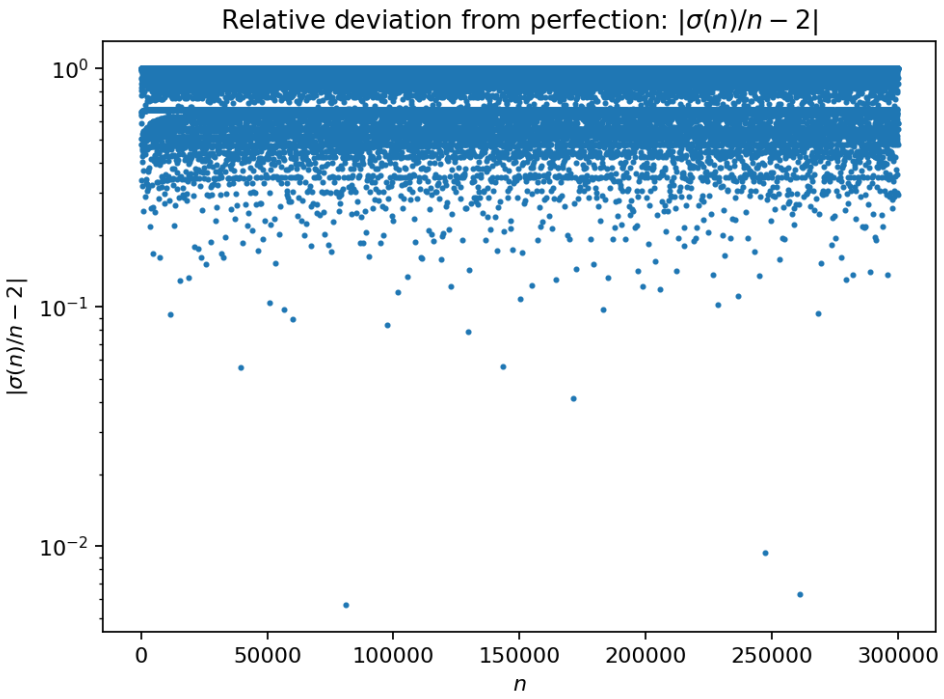
See ../references.

[Guy, 2004, Ochem and Rao, 2014, Stone, 2024, Voight, 1998]

5.5.12 Related experiments

- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)
- *E046: Prime-testing pipeline and tuning pitfalls* (Prime-testing pipeline and tuning pitfalls)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)

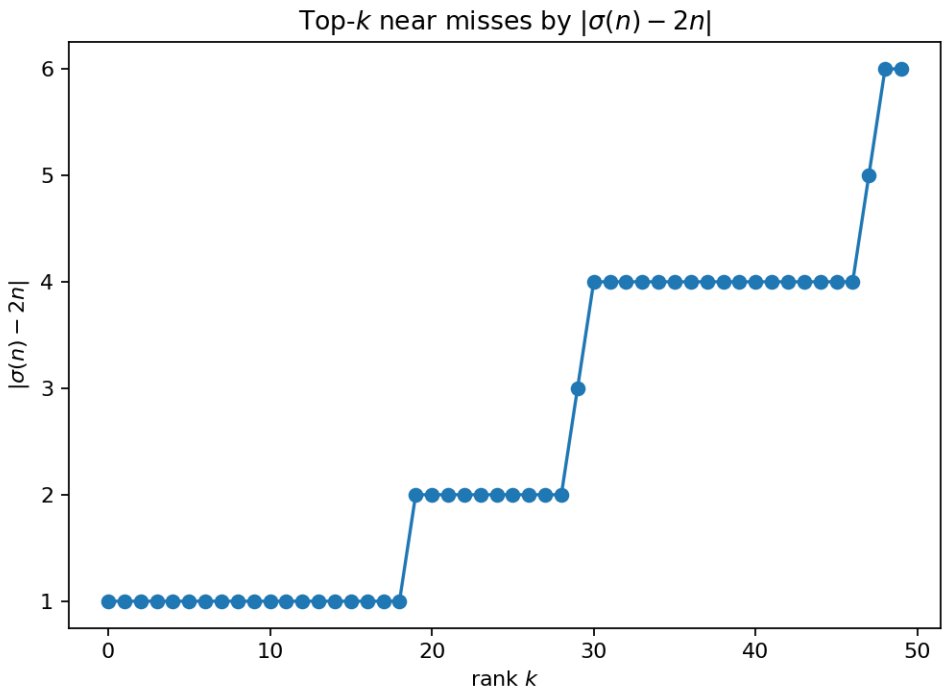
5.6 E006: Near Misses to Perfection



Tags: `number-theory`, `conjecture-generation`, `visualization`, `numerics` See: *Valid Tags*.

5.6.1 Highlights

- Search for n with $\sigma(n)$ unusually close to $2n$.
- Build leaderboards for absolute and relative deviation.
- Visualize how "near perfection" clusters by structure.



5.6.2 Goal

Find and visualize integers whose divisor sum is **unusually close** to the perfect condition $\sigma(n) = 2n$, without being perfect.

5.6.3 Research question

For integers $n \leq N$, which numbers minimize:

- absolute deviation:

$$D_1(n) = |\sigma(n) - 2n|$$

- relative deviation:

$$D_2(n) = \left|\frac{\sigma(n)}{n} - 2\right|?$$

Do "near misses" cluster in recognizable families (highly composite, abundant, etc.)?

5.6.4 Why this qualifies as a mathematical experiment

The perfect condition is a sharp equality. Studying the closest failures often reveals structure and suggests new questions (e.g., which multiplicative patterns drive $\sigma(n)$ toward $2n$).

5.6.5 Experiment design

Computation

- Compute $\sigma(1..N)$ via the divisor-sum sieve (as in E003).
- For each n, compute $D_1(n)$ and $D_2(n)$.
- Keep the top-k smallest deviations (excluding actual perfect numbers).

Outputs

- table: top-k near misses (with n, $\sigma(n)$, D_1, D_2)
- plot: n vs. $D_2(n)$ (log-scale on D_2 often helps)
- mark perfect numbers for reference

5.6.6 How to run

```
make run EXP=e006
```

or:

```
uv run python -m mathxlab.experiments.e006
```

5.6.7 Notes / pitfalls

- Use integer comparisons to identify perfect numbers (`sigma[n] == 2*n`).
- For D_2, floats are fine for plotting, but store exact rational values for ranking when possible (e.g., compare $|\sigma(n) - 2n|$ first, then normalize for reporting).
- Choose k small (e.g. 50 or 200) so the report stays readable.

5.6.8 Extensions

- Repeat for different N and compare stability of the "near miss" leaderboard.
- Add a second leaderboard restricted to odd n only.
- Compare near misses to known abundant/deficient classifications and prime factorizations.

5.6.9 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e006
```

Parameters

- N: `300000`
- top_k: `50`

5.6.10 Top near misses (excluding perfect numbers)

| n | (n) | | (n)-2n| | | (n)/n - 2| | |—:|—:|—:|—:| | 131072 | 262143 | 1 | 7.62939e-06 | | 8192 | 16383 | 1 | 0.00012207 | | 262144 | 524287 | 1 | 3.8147e-06 | | 64 | 127 | 1 | 0.015625 | | 32768 | 65535 | 1 | 3.05176e-05 | | 4096 | 8191 | 1 | 0.000244141 | | 65536 | 131071 | 1 | 1.52588e-05 | | 32 | 63 | 1 | 0.03125 | | 2048 | 4095 | 1 | 0.000488281 | | 1024 | 2047 | 1 | 0.000976562 | | 16384 | 32767 | 1 | 6.10352e-05 | | 128 | 255 | 1 | 0.0078125 | | 2 | 3 | 1 | 0.5 | | 4 | 7 | 1 | 0.25 | | 1 | 1 | 1 | 1 | | 512 | 1023 | 1 | 0.00195312 | | 8 | 15 | 1 | 0.125 | | 16 | 31 | 1 | 0.0625 | | 256 | 511 | 1 | 0.00390625 | | 136 | 270 | 2 | 0.0147059 | | 32896 | 65790 | 2 | 6.07977e-05 | | 1952 | 3906 | 2 | 0.00102459 | | 3 | 4 | 2 | 0.666667 | | 130304 | 260610 | 2 | 1.53487e-05 | | 650 | 1302 | 2 | 0.00307692 | | 10 | 18 | 2 | 0.2 | | 20 | 42 | 2 | 0.1 | | 464 | 930 | 2 | 0.00431034 | | 104 | 210 | 2 | 0.0192308 | | 18 | 39 | 3 | 0.166667 | | 2144 | 4284 | 4 | 0.00186567 | | 18632 | 37260 | 4 | 0.000214684 | | 70 | 144 | 4 | 0.0571429 | | 1888 | 3780 | 4 | 0.00211864 | | 110 | 216 | 4 | 0.0363636 | | 4030 | 8064 | 4 | 0.000992556 | | 884 | 1764 | 4 | 0.00452489 | | 14 | 24 | 4 | 0.285714 | | 12 | 28 | 4 | 0.333333 | | 5 | 6 | 4 | 0.8 | | 44 | 84 | 4 | 0.0909091 | | 8384 | 16764 | 4 | 0.000477099 | | 32128 | 64260 | 4 | 0.000124502 | | 152 | 300 | 4 | 0.0263158 | | 5830 | 11664 | 4 | 0.000686106 | | 116624 | 233244 | 4 | 3.42983e-05 | | 88 | 180 | 4 | 0.0454545 | | 9 | 13 | 5 | 0.555556 | | 315 | 624 | 6 | 0.0190476 | | 7 | 8 | 6 | 0.857143 |

Notes

- Ranking is primarily by absolute deviation | (n)-2n|, with relative deviation reported for context.

params.json (snapshot)


```
{
  "n_max": 300000,
  "stride_scatter": 10,
  "top_k": 50
}
```


5.6.11 References

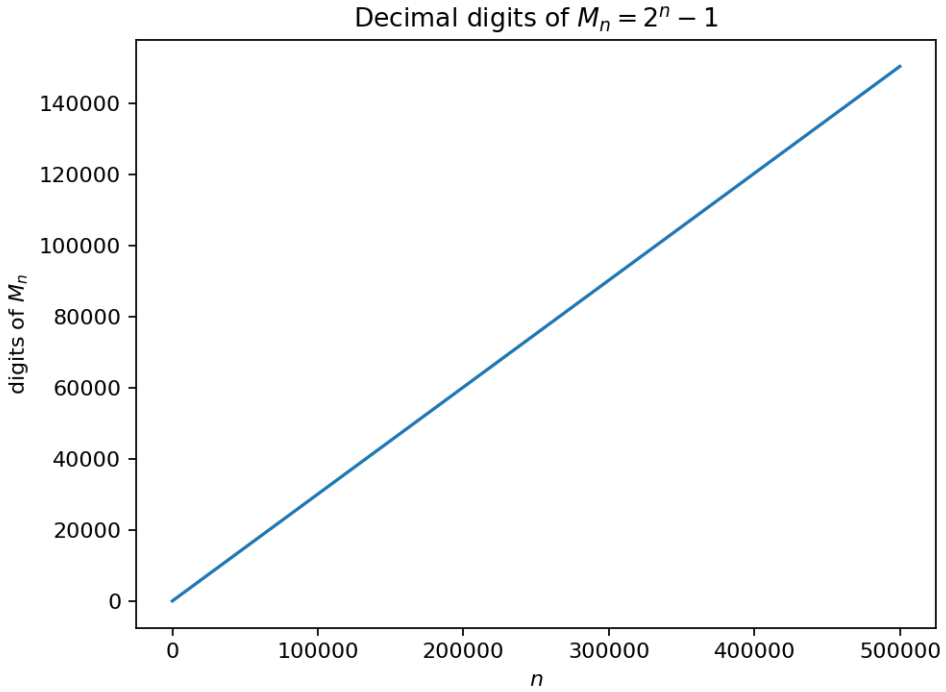
See ../references.

[Voight, 1998, Weisstein, 2003, OEIS Foundation Inc., 2025]

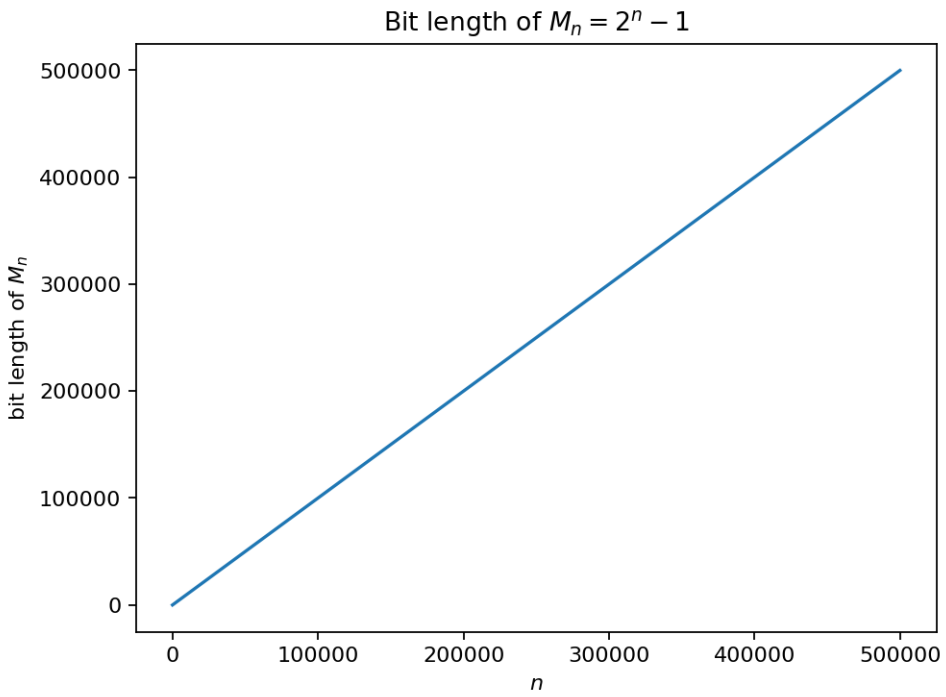
5.6.12 Related experiments

- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E029: Twin primes: observed vs. heuristic* (Twin primes: observed vs. heuristic)
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)

5.7 E007: Mersenne growth (bits and digits)



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.



5.7.1 Highlights

- Visualize how quickly $M_n = 2^n - 1$ grows in **bits** and **decimal digits**.
- Compute size metrics *without* constructing huge integers.
- Provide simple rules-of-thumb for choosing feasible bounds in later experiments.

5.7.2 Goal

Quantify and visualize the growth of Mersenne numbers $M_n = 2^n - 1$ as n increases.

This experiment focuses on **size** (bit-length and decimal digits), because it determines what later algorithms (Lucas–Lehmer, sieving) can realistically handle on a laptop.

5.7.3 Background (quick refresher)

- *Mersenne numbers and primes refresher*

5.7.4 Research question

How fast does the size of M_n grow with n, and what simple formulas approximate:

- the number of bits of M_n
- the number of decimal digits of M_n

5.7.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** evaluate size formulas on a range $n = 1..N$.
- **Observable(s):** bit-length and digit count as functions of n.
- **Parameter space:** vary N (and optional sampling density).
- **Outcome:** visual evidence (curves) and tables usable as planning guidance.
- **Reproducibility:** parameters written to `params.json`; figures written to `figures/`.

5.7.6 Experiment design

Computation

Key facts:

- $\text{bits}(2^n - 1) = n$ for $n \geq 1$.
- $\text{digits}(2^n - 1) = \lfloor n \log_{10}(2) \rfloor + 1$ for $n \geq 1$.

Compute both for a grid of n values.

Outputs

- plot: n vs. bits
- plot: n vs. digits
- table: selected n with digits (e.g., 10, 100, 1k, 10k, …)

5.7.7 How to run

```
make run EXP=e007
```

or:

```
uv run python -m mathxlab.experiments.e007
```

5.7.8 Notes / pitfalls

- Avoid converting huge M_n to decimal just to count digits. The logarithm formula is exact for the digit count.
- For very large n, use stable floating-point evaluation for $n \log_{10}(2)$ (double precision is fine up to very large n).
- This experiment is intentionally lightweight; it should run quickly.

5.7.9 Extensions

- Overlay “feasibility bands” (e.g., digits thresholds) for what your other experiments can handle.
- Compare M_n to other fast-growing families (factorials, primorials) on a log scale.

5.7.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e007
```

Parameters

- n_max: `500000`
- stride: `50`

5.7.11 Quick reference table

n	digits(M_n)	bits(M_n)
1	1	1
2	1	2
3	1	3
5	2	5
10	4	10
100	31	100
1000	302	1000
10000	3011	10000

Notes

- For $n \geq 1$, the bit-length of M_n is exactly n.
- digits(M_n) can be computed via $\text{floor}(n \cdot \log10(2)) + 1$ without building M_n.

params.json (snapshot)

```
{
  "n_max": 500000,
  "stride": 50
}
```

5.7.12 References

See ../references.

[contributors, 2025]

5.7.13 Related experiments

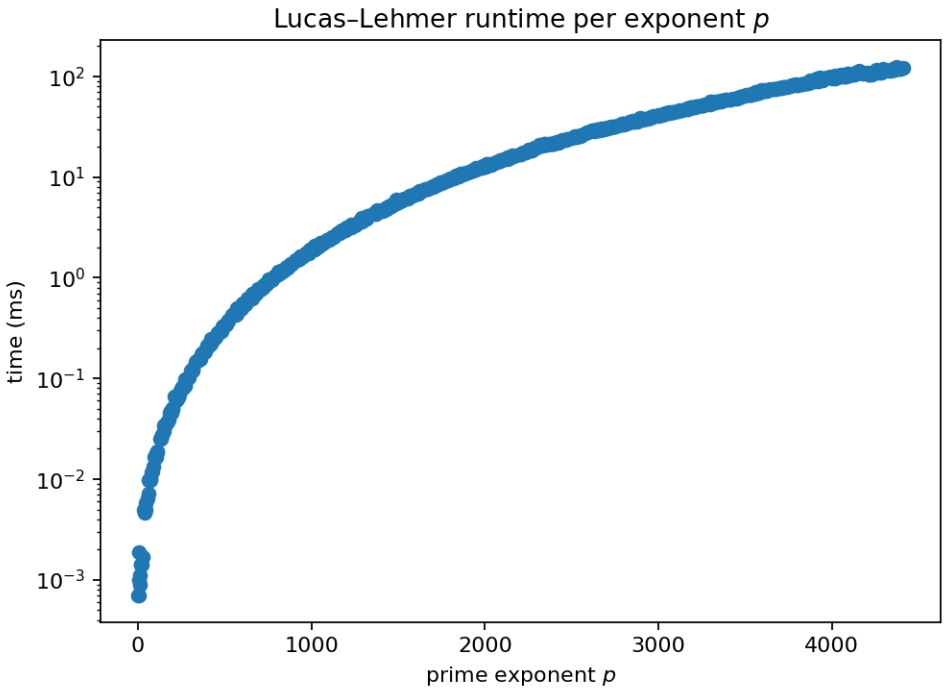
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)
- *E011: Heuristic rarity of Mersenne primes* (Heuristic rarity of Mersenne primes)
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)

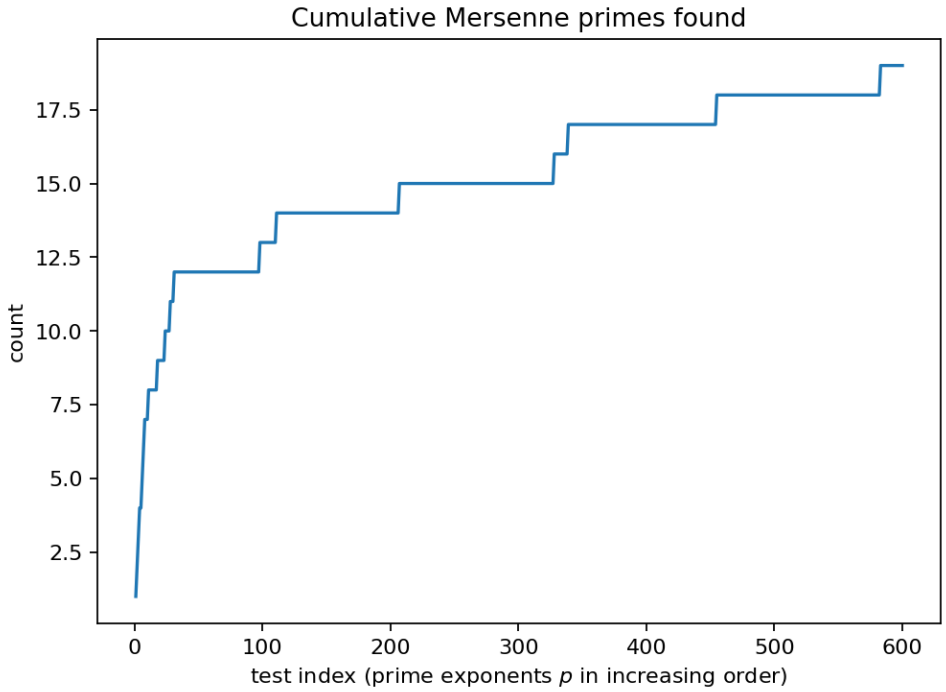
5.8 E008: Lucas–Lehmer scan (prime exponents)

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.8.1 Highlights

- Run the Lucas–Lehmer test for many prime exponents p.
- Record timing and residue behavior for composite vs. prime outcomes.
- Produce a reproducible "scan report" (which p were tested, which passed).


Lucas–Lehmer runtime per exponent p
time (ms)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
1000
2000
3000
4000
prime exponent p


Cumulative Mersenne primes found
count
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
100
200
300
400
500
600
test index (prime exponents p in increasing order)

5.8.2 Goal

Empirically explore Mersenne primality testing by scanning prime exponents p and applying the Lucas–Lehmer test to $M_p = 2^p - 1$.

5.8.3 Background (quick refresher)

- *Mersenne numbers and primes refresher*

5.8.4 Research question

For prime exponents $p \leq P$, how does:

- runtime of the Lucas–Lehmer test scale with p?
- the distribution of final residues (mod M_p) differ between prime and composite outcomes?

5.8.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** enumerate prime exponents up to a bound and test them.
- **Observable(s):** pass/fail, final residue, and runtime.
- **Parameter space:** vary P and optional "max tests" cap.
- **Outcome:** a dataset (tested p values) and plots/tables that suggest scaling behavior.
- **Failure modes:** naive implementations can be slow; parameters bound the search to keep runs feasible.

5.8.6 Experiment design

Computation

- Enumerate prime exponents p up to a bound P.
- For each p, compute $M_p = 2^p - 1$ and run Lucas–Lehmer:
  - $s_0 = 4$
  - $s_{k+1} = s_k^2 - 2 \pmod{M_p}$
  - M_p is prime iff $s_{p-2} \equiv 0 \pmod{M_p}$.
- Record (at minimum): p, pass/fail, final residue, and wall time.

Outputs

- table: scanned exponents with result and time
- plot: p vs. runtime (often log-scale is helpful)
- plot: residue magnitude / patterns (optional)

5.8.7 How to run

```
make run EXP=e008
```

or:

```
uv run python -m mathxlab.experiments.e008
```

5.8.8 Notes / pitfalls

- Only run Lucas–Lehmer for **prime** p; if p is composite then M_p is composite.
- The test is defined for odd prime p; treat $p = 2$ separately ($M_2 = 3$ is prime).
- Keep bounds modest at first; the cost grows quickly with p.

5.8.9 Extensions

- Add a "pre-sieve" that checks for small factors before running Lucas–Lehmer.
- Compare your timings to published implementations (as a sanity check, not a competition).

5.8.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e008
```

Parameters

- p_max: `10000`
- max_tests: `600`

5.8.11 Results

- tested exponents: `600`
- Mersenne primes found: `19`

Prime exponents p where M_p is prime

2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253

Notes

- LLT is a deterministic primality test specialized to M_p = 2^p - 1.
- Only prime exponents p need to be tested: if 2^n-1 is prime, then n must be prime.

params.json (snapshot)


```
{
  "max_tests": 600,
  "p_max": 10000
}
```


5.8.12 References

See ../references.

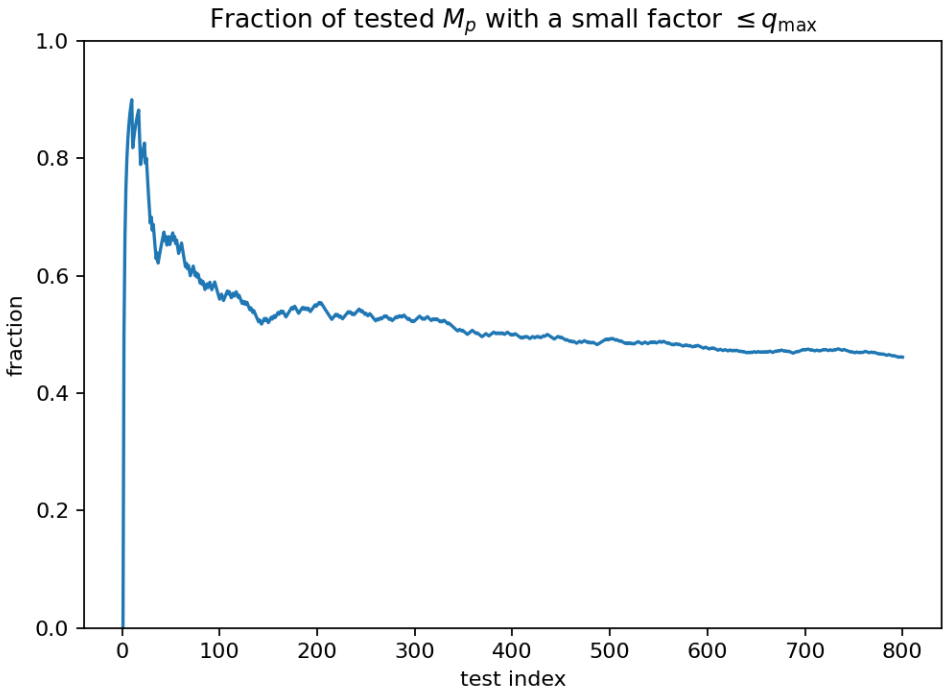
[Caldwell, 2021, contributors, 2025, Mersenne Research, 2024]

5.8.13 Related experiments

- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E042: Repunit primes (small k scan)* (Repunit primes (small k scan))

- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E049: Wieferich primes (base 2): scan and quotient visualization* (Wieferich primes (base 2): scan and quotient visualization)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.9 E009: Small-factor scan for Mersenne numbers



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.9.1 Highlights

- Quickly eliminate many M_p candidates by finding small prime factors.
- Use the structure of possible factors of M_p to reduce wasted work.
- Show how often "trivial" factors appear in a range of exponents.

5.9.2 Goal

Explore how often Mersenne numbers $M_p = 2^p - 1$ (with prime p) have **small factors**, and how a lightweight sieve can filter candidates before running Lucas–Lehmer.

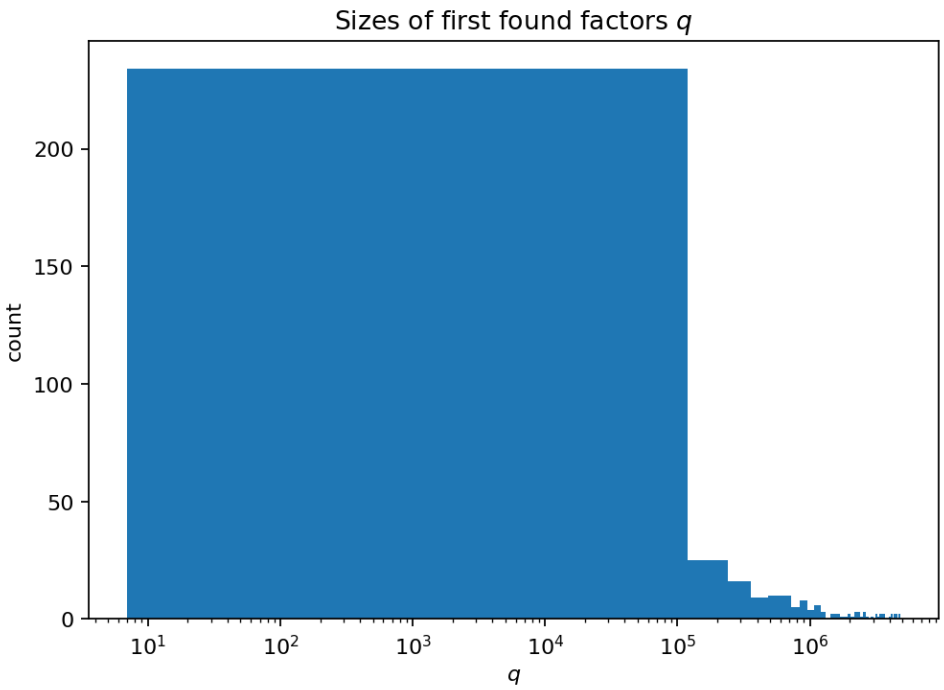
5.9.3 Background (quick refresher)

- *Mersenne numbers and primes refresher*

5.9.4 Research question

For prime exponents $p \leq P$ and a factor bound $q \leq Q$:

- how many M_p have a factor below Q?
- how much compute can be saved by sieving before Lucas–Lehmer?



5.9.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** scan a finite range of p and candidate factors.
- **Observable(s):** first small factor found (if any), counts by p and by factor size.
- **Parameter space:** vary P and Q.
- **Outcome:** tables/plots that show factor frequency and sieve effectiveness.
- **Failure modes:** incomplete sieve bounds can mislead; clearly report Q as a limitation.

5.9.6 Experiment design

Computation

For fixed prime exponent p, any prime factor q of M_p satisfies:

- $q \equiv 1 \pmod{2p}$ (a common necessary condition used to generate candidates)

A practical sieve approach:

- Generate candidate primes q of the form $q = 2pk + 1$ up to Q.
- For each candidate prime q, check:
  - $2^p \equiv 1 \pmod{q}$

  If true, then q divides M_p.

Outputs

- table: per p, smallest factor found (or "none up to Q")
- plot: p vs. smallest factor (when present)
- summary: fraction of p eliminated by the sieve

5.9.7 How to run

```
make run EXP=e009
```

or:

```
uv run python -m mathxlab.experiments.e009
```

5.9.8 Notes / pitfalls

- This is a **bounded** sieve: "no factor found" only means "none found up to Q".
- Prime testing for q should be fast; for small Q, a simple deterministic method is fine.
- Report both P and Q prominently in the report.

5.9.9 Extensions

- Combine with E008: run Lucas–Lehmer only on exponents not eliminated by the sieve.
- Track how much wall time is saved by the combined pipeline.

5.9.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e009
```

Parameters

- p_max: `10000`
- max_tests: `800`
- q_max: `5000000`
- require_q_prime: `True`

5.9.11 Results

- tested exponents: `800`
- with small factor found: `369`

5.9.12 Sample (first 20 with a found factor)

p	factor q
3	7
5	31
7	127
11	23
13	8191
17	131071
19	524287
23	47
29	233
37	223
41	13367
43	431
47	2351
53	6361
59	179951
71	228479
73	439
79	2687
83	167
97	11447

Notes

- The congruence filter q ≡ 1 (mod 2p) is necessary but not sufficient.
- A found q certifies M_p is composite; it does not factor M_p completely.

params.json (snapshot)

```
{
  "max_tests": 800,
  "p_max": 10000,
  "q_max": 5000000,
  "require_q_prime": true
}
```

5.9.13 References

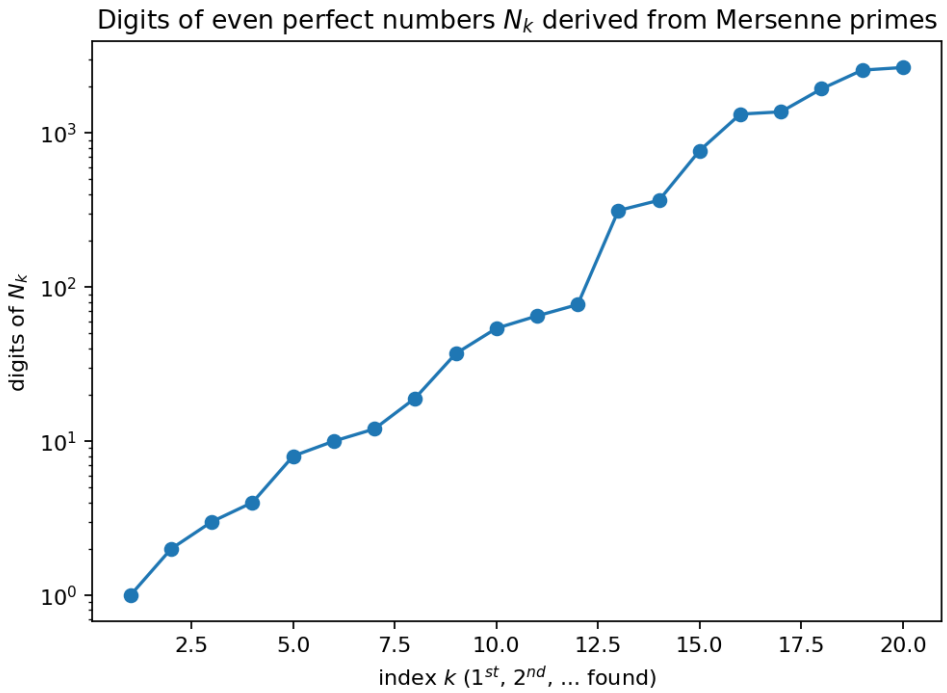
See ../references.

[Caldwell, 2021, contributors, 2025, Inc., 2025]

5.9.14 Related experiments

- *E042: Repunit primes (small k scan)* (Repunit primes (small k scan))
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)
- *E011: Heuristic rarity of Mersenne primes* (Heuristic rarity of Mersenne primes)

5.10 E010: Even perfect numbers from Mersenne primes



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.10.1 Highlights

- Generate even perfect numbers via the Euclid–Euler theorem.
- Connect Mersenne prime exponents p to perfect numbers $N = 2^{p-1}(2^p - 1)$.
- Visualize growth and verify the defining property $\sigma(N) = 2N$ for sampled cases.

5.10.2 Goal

Make the Mersenne $\leftrightarrow$ perfect-number connection computationally explicit:

If $M_p = 2^p - 1$ is prime, then:

$$N_p = 2^{p-1}(2^p - 1)$$

is an **even perfect number**.

5.10.3 Background (quick refresher)

- *Mersenne numbers and primes refresher*
- *Perfect numbers refresher*

5.10.4 Research question

Across the Mersenne prime exponents discovered within your scan bounds:

- how fast do the corresponding even perfect numbers grow?
- can we verify perfectness ($\sigma(N) = 2N$) efficiently for these cases?

5.10.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** find a set of Mersenne primes within a finite bound and generate perfect numbers.
- **Observable(s):** size metrics (digits), and validation checks of $\sigma(N) = 2N$.
- **Parameter space:** vary the exponent bound and validation depth.
- **Outcome:** concrete examples + growth plots that support intuition.
- **Reproducibility:** exponents tested and successes recorded in artifacts.

5.10.6 Experiment design

Computation

- Obtain a list of Mersenne prime exponents p from a scan (or a fixed list for small p).
- For each p, compute $N_p = 2^{p-1}(2^p - 1)$.
- Verify perfectness for these cases:

$$\sigma(N_p) = 2N_p.$$

For modest p, exact computation is feasible; for larger p, report size metrics and skip expensive checks.

Outputs

- table: p, M_p size, N_p size, and validation status
- plot: p vs. digits of N_p
- optional: prime-factor structure display for small cases

5.10.7 How to run

```
make run EXP=e010
```

or:

```
uv run python -m mathxlab.experiments.e010
```

5.10.8 Notes / pitfalls

- Don't attempt divisor-sum sieves for huge N_p; validation must be bounded and explicit.
- Clearly separate "constructed from theorem" (conditional on M_p being prime) from "validated by computation".

5.10.9 Extensions

- Cross-link to E002 outputs for perfect numbers and compare growth on the same axes.
- Explore which parts of the perfectness check can be done symbolically using known factorization.

5.10.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e010
```

Parameters

- p_max: 20000
- max_tests: 800

5.10.11 Mersenne prime exponents found

2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423

5.10.12 Derived even perfect numbers

p	digits(N)
2	1
3	2
5	3
7	4
13	8
17	10
19	12
31	19
61	37
89	54
107	65
127	77
521	314
607	366
1279	770
2203	1327
2281	1373
3217	1937
4253	2561
4423	2663

Notes

- N = 2^(p-1) · (2^p-1) is perfect iff M_p is prime (Euclid–Euler).
- For large p, we avoid constructing N explicitly and report its size (digits).

params.json (snapshot)


```
{
  "max_tests": 800,
  "p_max": 20000
}
```


5.10.13 References

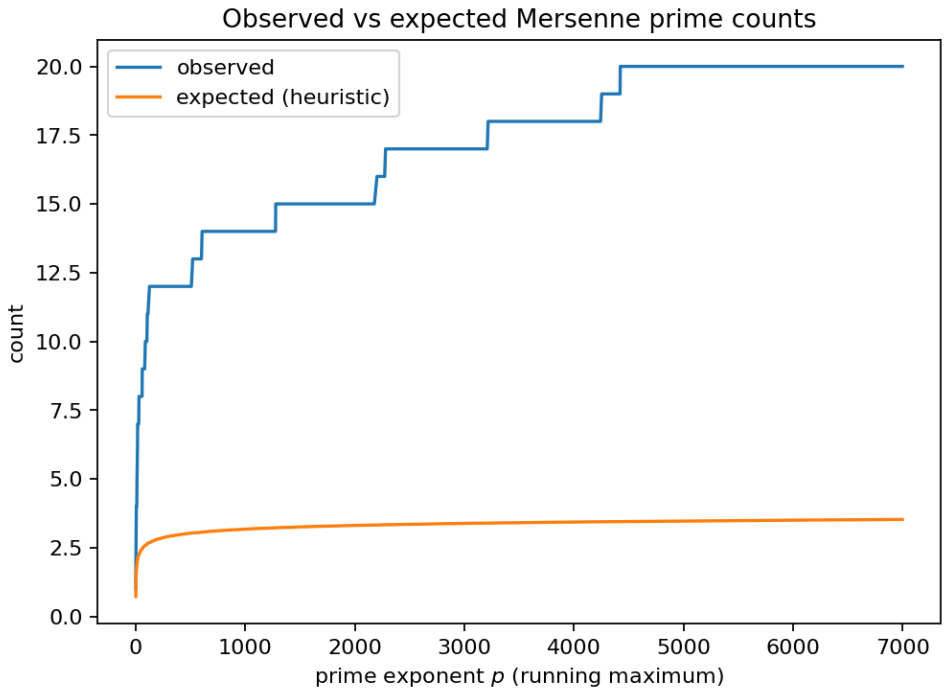
See ../references.

[Caldwell, n.d., contributors, 2025, Inc., 2025]

5.10.14 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E011: Heuristic rarity of Mersenne primes* (Heuristic rarity of Mersenne primes)
- *E097: (n)/n landscape: deficient, perfect, abundant* (E097: (n)/n landscape: deficient, perfect, abundant)

5.11 E011: Heuristic rarity of Mersenne primes



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.11.1 Highlights

- Compare observed Mersenne-prime counts to a simple heuristic expectation curve.
- Visualize cumulative "found vs. expected" as exponent bounds increase.
- Keep the discussion explicitly empirical (evidence, not proof).

5.11.2 Goal

Mersenne primes are extremely rare. This experiment compares:

- the observed count of Mersenne primes among prime exponents $p \leq P$
- a lightweight heuristic curve that grows slowly with P

The aim is intuition-building and trend visualization, not a rigorous model.

5.11.3 Background (quick refresher)

- *Mersenne numbers and primes refresher*

5.11.4 Research question

For increasing prime exponent bounds P:

- how does the cumulative count of "passes" from your scan grow?
- how does it compare to a simple expectation curve of the form:

$$E(P) \approx \sum_{p \le P,\ p \text{ prime}} \frac{1}{p \ln 2}$$

5.11.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a finite scan and compute a finite expected sum.
- **Observable(s):** cumulative count of passes vs. cumulative expected value.
- **Parameter space:** vary P (and optionally cap runtime per test).
- **Outcome:** plots that show agreement/divergence and suggest questions for deeper study.
- **Failure modes:** small ranges are noisy; the experiment should clearly label bounds.

5.11.6 Experiment design

Computation

- From E008 (or a fixed list), get the set of exponents tested and which passed.
- For each bound P, compute:
  - Found(P) = number of passing exponents $p \le P$
  - $E(P)$ = cumulative heuristic sum
- Plot both curves against P.

Outputs

- plot: P vs. Found(P) and E(P)
- table: selected checkpoints (P, Found, E)

5.11.7 How to run

```
make run EXP=e011
```

or:

```
uv run python -m mathxlab.experiments.e011
```

5.11.8 Notes / pitfalls

- The heuristic curve is a **comparison tool**, not a theorem.
- Results depend heavily on which exponents were actually tested; record the tested set in the report.
- A single large Mersenne prime can dominate impressions—keep axes and annotations clear.

5.11.9 Extensions

- Compare multiple heuristics (still empirical) and see which best matches the observed range.
- Extend the scan bound and see whether the deviation grows or stabilizes.

5.11.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e011
```

Parameters

- p_max: `20000`
- max_tests: `900`

5.11.11 Summary

- largest tested p: `6997`
- observed count: `20`
- expected (heuristic): `3.52597`

5.11.12 Found Mersenne prime exponents

2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423

Notes

- The heuristic probability is $\sim 1/(p \cdot \ln 2)$ for prime p.
- This is not a theorem; it is a back-of-the-envelope model for rarity.

params.json (snapshot)

```
{
  "max_tests": 900,
  "p_max": 20000
}
```

5.11.13 References

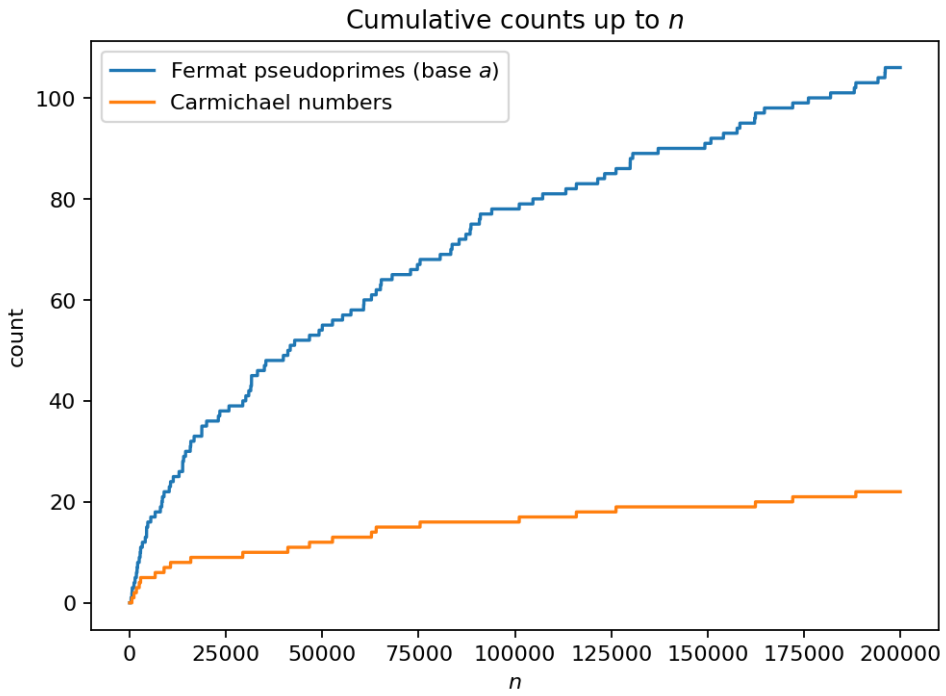
See ../references.

[Caldwell, 2021, Mersenne Research, 2025]

5.11.14 Related experiments

- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)
- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E029: Twin primes: observed vs. heuristic* (Twin primes: observed vs. heuristic)

5.12 E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)



Tags: `number-theory`, `counterexample-search`, `visualization` See: *Valid Tags*.

5.12.1 Highlights

- Counterexamples to naive Fermat primality testing: base-a pseudoprimes and Carmichael numbers.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.12.2 Goal

Counterexamples to naive Fermat primality testing: base-a pseudoprimes and Carmichael numbers.

5.12.3 Background (quick refresher)

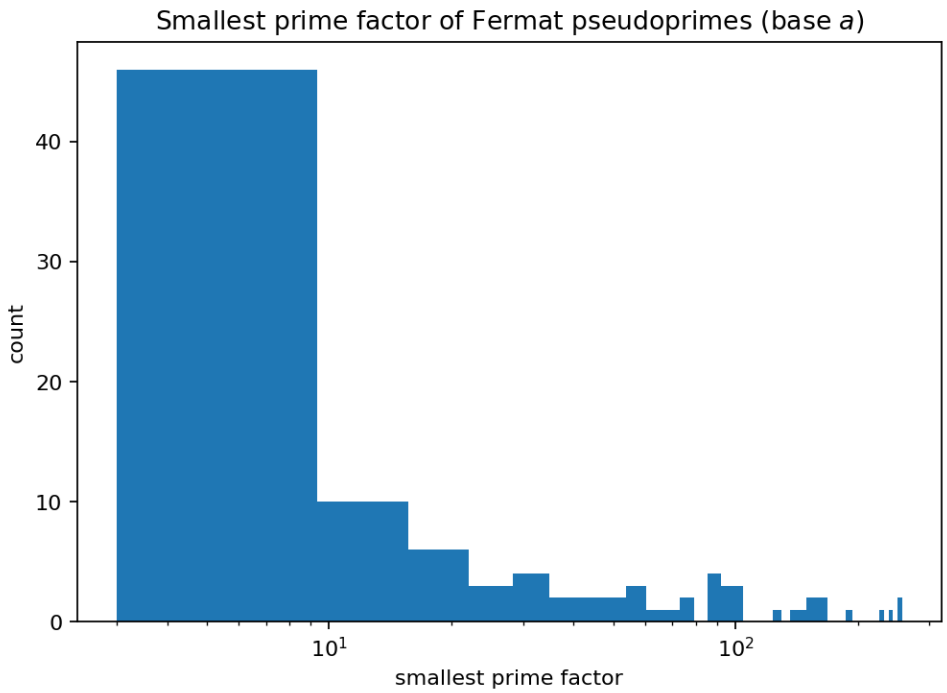
- *Prime numbers refresher*

5.12.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.12.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e012/` with a parameter snapshot.



5.12.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_01_cumulative_counts.png`
- `figures/fig_02_smallest_factor_hist.png`
- `params.json`
- `report.md`

5.12.7 How to run

```
make run EXP=e012
```

or:

```
uv run python -m mathxlab.experiments.e012
```

5.12.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.12.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.12.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e012 ARGS="--seed 1"
```

Parameters

- seed: `1`
- n_max: `200000`
- base a: `2`
- max_listed: `25`

5.12.11 Summary

- Fermat pseudoprimes to base 2: `106`
- Carmichael numbers: `22`

5.12.12 First examples (Fermat pseudoprimes)

#	n
1	341
2	561
3	645
4	1105
5	1387
6	1729
7	1905
8	2047
9	2465
10	2701
11	2821
12	3277
13	4033
14	4369
15	4371
16	4681
17	5461
18	6601
19	7957
20	8321
21	8481
22	8911
23	10261
24	10585
25	11305

5.12.13 First examples (Carmichael numbers)

#	n
1	561
2	1105
3	1729
4	2465
5	2821
6	6601
7	8911
8	10585
9	15841
10	29341
11	41041
12	46657
13	52633
14	62745
15	63973
16	75361
17	101101
18	115921
19	126217
20	162401
21	172081
22	188461

Notes

- Fermat's test is one-way: primes pass, but some composites also pass.
- Carmichael numbers are the strongest counterexamples: they pass for all coprime bases.
- Detection here uses Korselt's criterion (squarefree + divisibility conditions).

params.json (snapshot)

```
{
  "base": 2,
  "max_listed": 25,
  "n_max": 200000,
  "seed": 1
}
```

5.12.14 References

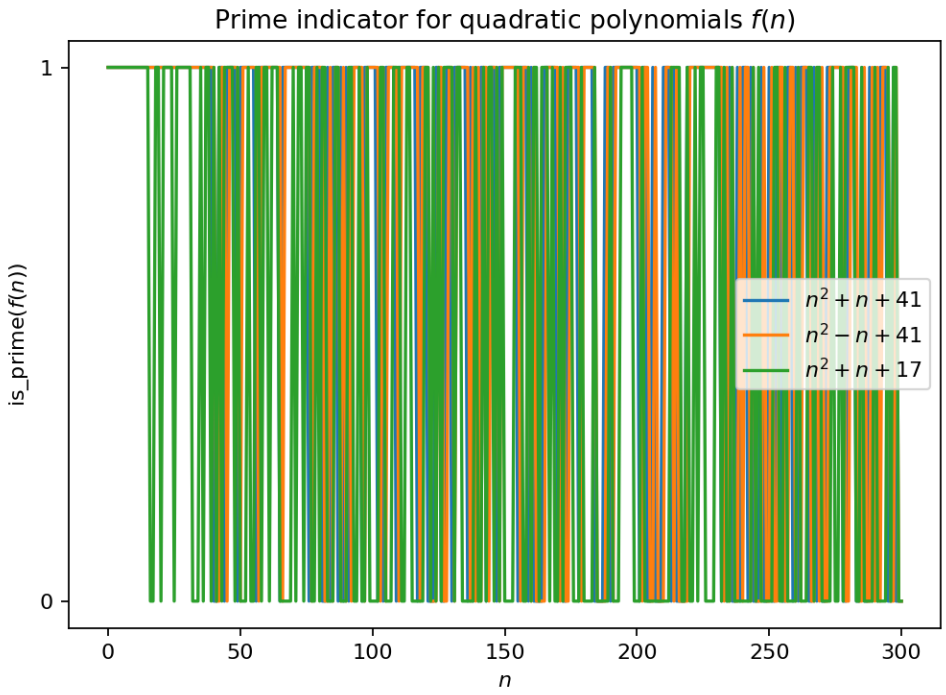
See ../references.

5.12.15 Related experiments

- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Prime-polynomial counterexamples (Euler's $n^2 + n + 41$))
- *E014: Primorial ± 1 counterexamples* (Primorial ± 1 counterexamples)
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)

- *E047: Fermat numbers: Pépin test + factor witnesses* (Fermat numbers: Pépin test + factor witnesses)

5.13 E013: Prime-polynomial counterexamples (Euler's n^2+n+41)



Tags: `number-theory`, `counterexample-search`, `visualization` See: *Valid Tags*.

5.13.1 Highlights

- Turn "prime-generating polynomials" folklore into a crisp counterexample (Euler's n²+n+41).
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.13.2 Goal

Turn "prime-generating polynomials" folklore into a crisp counterexample (Euler's n²+n+41).

5.13.3 Background (quick refresher)

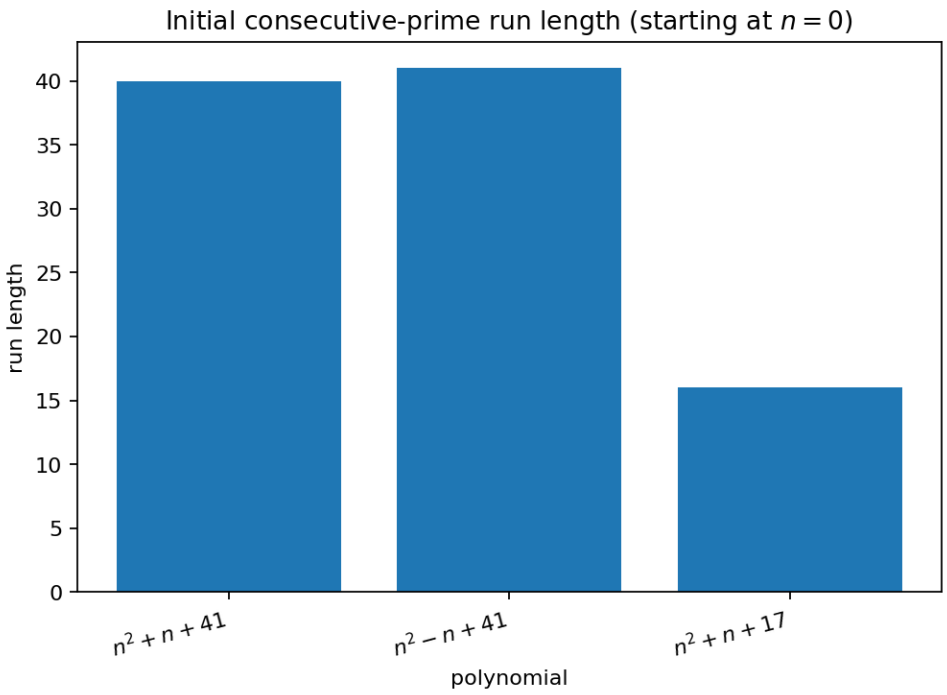
- *Prime numbers refresher*

5.13.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.13.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).



- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e013/` with a parameter snapshot.

5.13.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_01_prime_indicator.png`
- `figures/fig_02_initial_prime_run_lengths.png`
- `params.json`
- `report.md`

5.13.7 How to run

```
make run EXP=e013
```

or:

```
uv run python -m mathxlab.experiments.e013
```

5.13.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.13.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.13.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e013
```

Parameters

- n_max: `300`
- max_listed: `10`

5.13.11 Summary table

polynomial	initial prime run	first composite n	f(n)	factorization
`$n^2 + n + 41$`	40	40	1681	41^2
`$n^2 - n + 41$`	41	41	1681	41^2
`$n^2 + n + 17$`	16	16	289	17^2

Notes

- Euler's polynomial f(n)=n^2+n+41 is prime for n=0..39, but f(40)=41^2 is composite.
- Quadratics can look 'prime-rich' on small ranges, which is a classic trap for intuition.
- This experiment focuses on the *first* visible failure (counterexample) for each polynomial.

5.13.12 Outputs

- `figures/fig_01_prime_indicator.png`
- `figures/fig_02_initial_prime_run_lengths.png`
- `params.json`
- `report.md`

params.json (snapshot)


```
{
  "max_listed": 10,
  "n_max": 300,
  "seed": 1
}
```

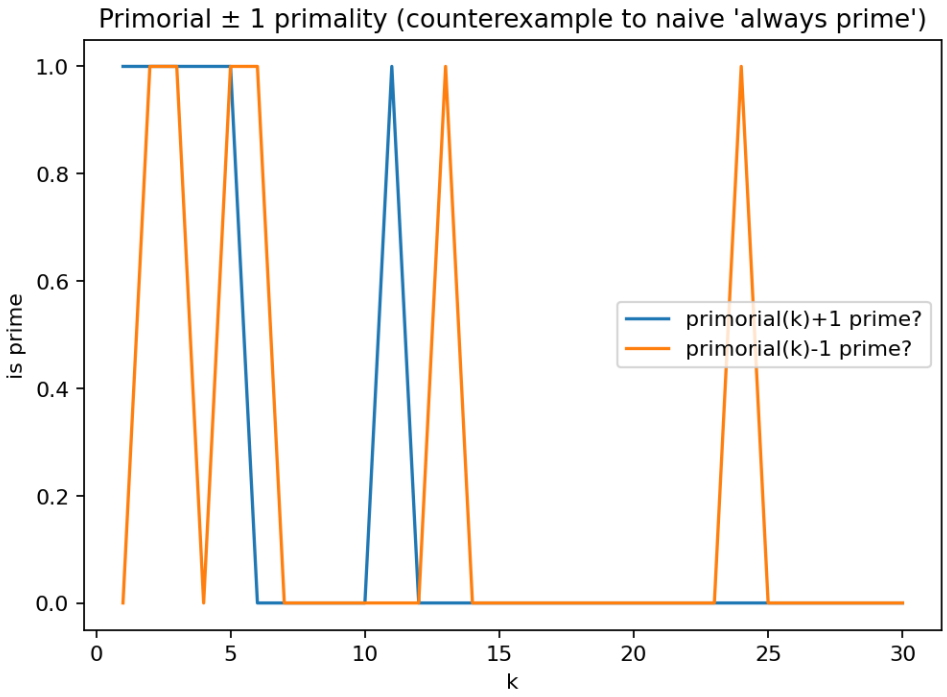

5.13.13 References

See ../references.

5.13.14 Related experiments

- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E014: Primorial ± 1 counterexamples* (Primorial ± 1 counterexamples)
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (Quadratic prime-run atlas around (a,b))
- *E128: Quadratic modular obstructions (Euler-type)* (Modular obstructions that force failures)
- *E129: Euler lucky constants for n^2 + n + b* (Euler lucky constants for $n^2 + n + b$)

5.14 E014: Primorial ± 1 counterexamples



Tags: `number-theory`, `counterexample-search`, `visualization` See: *Valid Tags*.

5.14.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.14.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.14.3 Background (quick refresher)

- *Prime numbers refresher*

5.14.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.14.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e014/` with a parameter snapshot.

5.14.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.14.7 How to run

```
make run EXP=e014
```

or:

```
uv run python -m mathxlab.experiments.e014
```

5.14.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.14.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.14.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e014
```

Parameters

- k_max: `30`

5.14.11 First composite examples

- first composite primorial(k)+1 at k=6: `30031` $= 59 \cdot 509$
- first composite primorial(k)-1 at k=1: `1` $= 1$

Notes

- Euclid's proof uses the idea `P+1` (where P is a product of primes) to show *some* new prime exists.
- It does **not** imply `P±1` is itself prime; composites appear very early.

5.14.12 Outputs

- `figures/fig_01_primorial_pm1_prime_indicator.png`
- `params.json`
- `report.md`

params.json (snapshot)

```
{
  "k_max": 30
}
```

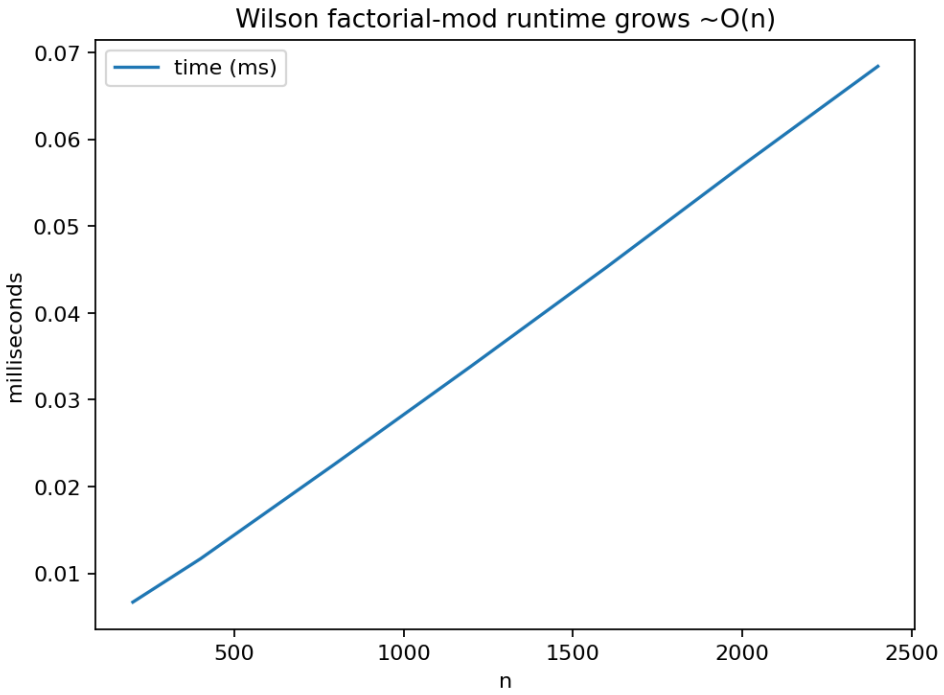
5.14.13 References

See ../references.

5.14.14 Related experiments

- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Prime-polynomial counterexamples (Euler's $n^2 + n + 41$))
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E005: Odd Perfect Numbers — Constraint Filter Pipeline* (Odd Perfect Numbers — Constraint Filter Pipeline)

5.15 E015: Wilson test infeasibility



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.15.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.15.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.15.3 Background (quick refresher)

- *Prime numbers refresher*

5.15.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.15.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e015/` with a parameter snapshot.

5.15.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.15.7 How to run

```
make run EXP=e015
```

or:

```
uv run python -m mathxlab.experiments.e015
```

5.15.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.15.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.15.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e015
```

Parameters

- n_values: `[200, 400, 800, 1200, 1600, 2000, 2400]`

Notes

- Wilson's theorem says (n-1)! ≡ -1 (mod n) iff n is prime.
- This is a beautiful characterization, but computing (n-1)! mod n is linear-time in n and becomes impractical fast.

5.15.11 Outputs

- `figures/fig_01_wilson_runtime.png`
- `params.json`
- `report.md`

params.json (snapshot)


```
{
  "n_values": [
    200,
    400,
    800,
    1200,
    1600,
    2000,
    2400
  ]
}
```

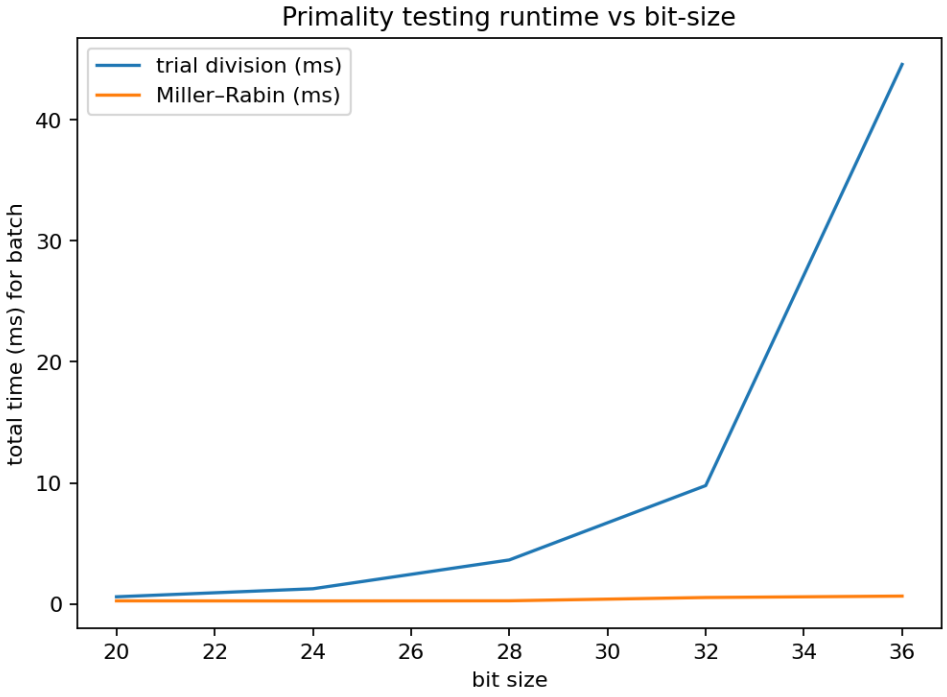

5.15.12 References

See ../references.

5.15.13 Related experiments

- *E047: Fermat numbers: Pépin test + factor witnesses* (Fermat numbers: Pépin test + factor witnesses)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))

5.16 E016: Trial division vs. Miller–Rabin scaling



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.16.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.16.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.16.3 Background (quick refresher)

- *Prime numbers refresher*

5.16.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.16.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e016/` with a parameter snapshot.

5.16.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.16.7 How to run

```
make run EXP=e016
```

or:

```
uv run python -m mathxlab.experiments.e016
```

5.16.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.16.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.16.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e016
```

Parameters

- bit_sizes: `[20, 24, 28, 32, 36]`
- samples_per_size: `200`

Notes

- Trial division is fine for small n, but runtime grows quickly with sqrt(n).
- Miller–Rabin stays fast and is the practical default for big integers.

params.json (snapshot)


```
{
  "bit_sizes": [
    20,
    24,
    28,
    32,
    36
  ],
  "samples_per_size": 200
}
```

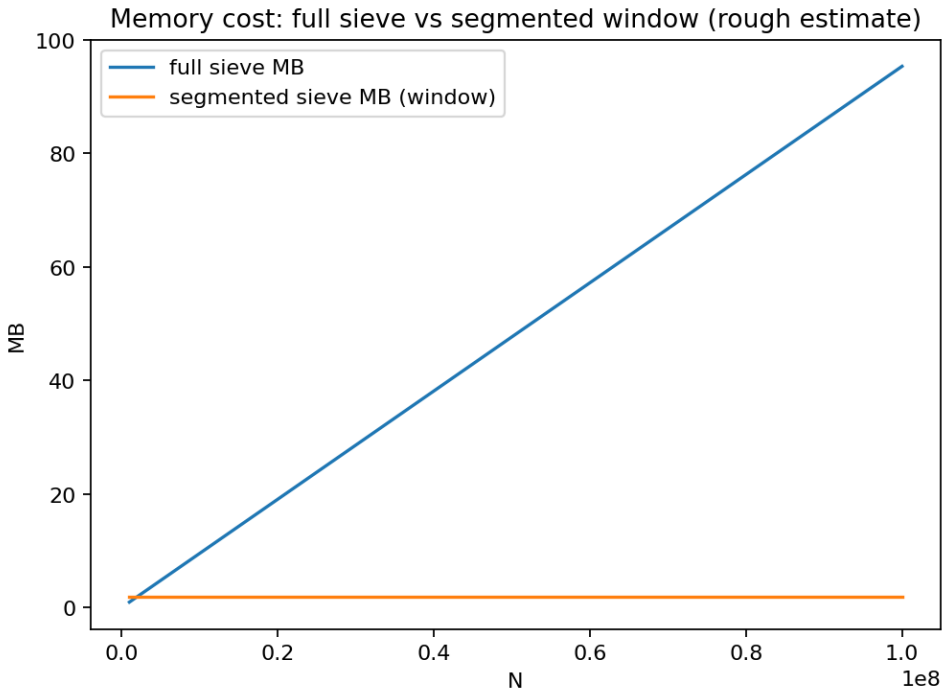

5.16.11 References

See ../references.

5.16.12 Related experiments

- *E044: Solovay–Strassen vs. Miller–Rabin (liars)* (Solovay–Strassen vs. Miller–Rabin (liars))
- *E105: Mertens M(x): scaling views* (E105: Mertens M(x): scaling views)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.17 E017: Sieve memory blow-up vs. segmented sieve



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.17.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.17.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.17.3 Background (quick refresher)

- *Prime numbers refresher*

5.17.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.17.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e017/` with a parameter snapshot.

5.17.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.17.7 How to run

```
make run EXP=e017
```

or:

```
uv run python -m mathxlab.experiments.e017
```

5.17.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.17.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.17.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e017
```

Parameters

- n_values: `[1000000, 5000000, 10000000, 50000000, 100000000]`
- segment_size: `2000000`

Notes

- A full sieve uses O(N) memory and can become the limiting factor.
- A segmented sieve keeps memory roughly constant by processing windows [L, R].

params.json (snapshot)

```
{
  "n_values": [
    1000000,
    5000000,
    10000000,
```

(continues on next page)

(continued from previous page)

```
    50000000,
    100000000
  ],
  "segment_size": 2000000
}
```

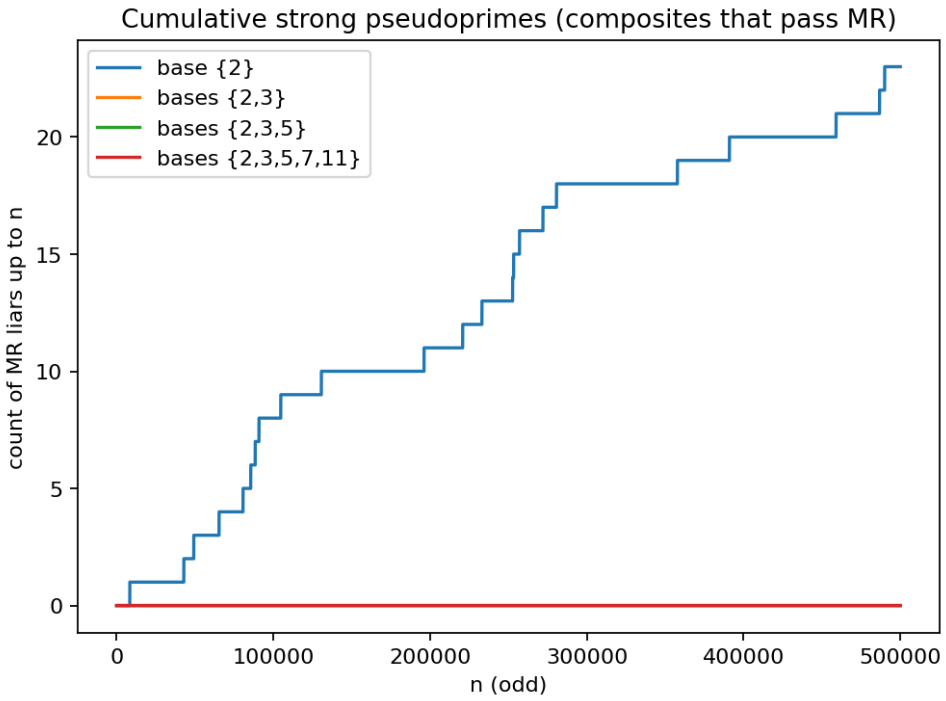
5.17.11 References

See ../references.

5.17.12 Related experiments

- *E004: Computing \sigma(n) at Scale — Sieve vs. Factorization* (Computing $\sigma(n)$ at Scale — Sieve vs. Factorization)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))

5.18 E018: Miller–Rabin base choice counterexamples



Tags: `number-theory`, `counterexample-search`, `visualization` See: *Valid Tags*.

5.18.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).

- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.18.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.18.3 Background (quick refresher)

- *Prime numbers refresher*

5.18.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.18.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e018/` with a parameter snapshot.

5.18.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.18.7 How to run

```
make run EXP=e018
```

or:

```
uv run python -m mathxlab.experiments.e018
```

5.18.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.18.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.18.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e018
```

Parameters

- n_max: 500000

Notes

- Miller–Rabin is reliable when you use enough bases (or a deterministic base set for bounded ranges).
- Using too few bases creates rare-but-real false positives (strong pseudoprimes).

params.json (snapshot)

```
{
  "n_max": 500000
}
```

5.18.11 References

See ../references.

5.18.12 Related experiments

- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Prime-polynomial counterexamples (Euler's $n^2 + n + 41$))
- *E014: Primorial ± 1 counterexamples* (Primorial ± 1 counterexamples)
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E049: Wieferich primes (base 2): scan and quotient visualization* (Wieferich primes (base 2): scan and quotient visualization)

5.19 E019: Prime counting and a PNT baseline

Published run: $\mathbf{n}_{\max} = 200\,000$ **and** $\pi(\mathbf{n}_{\max}) = 17\,984$

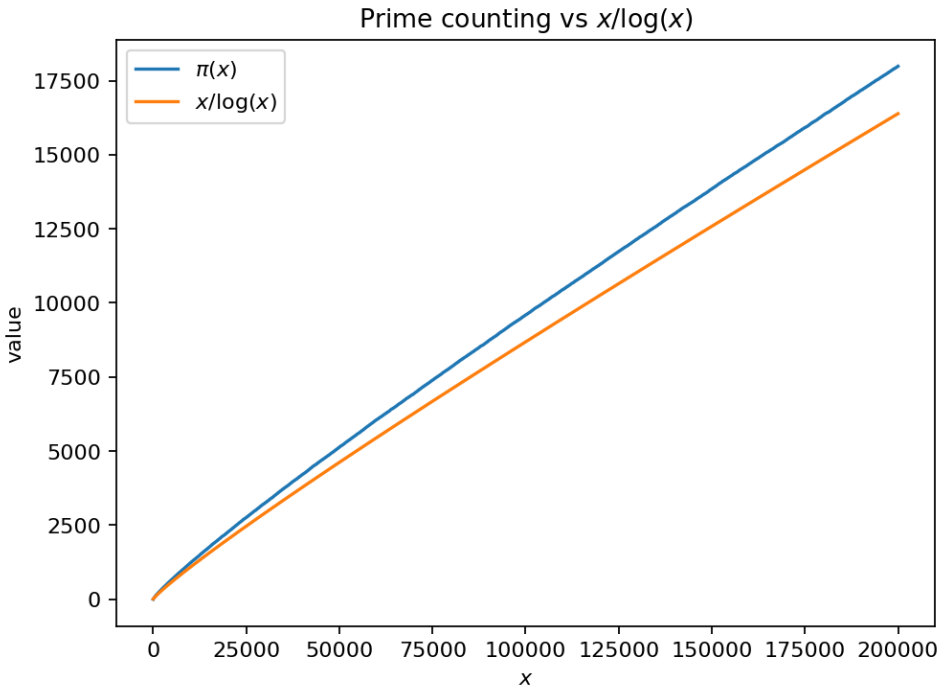
Published run: $\mathbf{n}_{\max} = 2\,000\,000$ **and** $\pi(\mathbf{n}_{\max}) = 148\,933$

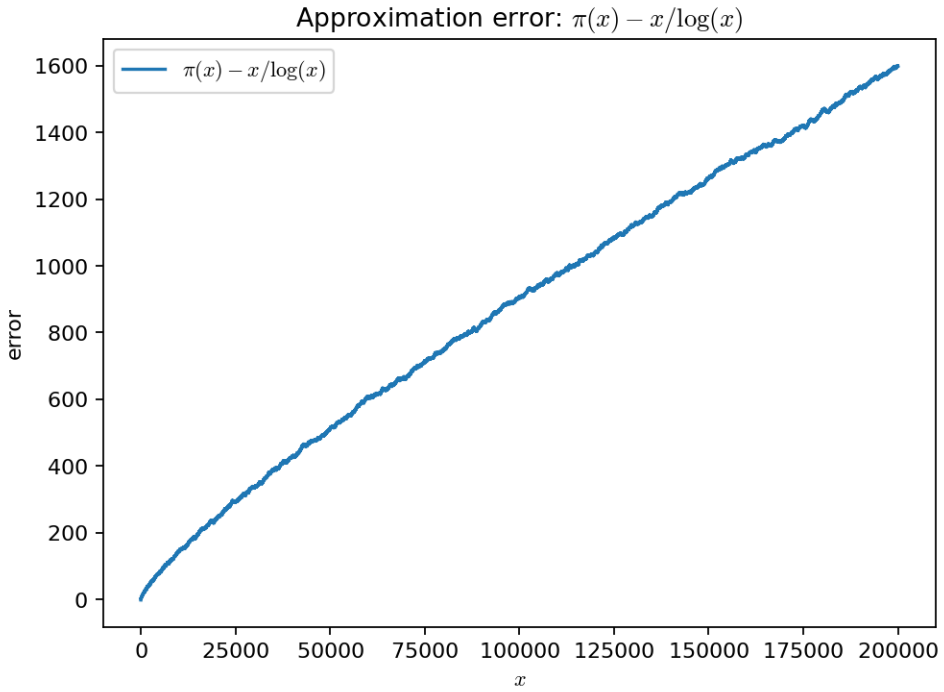
Published run: $\mathbf{n}_{\max} = 10\,000\,000$ **and** $\pi(\mathbf{n}_{\max}) = 664\,579$

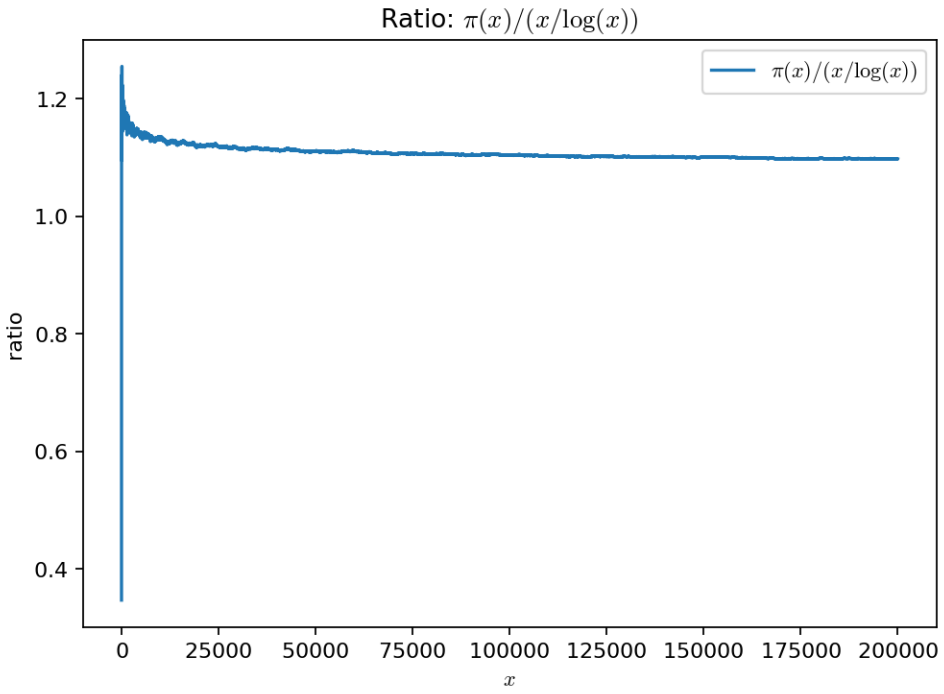
Published run: $\mathbf{n}_{\max} = 30\,000\,000$ **and** $\pi(\mathbf{n}_{\max}) = 1\,857\,859$

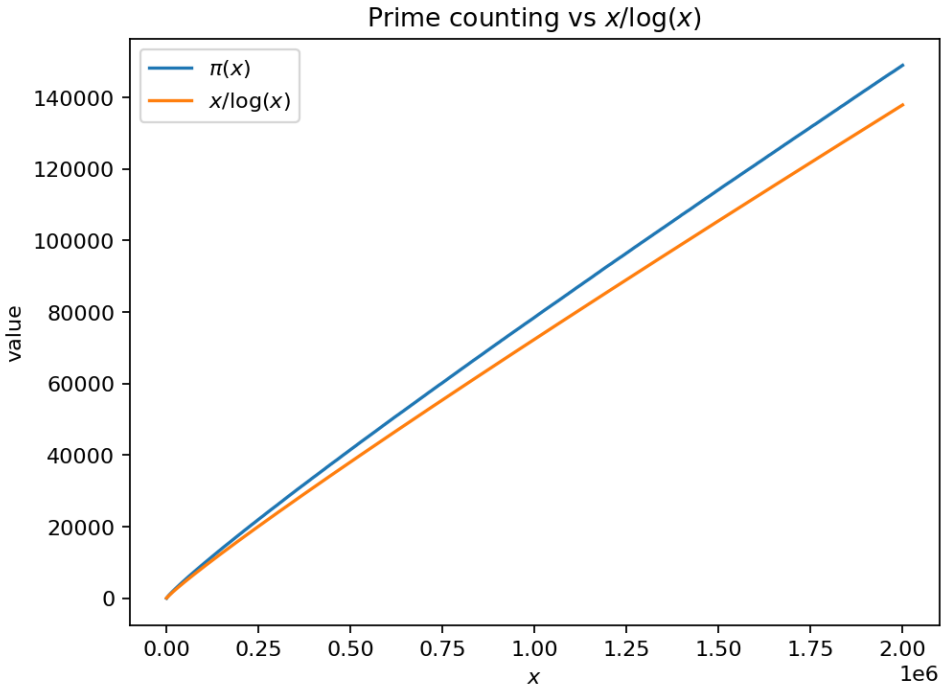
Prime counts shown are $\pi(n_{\max})$*, i.e., the number of primes* $\leq n_{\max}$. In this experiment we evaluate $\pi(x)$ for all integers x with $2 \leq x \leq \mathrm{n}_{\max}$.

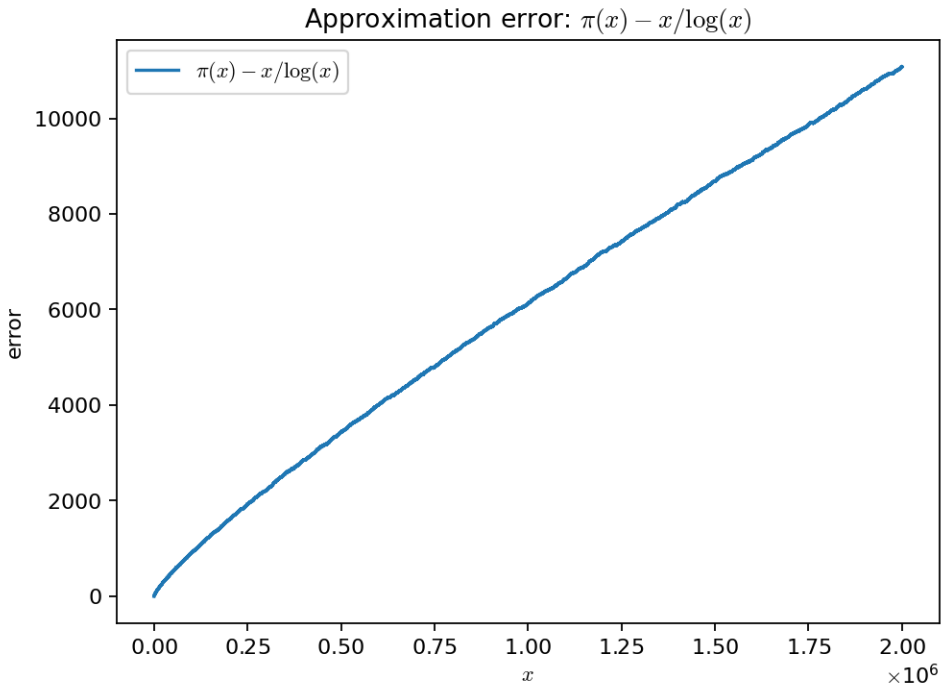
Tags: `number-theory`, `quantitative-exploration`, `visualization`
See: *Valid Tags*.

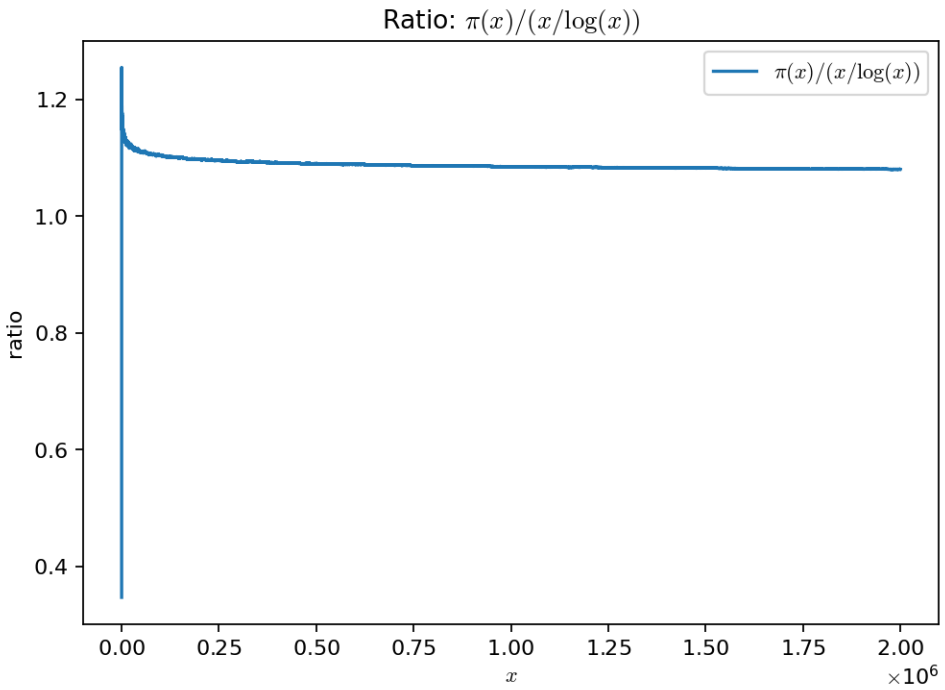

Prime counting vs x/log(x)
π(x)
x/log(x)
value
x
0
2500
5000
7500
10000
12500
15000
17500
0
25000
50000
75000
100000
125000
150000
175000
200000

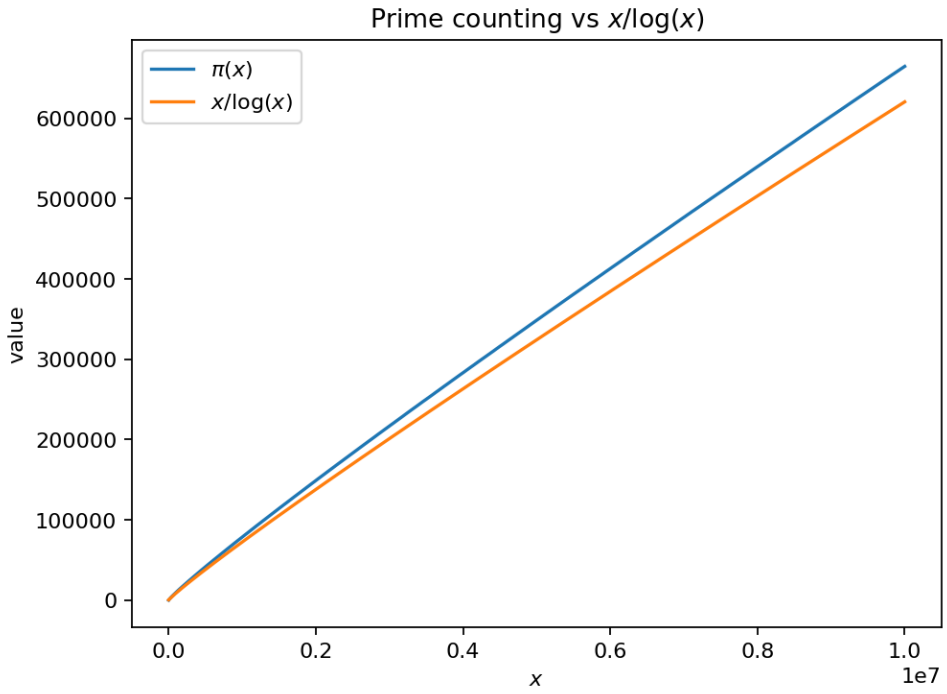

Approximation error: π(x) − x/log(x)
π(x) − x/log(x)
error
x
0
200
400
600
800
1000
1200
1400
1600
0
25000
50000
75000
100000
125000
150000
175000
200000

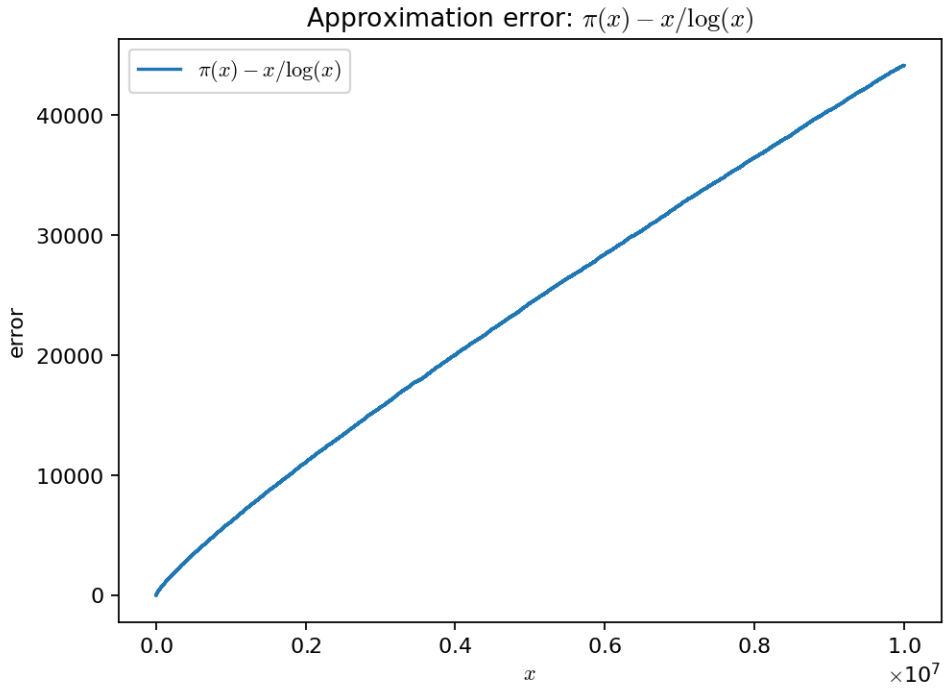

Ratio: $\pi(x)/(x/\log(x))$
$\pi(x)/(x/\log(x))$
1.2
1.0
0.8
0.6
0.4
ratio
0
25000
50000
75000
100000
125000
150000
175000
200000
x

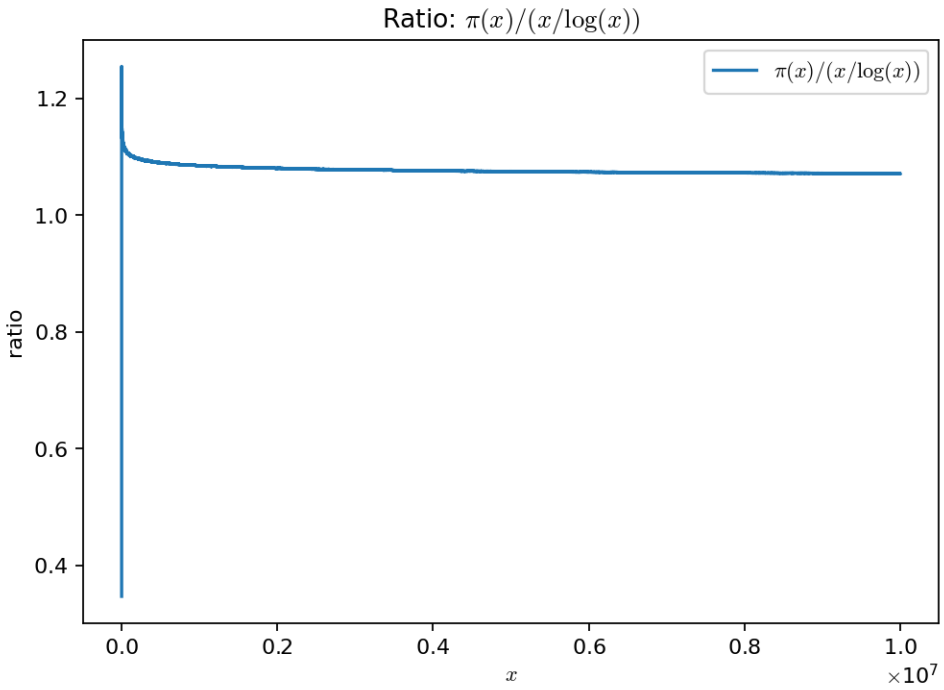

Prime counting vs $x/\log(x)$
$\pi(x)$
$x/\log(x)$
140000
120000
100000
80000
60000
40000
20000
0
value
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
x

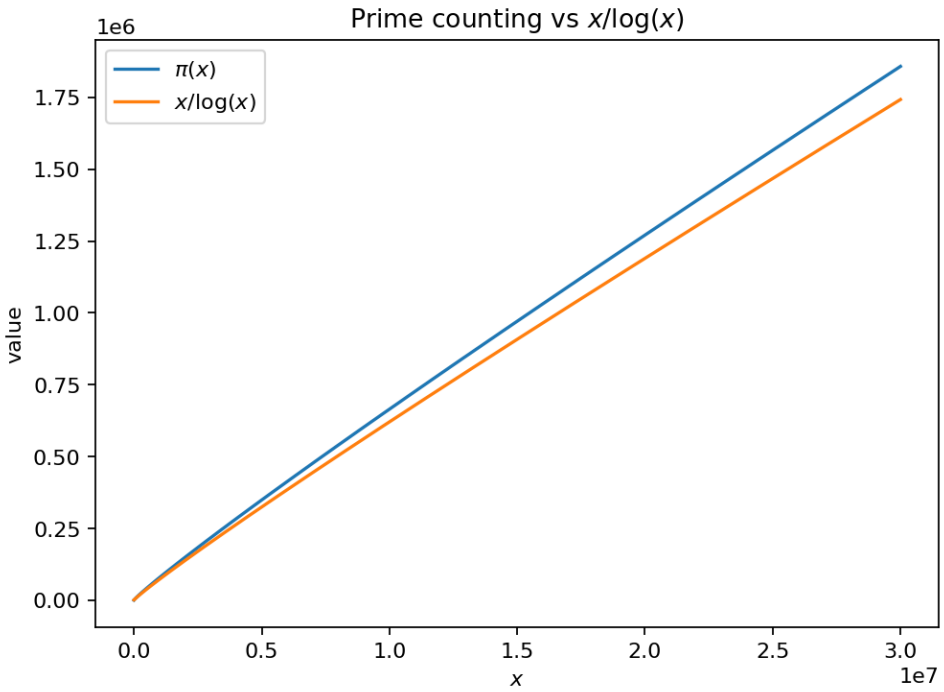

Approximation error: $\pi(x) - x/\log(x)$
$\pi(x) - x/\log(x)$
error
10000
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x
$\times 10^6$

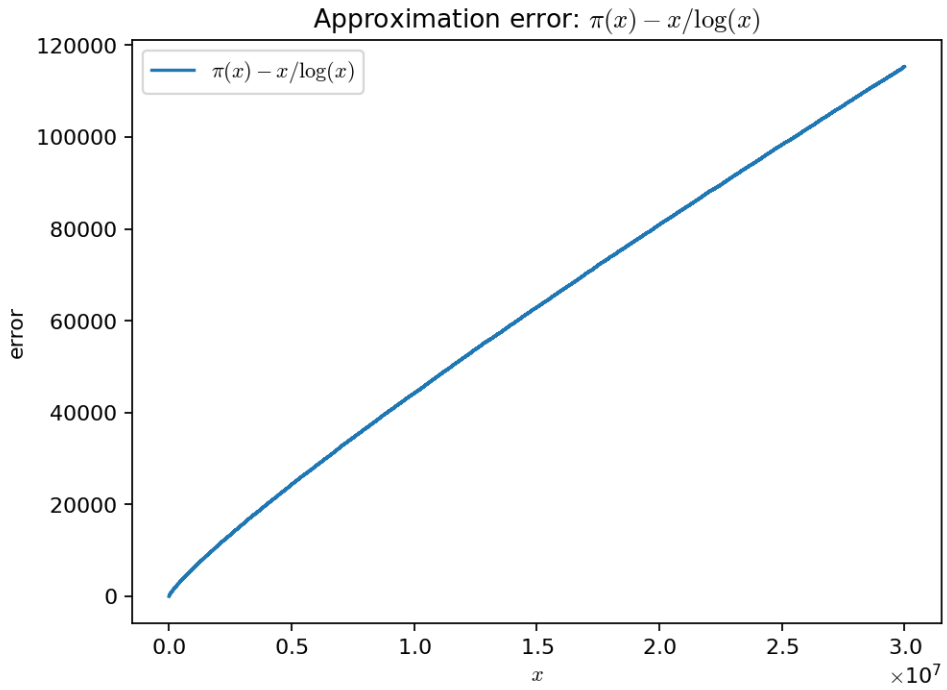

Ratio: $\pi(x)/(x/\log(x))$
$\pi(x)/(x/\log(x))$
ratio
1.2
1.0
0.8
0.6
0.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x
$\times 10^6$

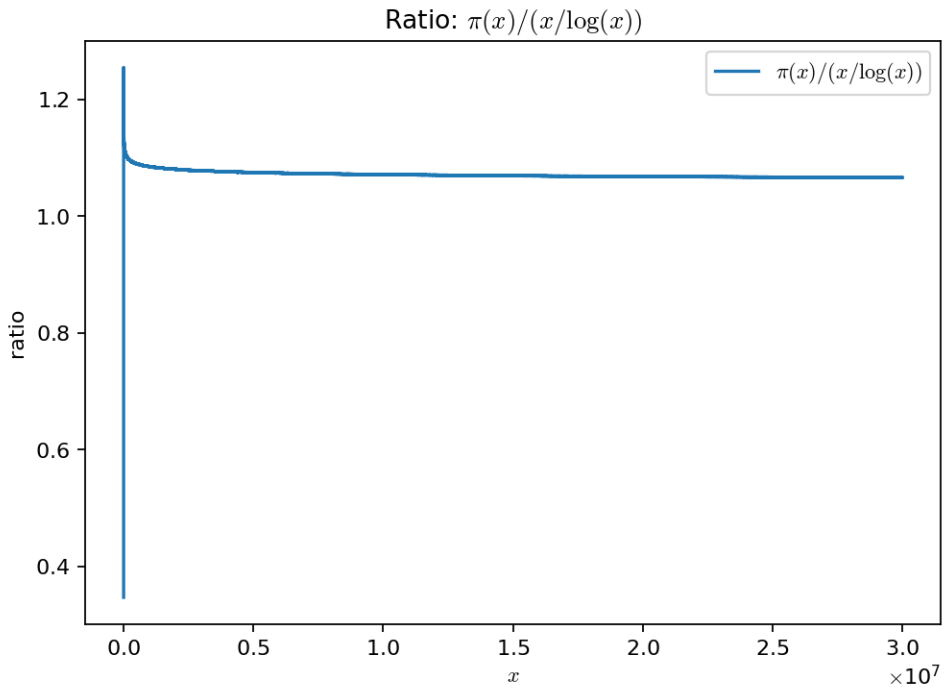

Prime counting vs x/log(x)
π(x)
x/log(x)
value
600000
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7
x


Approximation error: π(x) − x/log(x)
π(x) − x/log(x)
error
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0
×10^7
x


Ratio: $\pi(x)/(x/\log(x))$
$\pi(x)/(x/\log(x))$
ratio
1.2
1.0
0.8
0.6
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x
$\times 10^7$


Prime counting vs $x/\log(x)$
1e6
$\pi(x)$
$x/\log(x)$
value
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
1e7


Approximation error: $\pi(x) - x/\log(x)$
$\pi(x) - x/\log(x)$
error
120000
100000
80000
60000
40000
20000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
$\times 10^7$


Ratio: $\pi(x)/(x/\log(x))$
$\pi(x)/(x/\log(x))$
ratio
1.2
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
$\times 10^7$

5.19.1 Highlights

- Visual sanity check for your prime pipeline: compute $\pi(x)$ from a sieve mask and compare to a standard smooth baseline.
- Two complementary plots: $\pi(x)$ vs. $x/\log(x)$, and the finite-range error curve $\pi(x) - x/\log(x)$.
- Ratio plot: $\pi(x)/(x/\log(x))$ makes relative deviation visible when the two curves nearly overlap.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures) into `out/e019/`.

5.19.2 Goal

Visualize how well the Prime Number Theorem baseline $x/\log(x)$ tracks the prime-counting function $\pi(x)$ on a **finite** range and makes the approximation error visible.

5.19.3 Background (quick refresher)

Prime-counting function

The **prime-counting function** $\pi(x)$ is the number of primes p with $p \le x$.

The PNT baseline

The Prime Number Theorem (PNT) states that

$$\pi(x) \sim \frac{x}{\log x} \quad \text{as } x \to \infty.$$

Here log is the natural logarithm. The symbol "$\sim$" is **asymptotic**: it does not claim the two expressions are close for small or moderate x.

Optional background pages

- *Prime counting approximations: (x), Li(x), and R(x)*
- *Prime counting: explicit bounds (not just asymptotics)*
- *Prime numbers refresher*

5.19.4 Research question

On the finite range $2 \le x \le \mathrm{n}_{\max}$:

1. How close is $x/\log(x)$ to $\pi(x)$ as x grows?
2. What does the finite-range error $\pi(x) - x/\log(x)$ look like (trend, scale, rough smoothness)?

5.19.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** compute primes up to $\mathrm{n}_{\max}$, then compute the cumulative count $\pi(x)$.
- **Observable(s):** the comparison curve $\pi(x)$ vs. $x/\log(x)$ and the finite-range error curve.
- **Parameter space:** $\mathrm{n}_{\max}$ (and, if you extend it, sampling choices such as linear vs. log-spaced grids).
- **Outcome:** figures plus a short report capturing the key observation and caveats.
- **Reproducibility:** outputs saved to `out/e019/` with a parameter snapshot.

5.19.6 Experiment design

- **Computation:**
  - Build a prime mask for $\{0, 1, \ldots, n_{max}\}$ using a sieve.
  - Convert the mask to a cumulative array so that $\pi(x)$ can be read off quickly.
  - For each integer x with $2 \le x \le n_{max}$, compute the baseline $x/\log(x)$.
- **Axes (what the plots mean):**
  - In the first plot, the horizontal axis is x and the vertical axis is a **count**: it shows $\pi(x)$ and $x/\log(x)$ for the same x values.
  - In the second plot, the horizontal axis is x and the vertical axis is the **count difference** $\pi(x) - x/\log(x)$.
- **Outputs (typical):**
  - `figures/fig_01_pi_vs_x_logx.png`
  - `figures/fig_02_pi_minus_x_logx.png`
  - `figures/fig_03_pi_over_x_logx_ratio.png`
  - `params.json`
  - `report.md`

5.19.7 How to run

```
make run EXP=e019
```

Override the upper bound (computes primes up to n_{max}):

```
make run EXP=e019 ARGS="--n-max 5000000"
```

Direct invocation (always works):

```
uv run --extra dev python -m mathxlab.experiments.e019 --out out/e019
# Override the upper bound:
uv run --extra dev python -m mathxlab.experiments.e019 --out out/e019 --n-max␣
↪5000000
```

5.19.8 Notes / pitfalls

- **Finite-range behavior:** the PNT statement is asymptotic; interpret these plots as "finite-range behavior".
- **Parameter exposure:** use `--n-max` to change n_{max}; keep published snapshots consistent with `params.json` / `report.md`.
- **Scale masking:** on a linear axis, $\pi(x)$ and $x/\log(x)$ are close in *relative* terms, so runs with different n_{max} can look very similar. Use the ratio plot $\pi(x)/(x/\log(x))$ for a relative view.
- **Log domain:** avoid $x = 0$ or $x = 1$ because $\log(x)$ is 0 or undefined there; we start at $x = 2$.
- **Visual overclaims:** a smooth-looking error curve does not imply a theorem about error terms.

5.19.9 Extensions

- Add a second baseline $\mathrm{Li}(x)$ and compare $\pi(x) - \mathrm{Li}(x)$.
- Add explicit bounds (Rosser–Schoenfeld / Dusart) and visualize which ranges they dominate.
- Add a sampling mode:

- **linear**: evaluate all integers x (current version), and
- **log**: evaluate a log-spaced grid to "see" larger scales more evenly.

5.19.10 Published run snapshot

Reproduce:

```
make run EXP=e019
```

Parameters

- n_max: 2000000

Notes

- The Prime Number Theorem suggests $\pi(x)$ $x/\log(x)$.
- The error curve shows the approximation improves overall but wiggles persist.
- Ratio view: $\pi(x)/(x/\log(x))$ highlights relative deviation.

params.json (snapshot)

```
{
  "n_max": 2000000
}
```

5.19.11 References

- Prime-counting function overview: [Wikipedia contributors, 2025].
- Analytic number theory background and PNT context: [Apostol, 1976].
- Classic reference for $\pi(x)$, error terms, and explicit-formula viewpoints: [Titchmarsh, 1986].

See also ../references.

5.19.12 Related experiments

- *E020: Compare pi(x) to li(x) numerically* (Prime counting: compare $\pi(x)$ to $\mathrm{Li}(x)$)
- *E021: Explicit bounds sanity checks* (Prime counting: explicit bounds for $\pi(x)$)
- *E024: Ulam spiral structure* (Ulam spiral: prime distribution visualization)

5.19.13 Parameters (example)

- n_{max}: 2_000_000 (range: $2 \le x \le n_{max}$)

Recommended n_{max} range

This experiment classifies primes up to n_{max} using a sieve, so runtime and memory scale roughly with n_{max}.

A practical range that works well for most runs:

- **Minimum (still meaningful):** $n_{max} = 200\,000$
- **Default / recommended:** $n_{max} = 2\,000\,000$
- **Comfortable upper range on typical laptops (varies):** $n_{max} = 10\,000\,000$
- **Above ~$n_{max} = 30\,000\,000$:** expect noticeably higher runtime (and potentially memory pressure), depending on the machine and sieve implementation.

5.19.14 Summary

We compute the prime-counting function $\pi(x)$ up to $n_{max} = 2\,000\,000$ and compare it to the smooth baseline $x/\log(x)$ on the same range.

5.19.15 Key observations

- The baseline $x/\log(x)$ tracks the overall growth of $\pi(x)$ but underestimates it on this range.
- The error curve $\pi(x) - x/\log(x)$ is positive and grows to about 11 084 at $x = 2\,000\,000$.

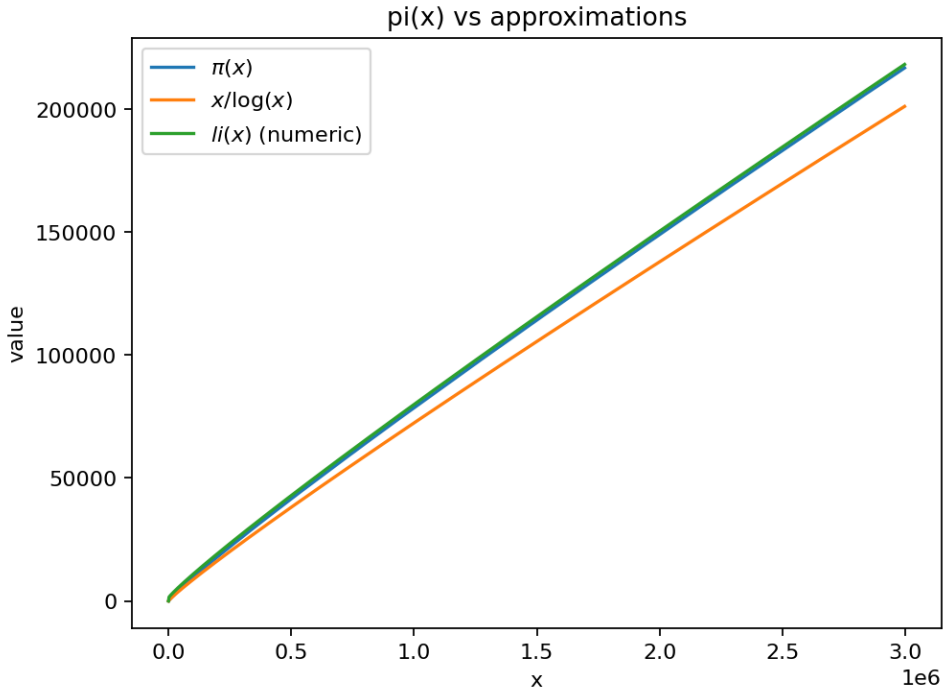
5.19.16 Interpretation

This is exactly the kind of behavior the PNT suggests: $\pi(x)$ is "close to" $x/\log(x)$ in a relative sense, but the absolute difference can still be large on finite ranges. The plots are therefore best read as **finite-range behavior** rather than as a direct numerical validation of the asymptotic statement.

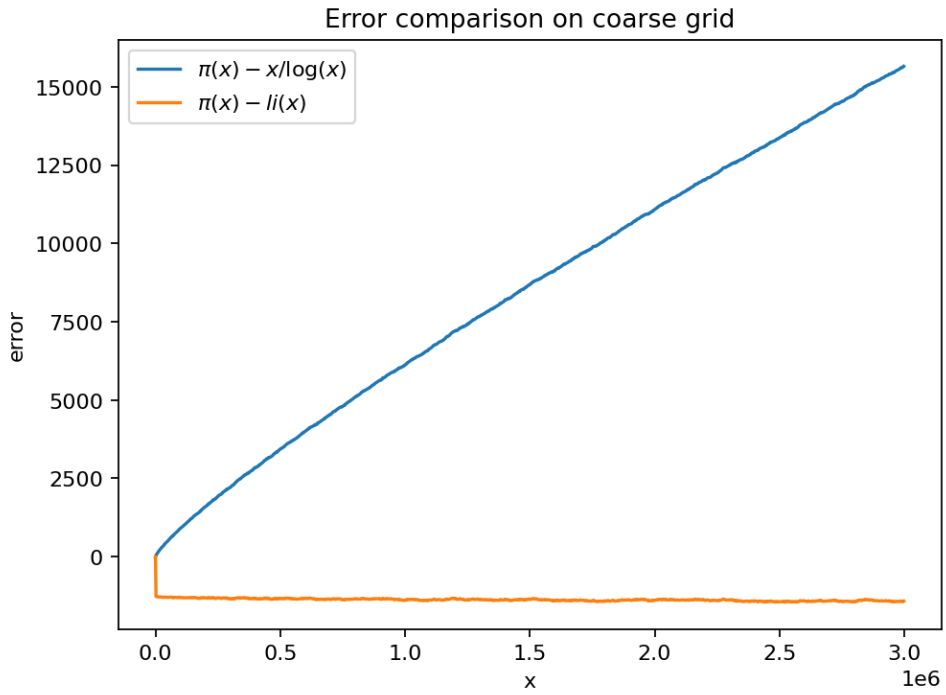
5.19.17 Caveats

- The PNT statement is asymptotic: do not treat any single finite-range plot as proof.
- Visualization choices (sampling density, line smoothing, axis scaling) can make convergence look better or worse than it is.
- For better baselines on moderate ranges, comparing to $\mathrm{Li}(x)$ is often more informative (see the related experiment).

5.20 E020: Compare pi(x) to li(x) numerically



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.



5.20.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.20.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.20.3 Background (quick refresher)

- *Prime numbers refresher*

5.20.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.20.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e020/` with a parameter snapshot.

5.20.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.20.7 How to run

```
make run EXP=e020
```

or:

```
uv run python -m mathxlab.experiments.e020
```

5.20.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.20.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.20.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e020
```

Parameters

- n_max: `3000000`
- step: `2000`

Notes

- li(x) is defined by an integral of 1/log t and often tracks $\pi(x)$ more closely than $x/\log(x)$.
- Here we use a coarse trapezoidal approximation (good enough for a visual experiment).

params.json (snapshot)

```
{
  "n_max": 3000000,
  "step": 2000
}
```

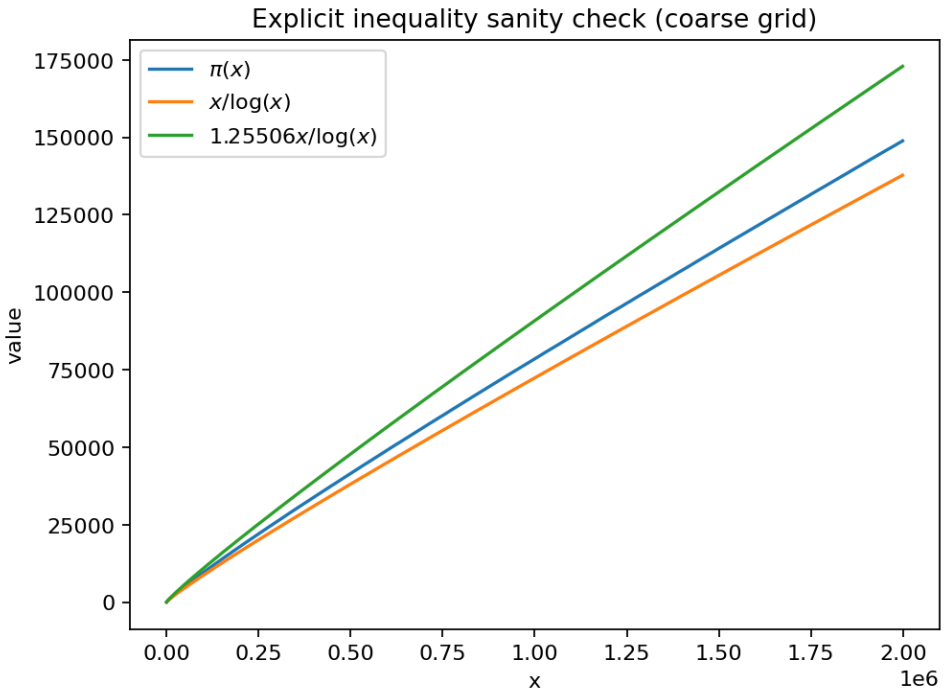
5.20.11 References

See ../references.

5.20.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.21 E021: Explicit bounds sanity checks



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.21.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.21.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.21.3 Background (quick refresher)

- *Prime numbers refresher*

5.21.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.21.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e021/` with a parameter snapshot.

5.21.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.21.7 How to run

```
make run EXP=e021
```

or:

```
uv run python -m mathxlab.experiments.e021
```

5.21.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.21.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.21.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e021
```

Parameters

- n_max: 2000000

Notes

- Many explicit inequalities for $\pi(x)$ have *starting points* (valid only for $x \geq x_0$).
- Experiments should always verify the assumptions before using a bound as a 'test oracle'.

params.json (snapshot)


```
{
  "n_max": 2000000
}
```

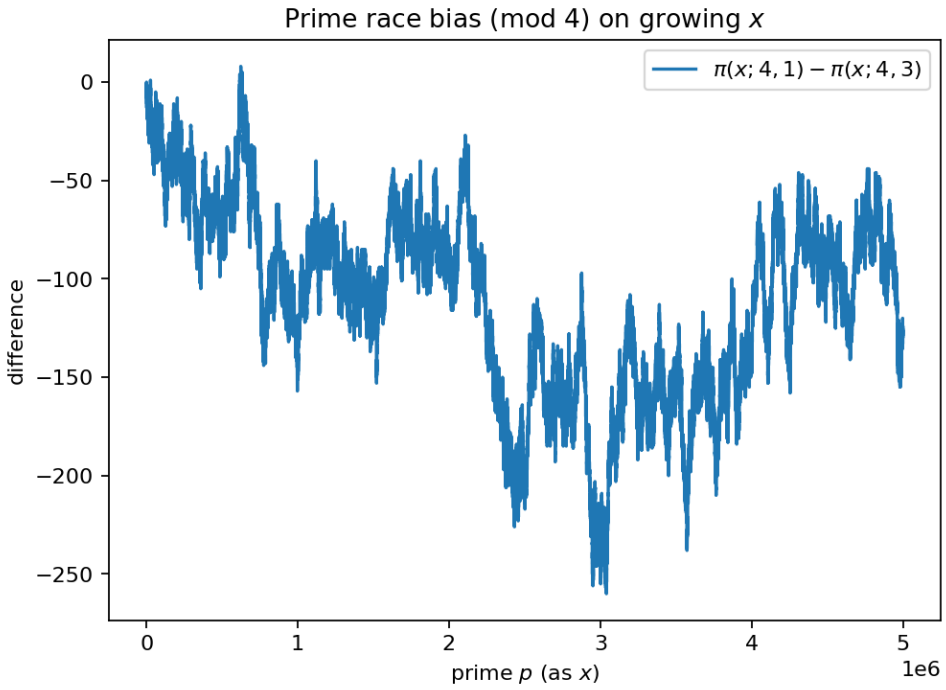

5.21.11 References

See ../references.

5.21.12 Related experiments

- *E077: Indicator via character orthogonality (sanity check).* (Indicator via character orthogonality (sanity check).)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))

5.22 E022: Prime race modulo 4



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.22.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.22.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.22.3 Background (quick refresher)

- *Prime numbers refresher*

5.22.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.22.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e022/` with a parameter snapshot.

5.22.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.22.7 How to run

```
make run EXP=e022
```

or:

```
uv run python -m mathxlab.experiments.e022
```

5.22.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.22.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.22.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e022
```

Parameters

- n_max: `5000000`

Notes

- Dirichlet's theorem implies each residue class (1 and 3 mod 4) gets infinitely many primes.
- Yet finite ranges show biases (the 'prime race' phenomenon).

params.json (snapshot)

```
{
  "n_max": 5000000
}
```

5.22.11 References

See ../references.

5.22.12 Related experiments

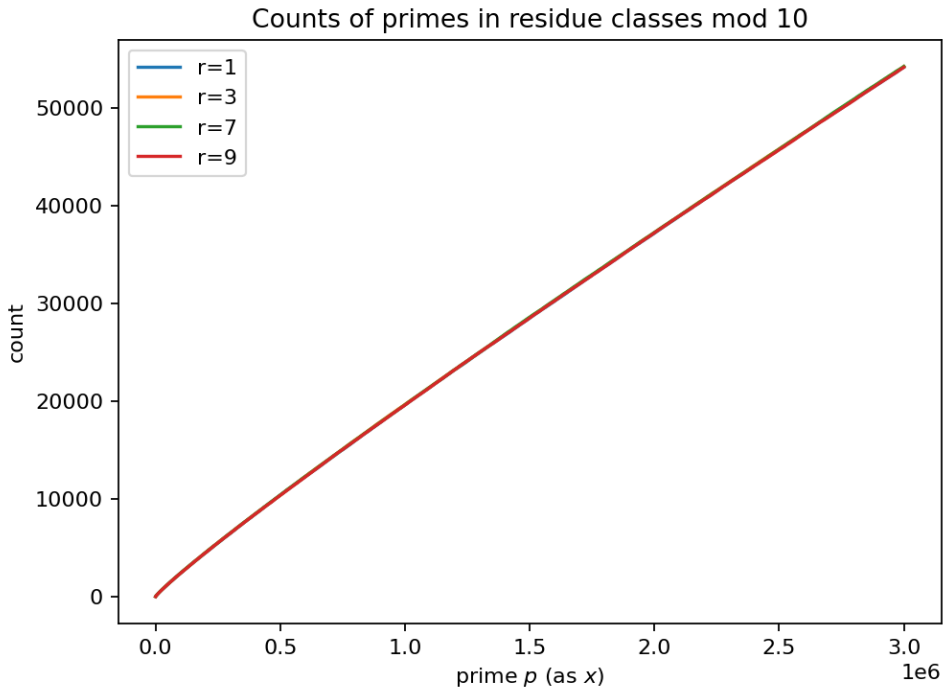
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.23 E023: Residue class distribution mod q

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.23.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.



5.23.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.23.3 Background (quick refresher)

- *Prime numbers refresher*

5.23.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.23.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e023/` with a parameter snapshot.

5.23.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.23.7 How to run

```
make run EXP=e023
```

or:

```
uv run python -m mathxlab.experiments.e023
```

5.23.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.23.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.23.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e023
```

Parameters

- n_max: `3000000`
- q: `10`

Notes

- In the limit, reduced residue classes should balance, but finite ranges can show visible drift.

params.json (snapshot)

```
{
  "n_max": 3000000,
  "q": 10
}
```

5.23.11 References

See ../references.

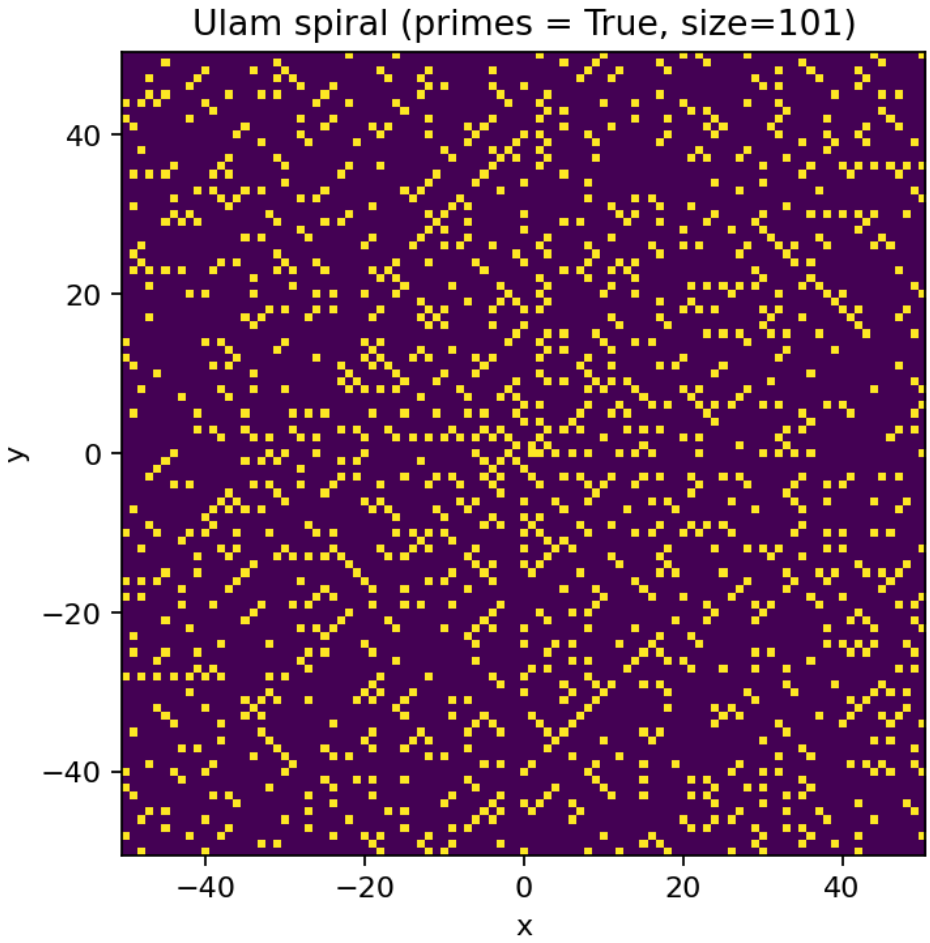
5.23.12 Related experiments

- *E113: First prime in each residue class* (E113: First prime in each residue class)
- *E070: Primes in residue classes: pi(x; q, a).* (Primes in residue classes: pi(x; q, a).)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)

5.24 E024: Ulam spiral structure

size 101: 10,201 numbers and 1,252 primes



size 201: 40,401 numbers and 4,236 primes

size 301: 90,601 numbers and 8,769 primes

size 401: 160,801 numbers and 14,752 primes

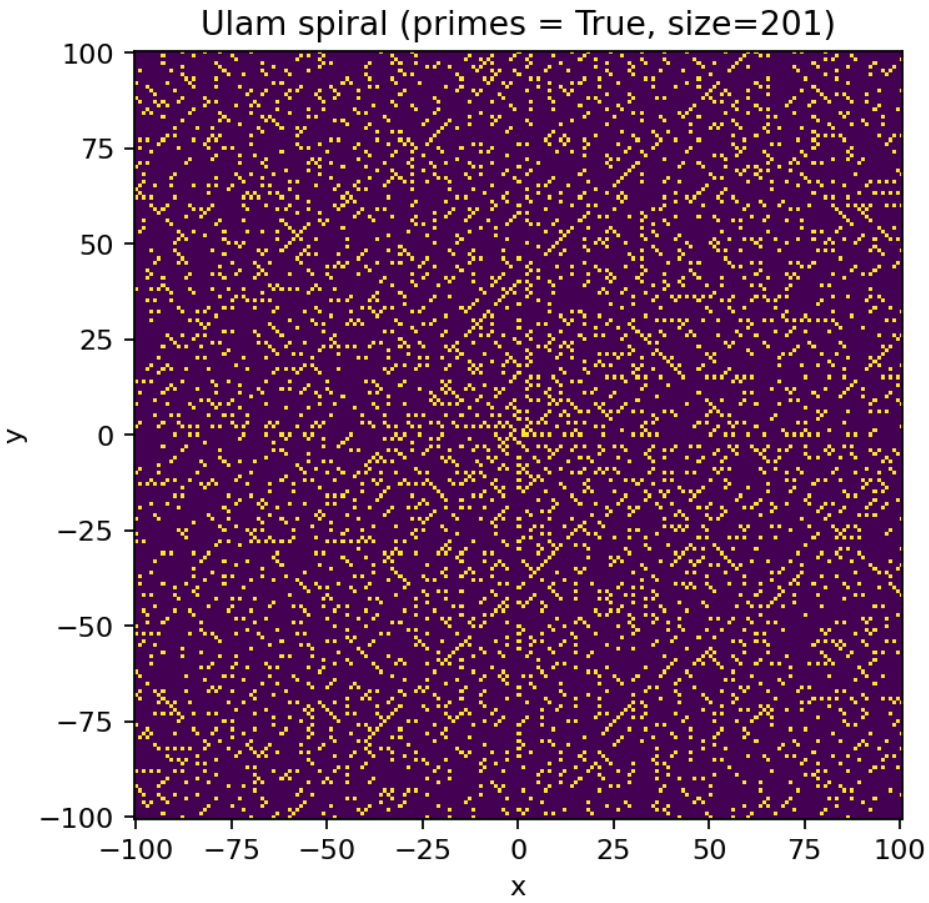
size 999: 998,001 numbers and 78,359 primes

Prime counts shown are $\pi(size^2)$, i.e., the number of primes $\leq size^2$.

Tags: `number-theory`, `quantitative-exploration`, `visualization`
See: *Valid Tags*.

5.24.1 Highlights

- Visual "pattern detector" for primes: highlights a diagonal structure that emerges in the Ulam spiral.
- Parameterizable run: vary `--size` from the command line (default: `301`).
- Writes reproducible artifacts (`params.json`, `report.md`, and figures) into `out/e024/`.



5.24.2 Goal

Render an Ulam spiral and highlight primes on the spiral grid to make diagonal structure (prime-rich diagonals) visible, then compare how that structure changes as the spiral size increases.

5.24.3 Background (quick refresher)

What is an Ulam spiral?

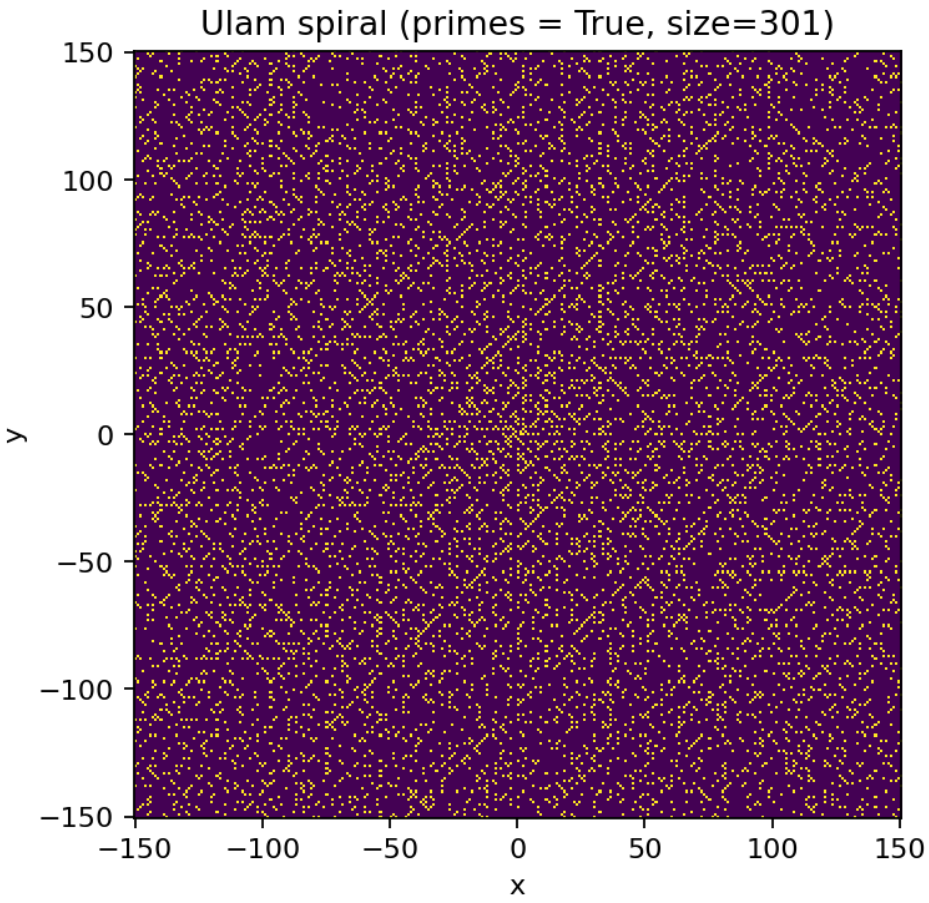
An **Ulam spiral** places the positive integers on a square spiral (starting at 1 in the center, then winding outward) and then marks which positions are prime. A striking visual phenomenon appears: **primes cluster along certain diagonals**. A common explanation is that many diagonals correspond to simple quadratic polynomials in the spiral index; some of these produce unusually many primes for small inputs. See [contributors, 2025].

Optional background pages

- *Prime numbers refresher*
- *Quadratic polynomials (algebraic) refresher*
- *Euler's prime-generating polynomial refresher*

5.24.4 Research question

Which diagonals (or families of diagonals) in the Ulam spiral show unusually high prime density for a given window size, and how does that visual structure change as `size` increases?

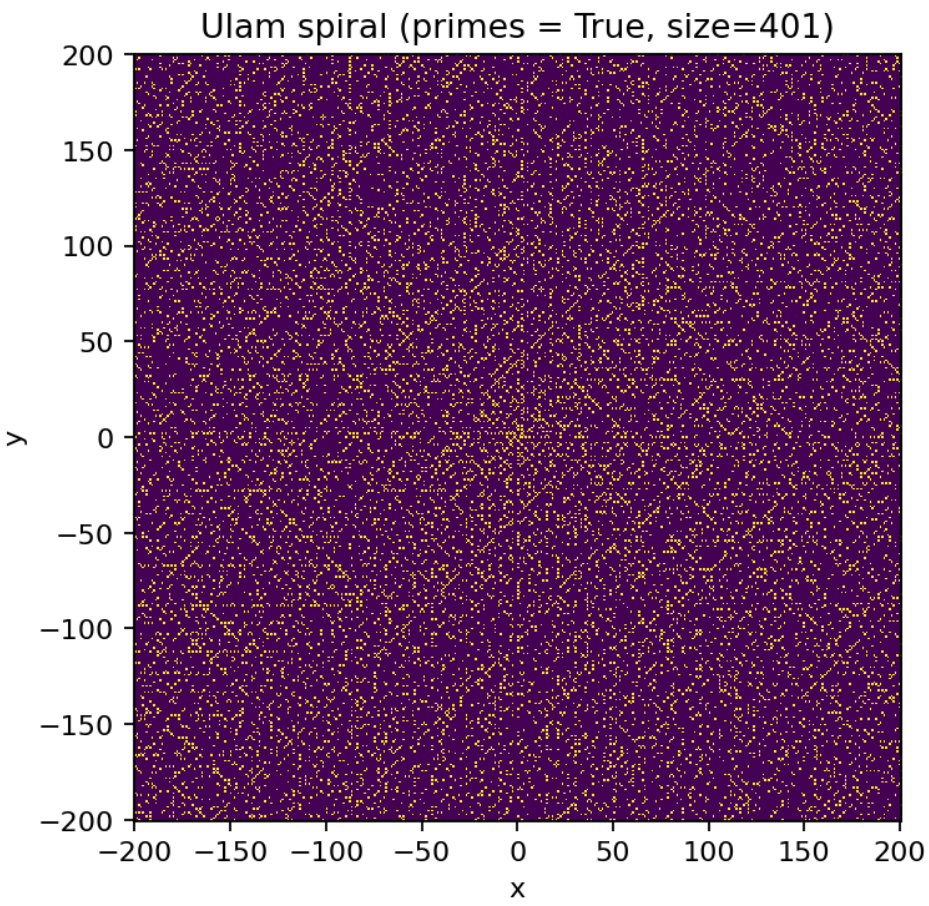


5.24.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** build a finite spiral window and run a deterministic primality test up to size^2.
- **Observable(s):** diagonal structure, local "prime-rich" streaks, and how visibility changes with `size`.
- **Parameter space:** the window size `size` (and, if you extend it, rendering/annotation choices).
- **Outcome:** figures and a short report capturing the key observation and caveats.
- **Reproducibility:** outputs saved to `out/e024/` with a parameter snapshot.

5.24.6 Experiment design

- **Computation:** map integers $1, \ldots,$ size^2 to lattice coordinates on a square spiral; mark primes.
- **Coordinates:** are centered at (0,0) in the middle of the grid; x increases to the right, y increases upward; the spiral starts with 1 at the origin, then moves right to 2, and continues counterclockwise outward.
- **Prime classification:** primes are computed using a sieve up to size^2.
- **Outputs:** one or more figures showing prime positions on the spiral grid.
- **Artifacts written:**
  - `figures/fig_01_ulam_spiral*.png` (main figure) and `figures/e024_hero_<size>.png` (published hero)
  - `params.json`
  - `report.md`



Plot axes and coordinate conventions

The spiral is plotted on a centered Cartesian grid:

- **x-axis:** horizontal offset from the center cell (0 at the center, negative left, positive right)
- **y-axis:** vertical offset from the center cell (0 at the center, negative down, positive up)

With an odd window size `size`, define $k = (\text{size} - 1)/2$. The visible coordinate range is then $x, y \in \{-k, \ldots, k\}$.

If the implementation renders a 2D array with Matplotlib `imshow`, make sure the axis tick labels match these offsets (using an `extent` such as `(-k-0.5, k+0.5, -k-0.5, k+0.5)` and a consistent `origin`). When `origin="upper"` is used, the y-axis is flipped visually; interpret tick labels accordingly.

5.24.7 How to run

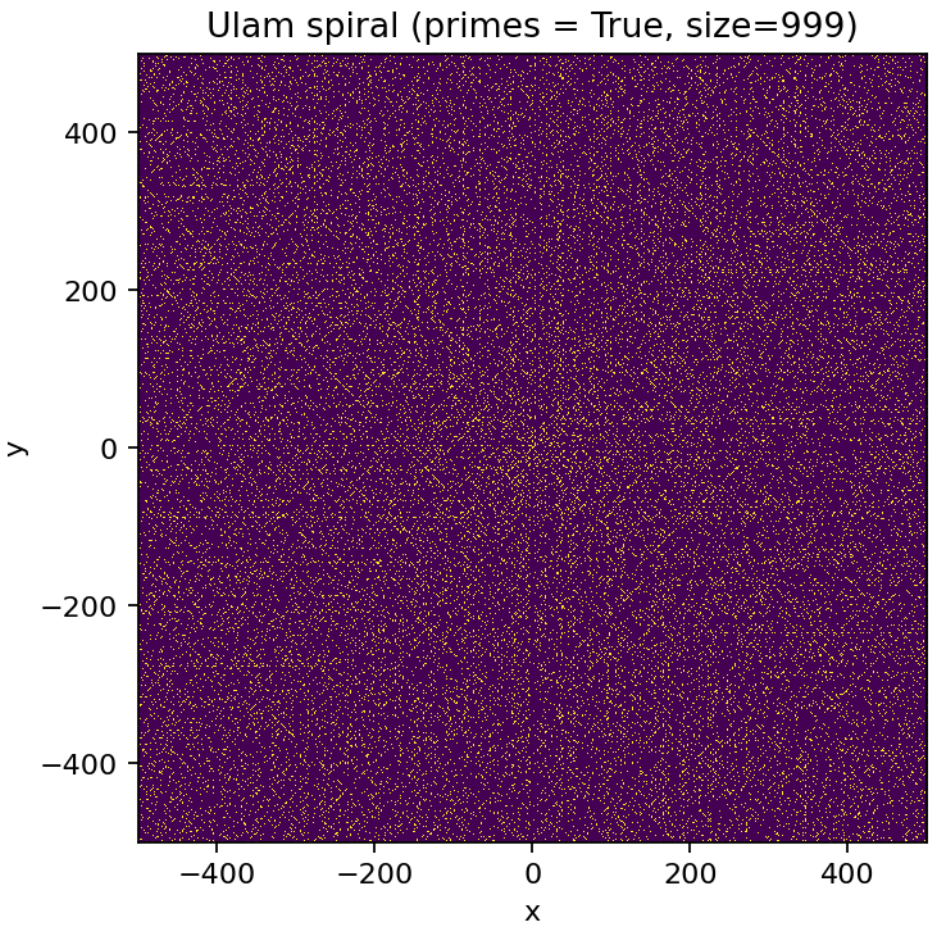
```
make run EXP=e024
```

Override the spiral size (must be **odd**):

```
make run EXP=e024 ARGS="--size 501"
```

Direct invocation (always works):

```
uv run --extra dev python -m mathxlab.experiments.e024 --out out/e024 --size 501
```


Ulam spiral (primes = True, size=999)
400
200
0
−200
−400
y
−400
−200
0
200
400
x

5.24.8 Notes / pitfalls

- `size` must be **odd** so that the spiral has a single center cell.
- Larger `size` means more numbers (size^2) and a longer run time.
- "Looks-true" trap: visual regularity suggests hypotheses but does not prove theorems about primes.

5.24.9 Extensions

- Add a "diagonal density" quantitative overlay: estimate prime density per diagonal and plot it.
- Annotate diagonals with the corresponding quadratic polynomials (for small windows).
- Compare multiple sizes side-by-side (same styling and axis conventions).

5.24.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e024
```

Parameters

- size: `301`

Notes

- Diagonal streaks correspond to quadratic polynomials that produce many primes for small n.
- This is a visual 'pattern trap': structure is real, but it does not imply a simple rule for primes.

params.json (snapshot)

```
{
  "size": 301
}
```

5.24.11 References

- Ulam spiral overview and basic properties: [contributors, 2025].
- Popular exposition and recreational-math context: [Gardner, 1983].
- Narrative-style discussion of mathematical discovery/pitfalls (useful for "looks-true" effects): [Hoffman, 1989].

See also ../references.

5.24.12 Related experiments

- *E124: Klauber triangle structure* (Klauber triangle structure)
- *E125: Sacks spiral structure* (Sacks spiral structure)
- *E126: Hexagonal number spiral structure* (Hexagonal number spiral structure)
- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (Quadratic prime-run atlas ($n^2 + an + b$))
- *E128: Quadratic modular obstructions (Euler-type)* (Quadratic modular obstructions (Euler-type))
- *E129: Euler lucky constants for n^2 + n + b* (Euler lucky constants for $n^2 + n + b$)

5.24.13 Parameters (example)

- size: `301` (implied grid: 301 × 301, integers `1..90,601`)

Recommended `size` range

The Ulam spiral is computed on a `size × size` grid, so runtime and memory scale roughly with size^2 (because about size^2 integers are classified as prime/not-prime).

A practical range that works well for most runs:

- **Minimum (still meaningful):** `size = 101`
- **Default / recommended:** `size = 301`
- **Comfortable upper range on typical laptops (varies):** `size = 999`
- **Above ~`size = 1501`:** expect noticeably higher runtime (and potentially memory pressure), depending on the machine and the primality implementation.

Rule of thumb: start with `301`, then try `501`, `701`, `901`. Increase further only if runtime remains acceptable.

5.24.14 Summary

We render the classic Ulam spiral for the first 90,601 positive integers and highlight primes on the spiral grid. Even at this moderate scale, faint diagonal "streaks" are visible: some diagonals contain noticeably more primes than nearby diagonals.

5.24.15 Key observations

- Several diagonals show higher prime density than the surrounding background.
- Many regions still look close to "noise-like" at this resolution; the strongest diagonal effects are easiest to see when zooming.

5.24.16 Interpretation

The diagonal structure is a real phenomenon and is commonly explained by the fact that many diagonals correspond to simple quadratic polynomials in the spiral index; some quadratics produce unusually many primes for small inputs. This is therefore a useful "pattern detector" for prime-rich formulas — but it is not evidence that primes follow a simple global rule.

5.24.17 Caveats

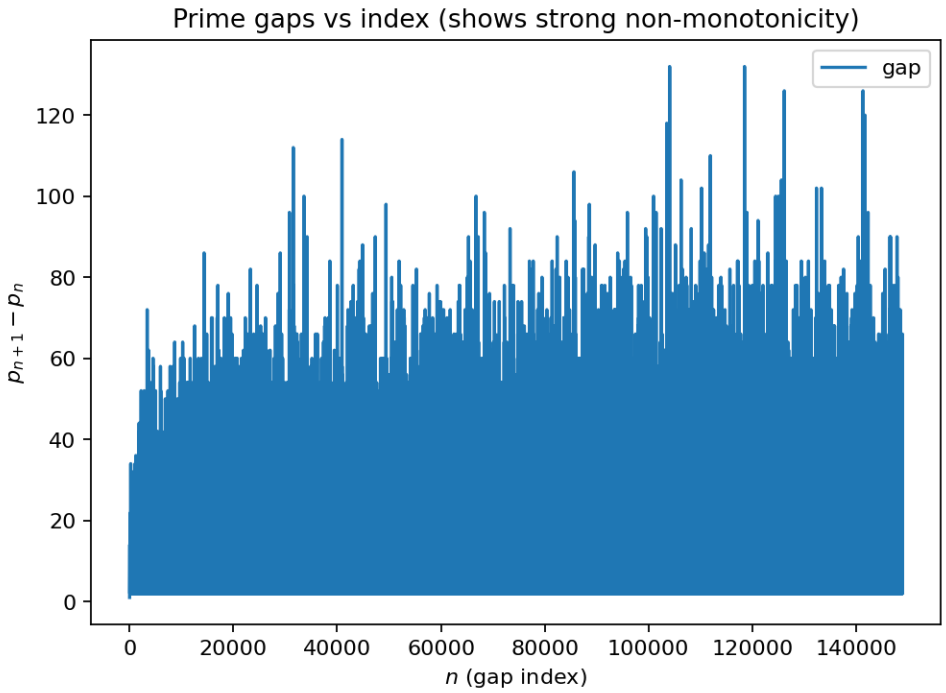
- Finite window: patterns can look stronger/weaker depending on the chosen size and zoom level.
- Visualization choices matter: marker size and interpolation can hide or exaggerate diagonal structure.
- Grid mapping details (exact spiral convention and indexing) affect which diagonals appear most prominent.

5.25 E025: Prime gaps are not monotone

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.25.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).



- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.25.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.25.3 Background (quick refresher)

- *Prime numbers refresher*

5.25.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.25.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e025/` with a parameter snapshot.

5.25.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`

- `params.json`
- `report.md`

5.25.7 How to run

```
make run EXP=e025
```

or:

```
uv run python -m mathxlab.experiments.e025
```

5.25.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.25.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.25.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e025
```

Parameters

- n_max: `2000000`

Notes

- Even though the *typical* gap near x is about log x, gaps fluctuate wildly.
- This plot is a quick antidote to monotonic thinking.

params.json (snapshot)

```
{
  "n_max": 2000000
}
```

5.25.11 References

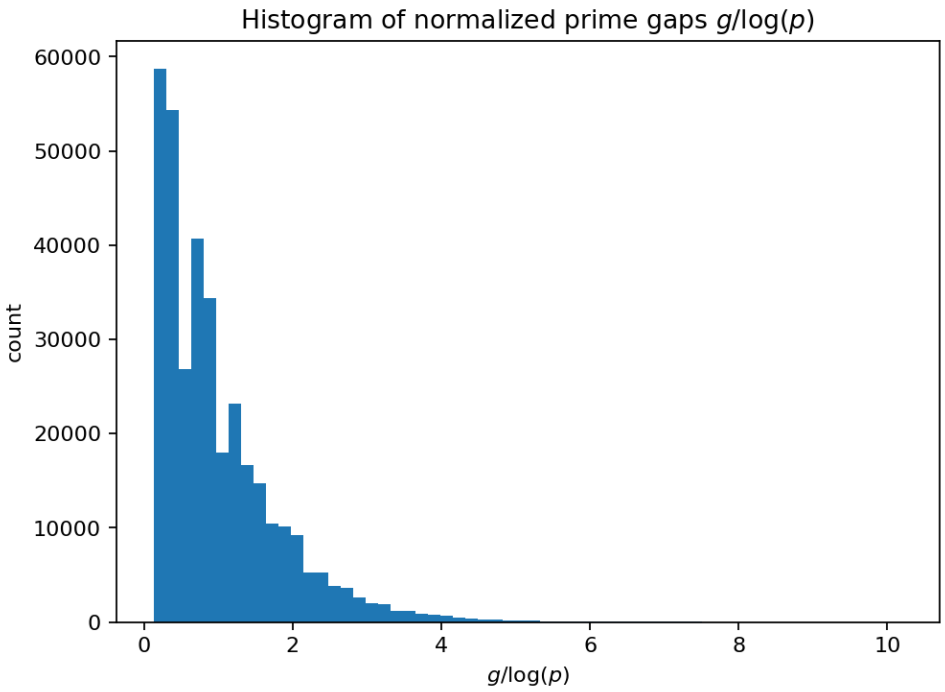
See ../references.

5.25.12 Related experiments

- *E026: Normalized prime gaps* (Normalized prime gaps)
- *E028: Jumping champions (most frequent gaps)* (Jumping champions (most frequent gaps))
- *E033: Bounded gaps vs. twin primes (not the same)* (Bounded gaps vs. twin primes (not the same))

- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.26 E026: Normalized prime gaps



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.26.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.26.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.26.3 Background (quick refresher)

- *Prime numbers refresher*

5.26.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.26.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e026/` with a parameter snapshot.

5.26.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.26.7 How to run

```
make run EXP=e026
```

or:

```
uv run python -m mathxlab.experiments.e026
```

5.26.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.26.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.26.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e026
```

Parameters

- n_max: `5000000`
- bins: `60`

Notes

- Normalization lets you compare gap statistics across different magnitudes of p.

params.json (snapshot)

```
{
  "bins": 60,
  "n_max": 5000000
}
```

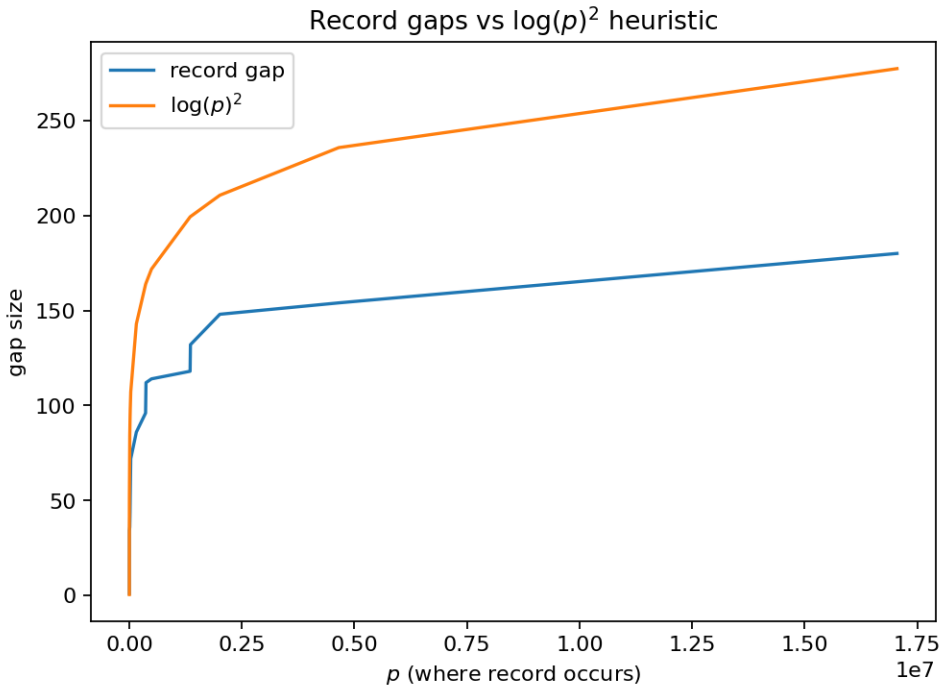
5.26.11 References

See ../references.

5.26.12 Related experiments

- *E025: Prime gaps are not monotone* (Prime gaps are not monotone)
- *E028: Jumping champions (most frequent gaps)* (Jumping champions (most frequent gaps))
- *E033: Bounded gaps vs. twin primes (not the same)* (Bounded gaps vs. twin primes (not the same))
- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.27 E027: Record prime gaps vs. log^2 heuristic



Tags: `number-theory`, `conjecture-generation`, `visualization` See: *Valid Tags*.

5.27.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.27.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.27.3 Background (quick refresher)

- *Prime numbers refresher*

5.27.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.27.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e027/` with a parameter snapshot.

5.27.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.27.7 How to run

```
make run EXP=e027
```

or:

```
uv run python -m mathxlab.experiments.e027
```

5.27.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.27.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.27.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e027
```

Parameters

- n_max: `20000000`

Notes

- Cramér-style heuristics suggest maximal gaps around x scale like $O(\log^2 x)$.
- This experiment is empirical: we compare record gaps to $\log(p)^2$ on a finite range.

params.json (snapshot)

```
{
  "n_max": 20000000
}
```

5.27.11 References

See ../references.

5.27.12 Related experiments

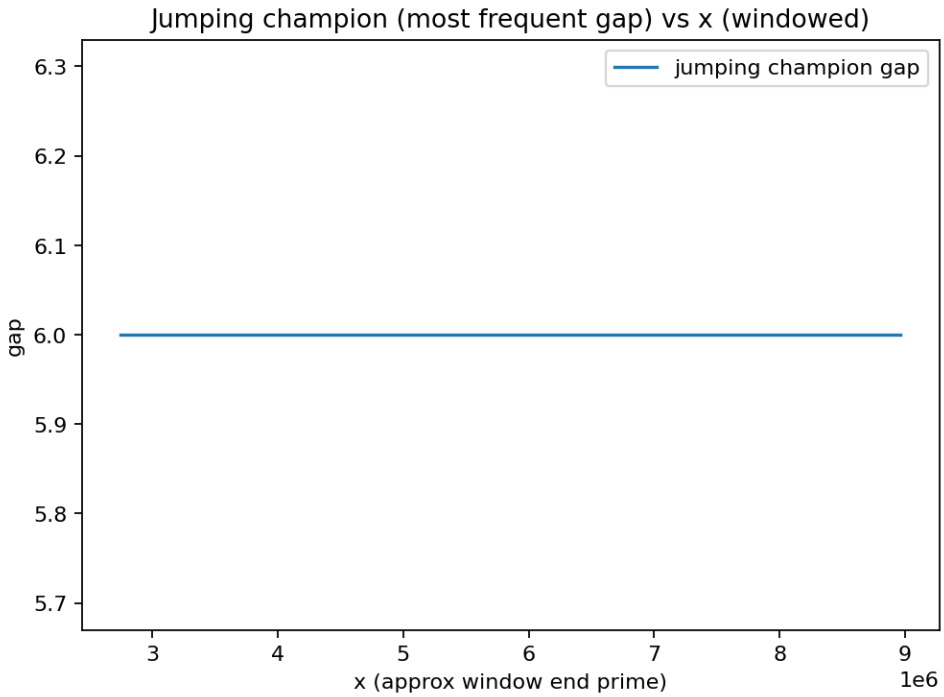
- *E029: Twin primes: observed vs. heuristic* (Twin primes: observed vs. heuristic)
- *E006: Near Misses to Perfection* (Near Misses to Perfection)
- *E011: Heuristic rarity of Mersenne primes* (Heuristic rarity of Mersenne primes)
- *E025: Prime gaps are not monotone* (Prime gaps are not monotone)
- *E026: Normalized prime gaps* (Normalized prime gaps)

5.28 E028: Jumping champions (most frequent gaps)

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.28.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.



5.28.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.28.3 Background (quick refresher)

- *Prime numbers refresher*

5.28.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.28.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e028/` with a parameter snapshot.

5.28.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.28.7 How to run

```
make run EXP=e028
```

or:

```
uv run python -m mathxlab.experiments.e028
```

5.28.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.28.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.28.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e028
```

Parameters

- n_max: `10000000`
- window: `200000` gaps
- step: `100000` gaps

Notes

- The most frequent gap tends to be a small primorial-related number (often 6 for quite a while).
- This is a fun example of 'typical behavior' that changes slowly with scale.

params.json (snapshot)


```
{
  "n_max": 10000000,
  "step": 100000,
  "window": 200000
}
```


5.28.11 References

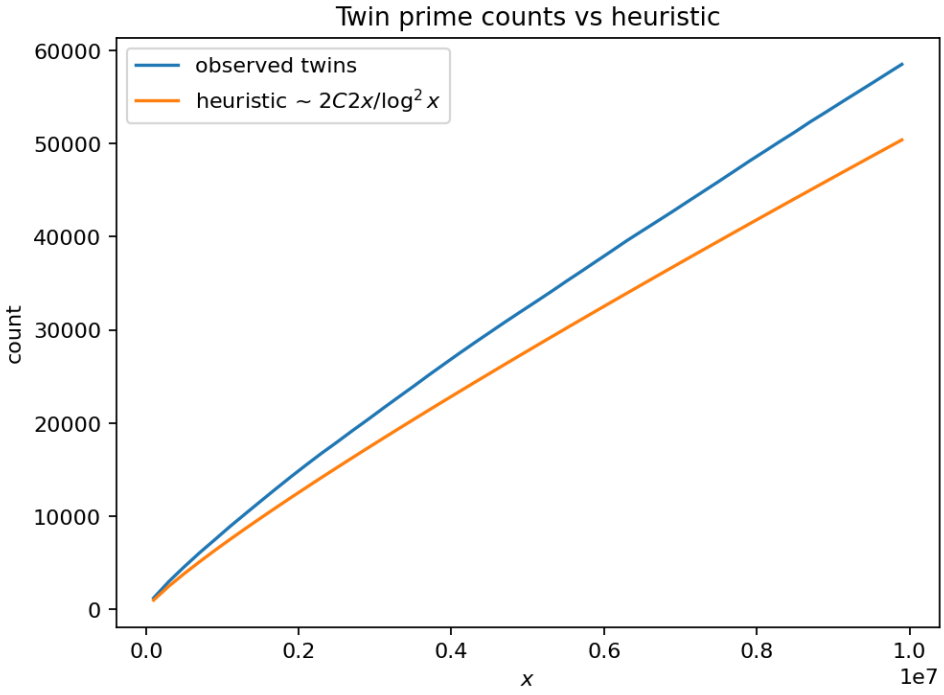
See ../references.

5.28.12 Related experiments

- *E025: Prime gaps are not monotone* (Prime gaps are not monotone)
- *E026: Normalized prime gaps* (Normalized prime gaps)

- *E033: Bounded gaps vs. twin primes (not the same)* (Bounded gaps vs. twin primes (not the same))
- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.29 E029: Twin primes: observed vs. heuristic



Tags: `number-theory`, `conjecture-generation`, `visualization` See: *Valid Tags*.

5.29.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.29.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.29.3 Background (quick refresher)

- *Prime numbers refresher*

5.29.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.29.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e029/` with a parameter snapshot.

5.29.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.29.7 How to run

```
make run EXP=e029
```

or:

```
uv run python -m mathxlab.experiments.e029
```

5.29.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.29.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.29.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e029
```

Parameters

- n_max: `10000000`

Notes

- The heuristic curve is not a theorem; it's an asymptotic guess from prime k-tuple heuristics.
- The point is to compare shapes and scaling, not to expect perfect agreement at small x.

params.json (snapshot)

```
{
  "n_max": 10000000
}
```

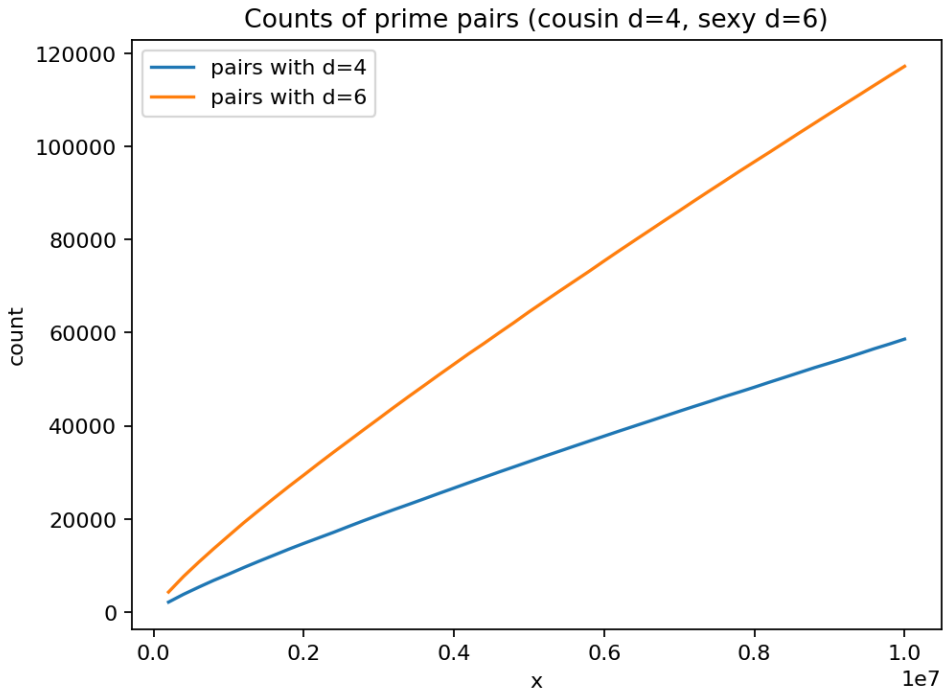
5.29.11 References

See ../references.

5.29.12 Related experiments

- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E006: Near Misses to Perfection* (Near Misses to Perfection)
- *E011: Heuristic rarity of Mersenne primes* (Heuristic rarity of Mersenne primes)
- *E033: Bounded gaps vs. twin primes (not the same)* (Bounded gaps vs. twin primes (not the same))
- *E034: Twin primes in sliding windows* (Twin primes in sliding windows)

5.30 E030: Cousin and sexy prime pairs



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.30.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.30.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.30.3 Background (quick refresher)

- *Prime numbers refresher*

5.30.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.30.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e030/` with a parameter snapshot.

5.30.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.30.7 How to run

```
make run EXP=e030
```

or:

```
uv run python -m mathxlab.experiments.e030
```

5.30.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.30.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.30.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e030
```

Parameters

- n_max: `10000000`
- d_values: `[4, 6]`

Notes

- Prime pairs with fixed even gap d are all instances of prime constellations.
- Different d values have different 'local obstructions' (mod constraints) and different constants.

params.json (snapshot)

```
{
  "d_values": [
    4,
    6
  ],
  "n_max": 10000000
}
```

5.30.11 References

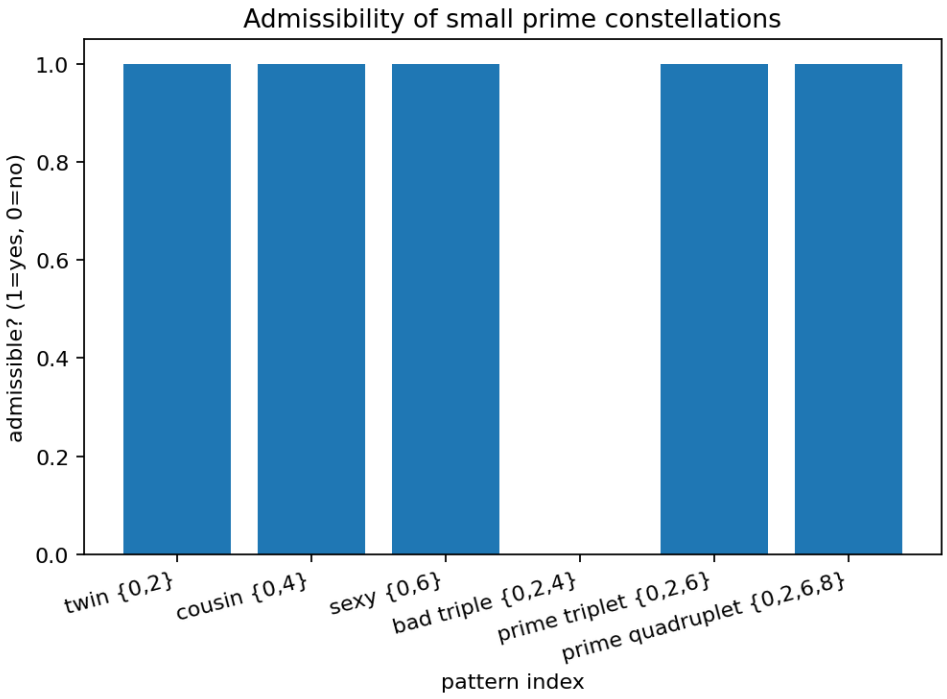
See ../references.

5.30.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.31 E031: Admissibility and modular obstructions

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.



5.31.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.31.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.31.3 Background (quick refresher)

- *Prime numbers refresher*

5.31.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.31.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e031/` with a parameter snapshot.

5.31.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.31.7 How to run

```
make run EXP=e031
```

or:

```
uv run python -m mathxlab.experiments.e031
```

5.31.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.31.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.31.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e031
```

Parameters

- max_mod: `29`

5.31.11 Results

- twin {0,2}: admissible
- cousin {0,4}: admissible
- sexy {0,6}: admissible
- bad triple {0,2,4}: NOT admissible
- prime triplet {0,2,6}: admissible
- prime quadruplet {0,2,6,8}: admissible

Notes

- A pattern must be admissible (no modulus p blocks it completely) to have any chance of occurring infinitely often.
- This is a crisp 'counterexample filter' for naive prime-pattern claims.

params.json (snapshot)


```
{
  "max_mod": 29
}
```

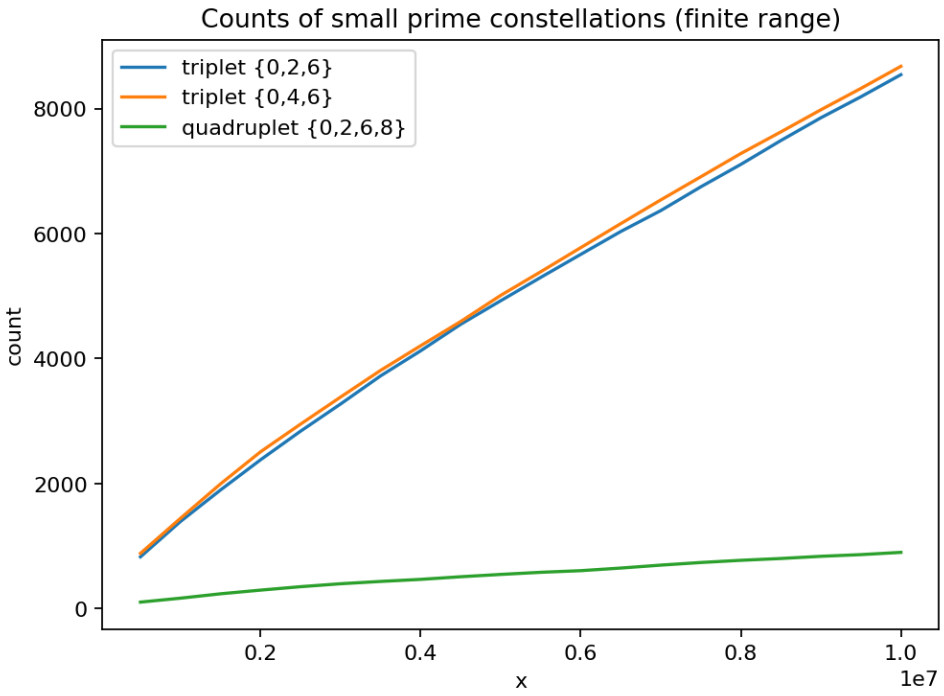

5.31.12 References

See ../references.

5.31.13 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.32 E032: Prime triplets and quadruplets



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.32.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.32.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.32.3 Background (quick refresher)

- *Prime numbers refresher*

5.32.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.32.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e032/` with a parameter snapshot.

5.32.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.32.7 How to run

```
make run EXP=e032
```

or:

```
uv run python -m mathxlab.experiments.e032
```

5.32.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.32.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.32.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e032
```

Parameters

- n_max: `10000000`

Notes

- These patterns are rare; counts grow slowly.
- The plot helps compare how quickly different constellations appear in the same range.

params.json (snapshot)

```
{
  "n_max": 10000000
}
```

5.32.11 References

See ../references.

5.32.12 Related experiments

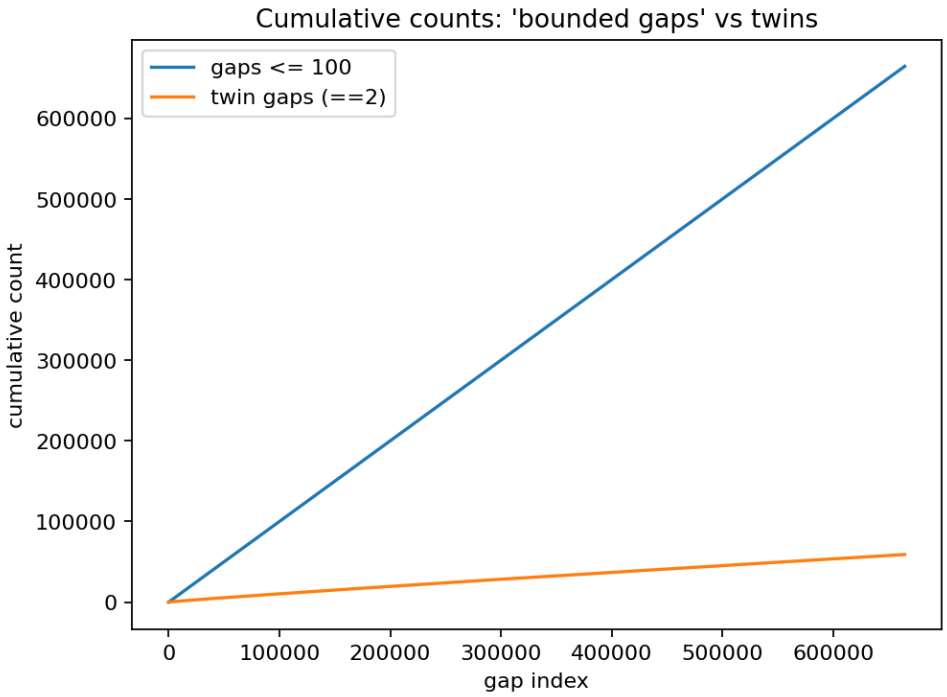
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.33 E033: Bounded gaps vs. twin primes (not the same)

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.33.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* “looks-true-until-it-breaks” behavior.



5.33.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.33.3 Background (quick refresher)

- *Prime numbers refresher*

5.33.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.33.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e033/` with a parameter snapshot.

5.33.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.33.7 How to run

```
make run EXP=e033
```

or:

```
uv run python -m mathxlab.experiments.e033
```

5.33.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.33.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.33.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e033
```

Parameters

- n_max: `10000000`
- threshold: `100`

Notes

- Seeing many small gaps does not imply the smallest gap (2) occurs infinitely often.
- This is not a proof statement, just a numerical 'intuition guardrail'.

params.json (snapshot)

```
{
  "n_max": 10000000,
  "threshold": 100
}
```

5.33.11 References

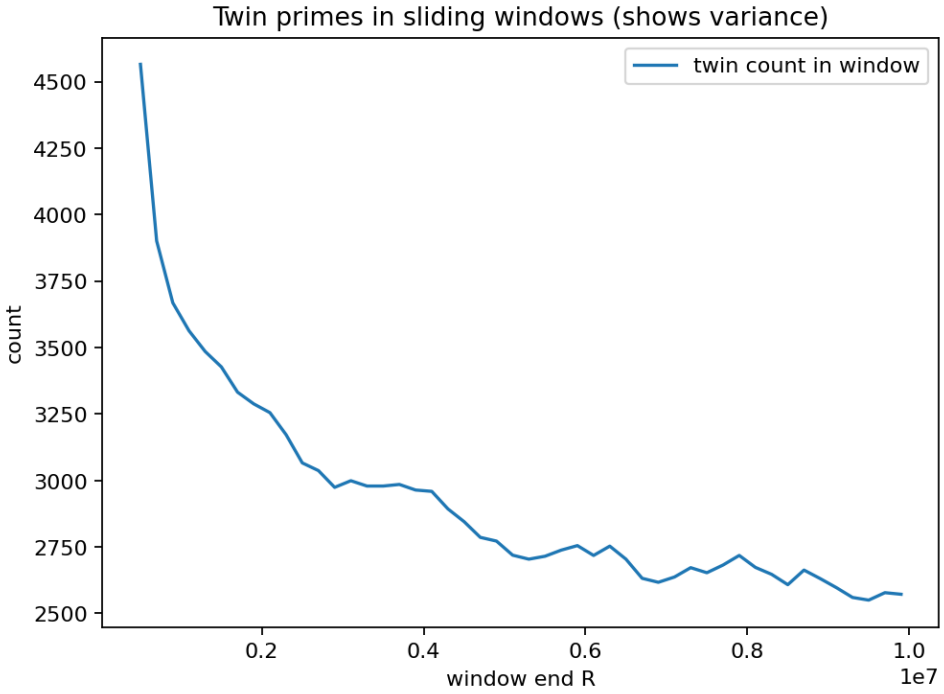
See ../references.

5.33.12 Related experiments

- *E025: Prime gaps are not monotone* (Prime gaps are not monotone)
- *E026: Normalized prime gaps* (Normalized prime gaps)
- *E028: Jumping champions (most frequent gaps)* (Jumping champions (most frequent gaps))
- *E034: Twin primes in sliding windows* (Twin primes in sliding windows)

- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)

5.34 E034: Twin primes in sliding windows



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.34.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.34.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.34.3 Background (quick refresher)

- *Prime numbers refresher*

5.34.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.34.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).

- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e034/` with a parameter snapshot.

5.34.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.34.7 How to run

```
make run EXP=e034
```

or:

```
uv run python -m mathxlab.experiments.e034
```

5.34.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.34.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.34.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e034
```

Parameters

- n_max: `10000000`
- window: `500000`
- step: `200000`

Notes

- Even if a heuristic gives an average density, local counts in windows vary a lot.
- This is a counterexample to 'smoothness' assumptions when eyeballing primes.

params.json (snapshot)


```
{
  "n_max": 10000000,
  "step": 200000,
  "window": 500000
}
```

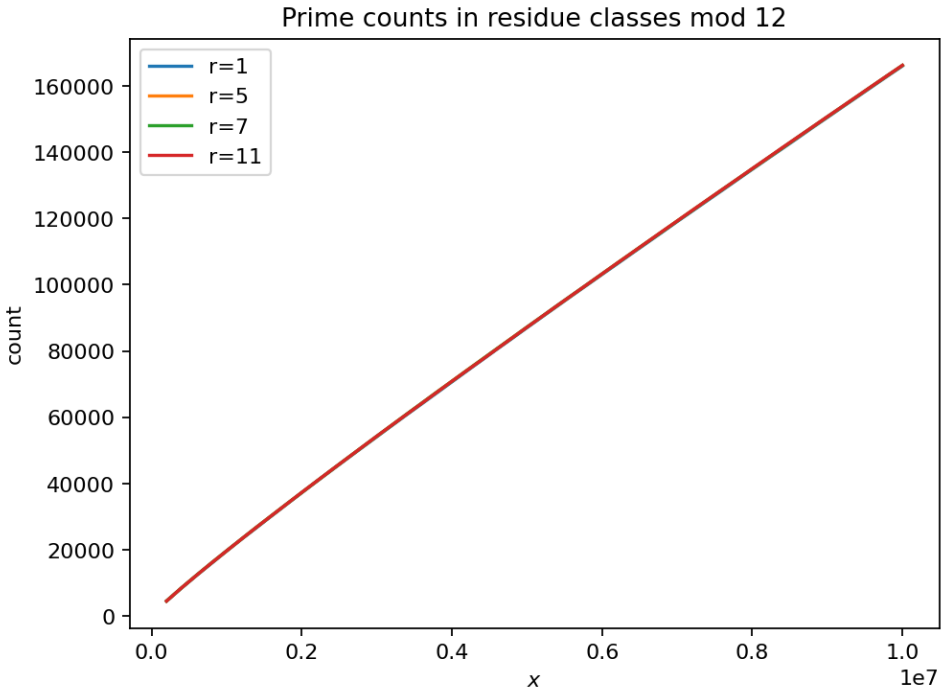

5.34.11 References

See ../references.

5.34.12 Related experiments

- *E033: Bounded gaps vs. twin primes (not the same)* (Bounded gaps vs. twin primes (not the same))
- *E029: Twin primes: observed vs. heuristic* (Twin primes: observed vs. heuristic)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))

5.35 E035: Primes in arithmetic progressions mod q



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.35.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).

- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.35.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.35.3 Background (quick refresher)

- *Prime numbers refresher*

5.35.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.35.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e035/` with a parameter snapshot.

5.35.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.35.7 How to run

```
make run EXP=e035
```

or:

```
uv run python -m mathxlab.experiments.e035
```

5.35.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.35.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.35.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e035
```

Parameters

- n_max: `10000000`
- q: `12`

Notes

- Dirichlet's theorem guarantees infinitely many primes in each reduced residue class.
- Finite ranges show biases and slow convergence to equal proportions.

params.json (snapshot)

```
{
  "n_max": 10000000,
  "q": 12
}
```

5.35.11 References

See ../references.

5.35.12 Related experiments

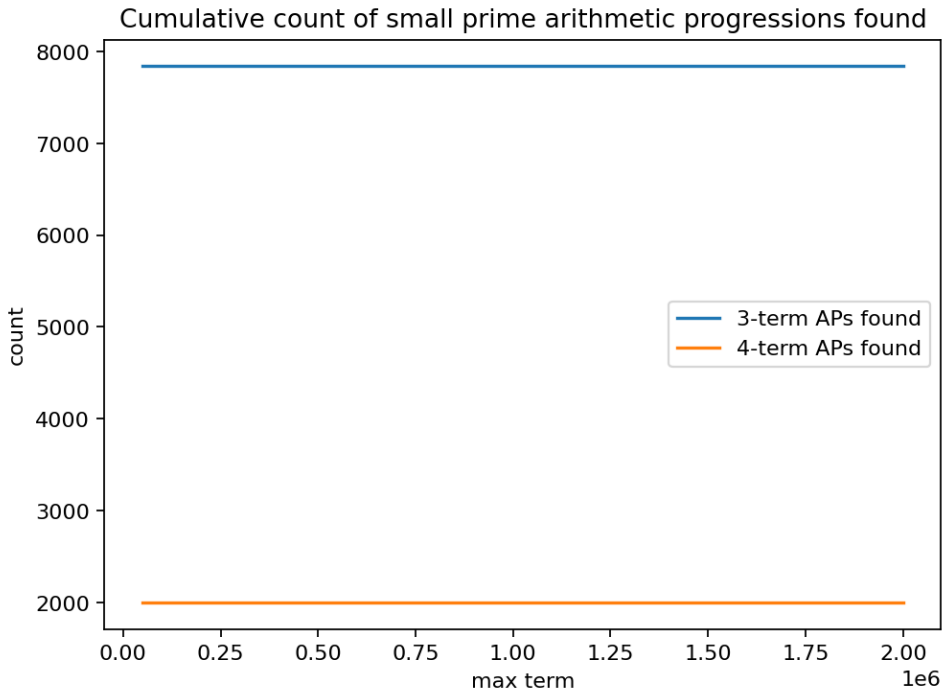
- *E036: Prime arithmetic progressions (small search)* (Prime arithmetic progressions (small search))
- *E076: Chebyshev (x;q,a): weighted prime counts in progressions.* (Chebyshev (x;q,a): weighted prime counts in progressions.)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))

5.36 E036: Prime arithmetic progressions (small search)

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.36.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.



5.36.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.36.3 Background (quick refresher)

- *Prime numbers refresher*

5.36.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.36.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e036/` with a parameter snapshot.

5.36.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.36.7 How to run

```
make run EXP=e036
```

or:

```
uv run python -m mathxlab.experiments.e036
```

5.36.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.36.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.36.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e036
```

Parameters

- n_max: `2000000`
- max_d: `5000`

Notes

- This is a finite search for short APs, not a proof of existence for arbitrary lengths.
- Even small ranges contain many 3-term APs; 4-term APs are rarer but still appear.

5.36.11 Sample progressions

- 3-term examples: `[(3, 5, 7), (3, 7, 11), (3, 11, 19), (3, 13, 23), (3, 17, 31)]`
- 4-term examples: `[(5, 11, 17, 23), (5, 17, 29, 41), (5, 23, 41, 59), (5, 47, 89, 131), (5, 53, 101, 149)]`

params.json (snapshot)


```
{
  "max_d": 5000,
  "n_max": 2000000
}
```

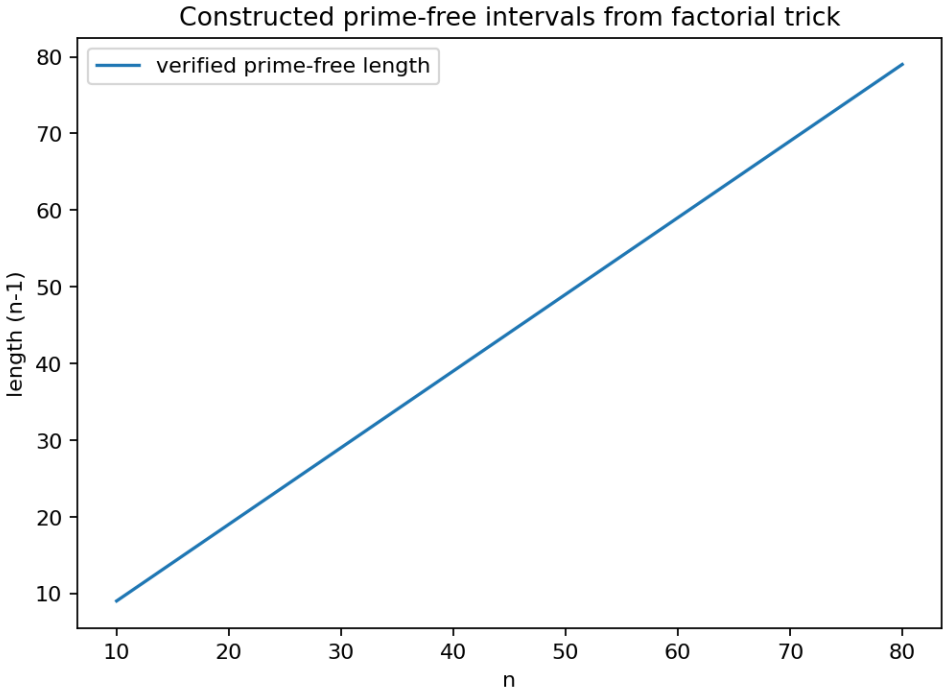

5.36.12 References

See ../references.

5.36.13 Related experiments

- *E035: Primes in arithmetic progressions mod q* (Primes in arithmetic progressions mod q)
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E042: Repunit primes (small k scan)* (Repunit primes (small k scan))
- *E076: Chebyshev (x;q,a): weighted prime counts in progressions.* (Chebyshev (x;q,a): weighted prime counts in progressions.)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.37 E037: Prime-free intervals via factorial construction



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.37.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.37.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.37.3 Background (quick refresher)

- *Prime numbers refresher*

5.37.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.37.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e037/` with a parameter snapshot.

5.37.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.37.7 How to run

```
make run EXP=e037
```

or:

```
uv run python -m mathxlab.experiments.e037
```

5.37.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.37.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.37.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e037
```

Parameters

- n_values: `[10, 20, 30, 40, 50, 60, 80]`

Notes

- For each n, the numbers n!+2, n!+3, …, n!+n are divisible by 2,3,…,n respectively.
- This provides explicit long runs of composites (a counterexample to 'primes appear regularly').

params.json (snapshot)

```
{
  "n_values": [
    10,
    20,
    30,
    40,
    50,
    60,
    80
  ]
}
```

5.37.11 References

See ../references.

5.37.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.38 E038: Bertrand's postulate (computational verification)

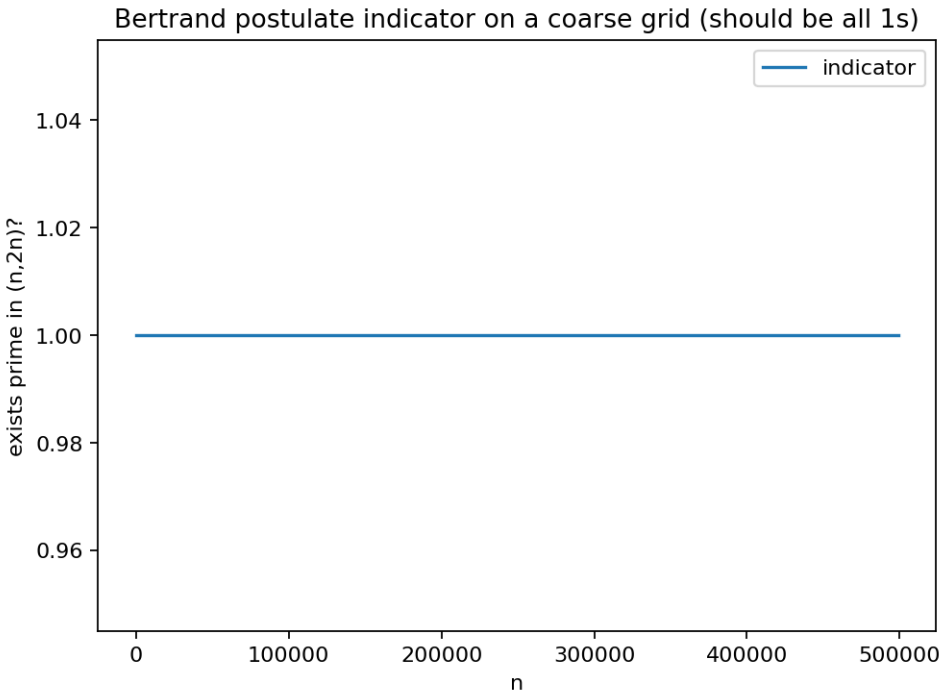
Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.38.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.38.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.



5.38.3 Background (quick refresher)

- *Prime numbers refresher*

5.38.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.38.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e038/` with a parameter snapshot.

5.38.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.38.7 How to run

```
make run EXP=e038
```

or:

```
uv run python -m mathxlab.experiments.e038
```

5.38.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.38.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.38.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e038
```

Parameters

- n_max: `500000`

Notes

- Bertrand's postulate is a theorem; this experiment is a quick computational sanity check.
- It's also a useful 'prime existence oracle' for constructing examples.

params.json (snapshot)

```
{
  "n_max": 500000
}
```

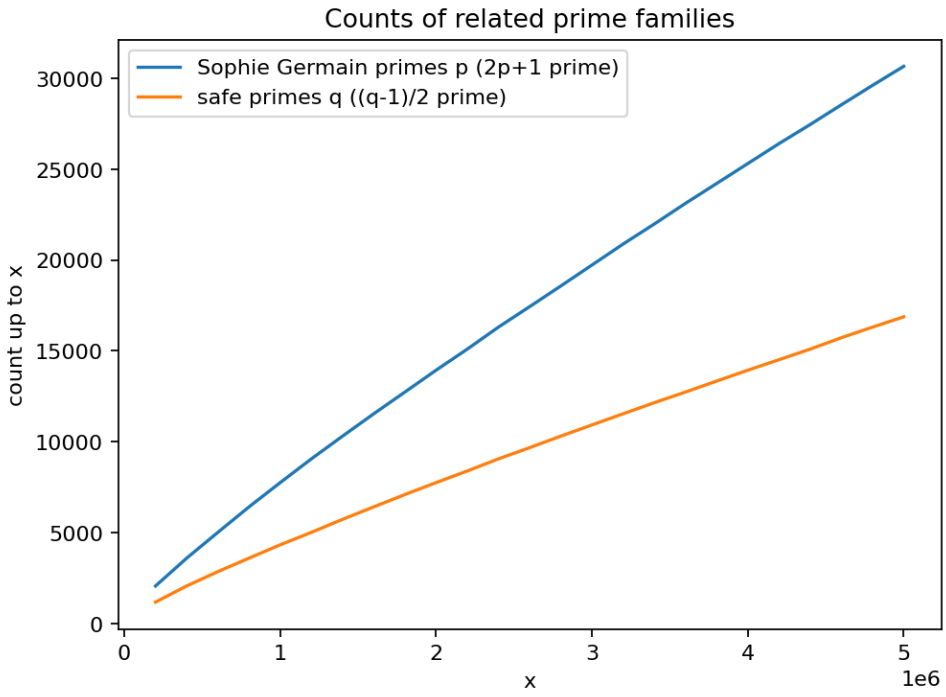
5.38.11 References

See ../references.

5.38.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.39 E039: Sophie Germain and safe primes



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.39.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.39.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.39.3 Background (quick refresher)

- *Prime numbers refresher*

5.39.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.39.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e039/` with a parameter snapshot.

5.39.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.39.7 How to run

```
make run EXP=e039
```

or:

```
uv run python -m mathxlab.experiments.e039
```

5.39.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.39.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.39.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e039
```

Parameters

- n_max: `5000000`

Notes

- These prime families matter in cryptography and in prime pattern heuristics.
- Counts grow slowly; local fluctuations are visible at moderate x.

params.json (snapshot)

```
{
  "n_max": 5000000
}
```

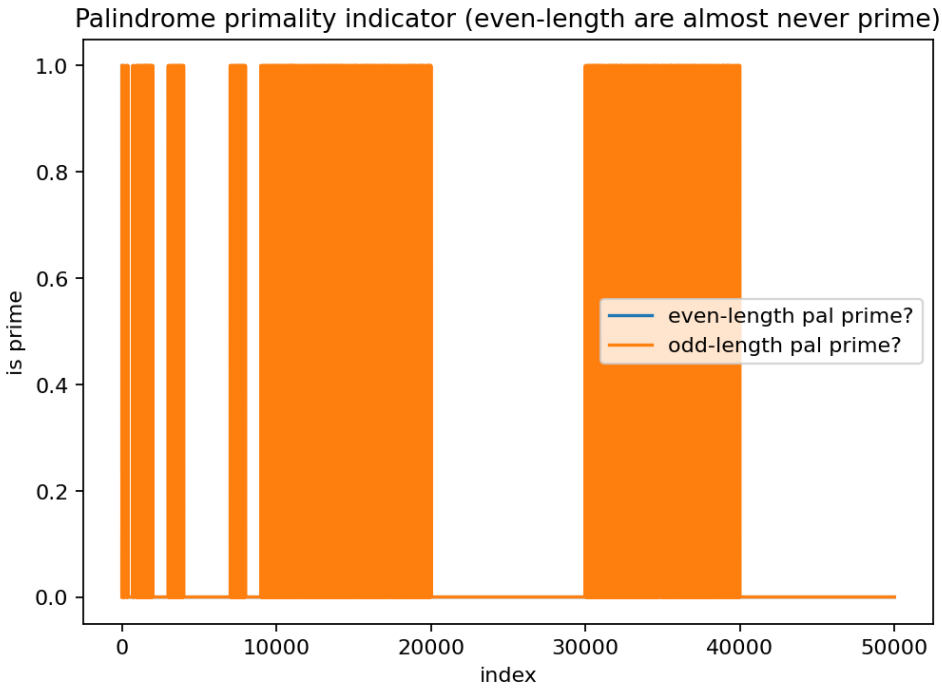
5.39.11 References

See ../references.

5.39.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.40 E040: Palindromic primes and the '11 trap'



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.40.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.40.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.40.3 Background (quick refresher)

- *Prime numbers refresher*

5.40.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.40.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e040/` with a parameter snapshot.

5.40.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.40.7 How to run

```
make run EXP=e040
```

or:

```
uv run python -m mathxlab.experiments.e040
```

5.40.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.40.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.40.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e040
```

Parameters

- digits_max: `10`
- limit: `50000`

Notes

- Every even-length base-10 palindrome is divisible by 11, hence composite (except 11 itself).
- Even-length palindromic primes found in this scan: `[11]`

params.json (snapshot)

```
{
  "digits_max": 10,
  "limit": 50000
}
```

5.40.11 References

See ../references.

5.40.12 Related experiments

- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.41 E041: Fermat numbers: not all prime

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.41.1 Highlights

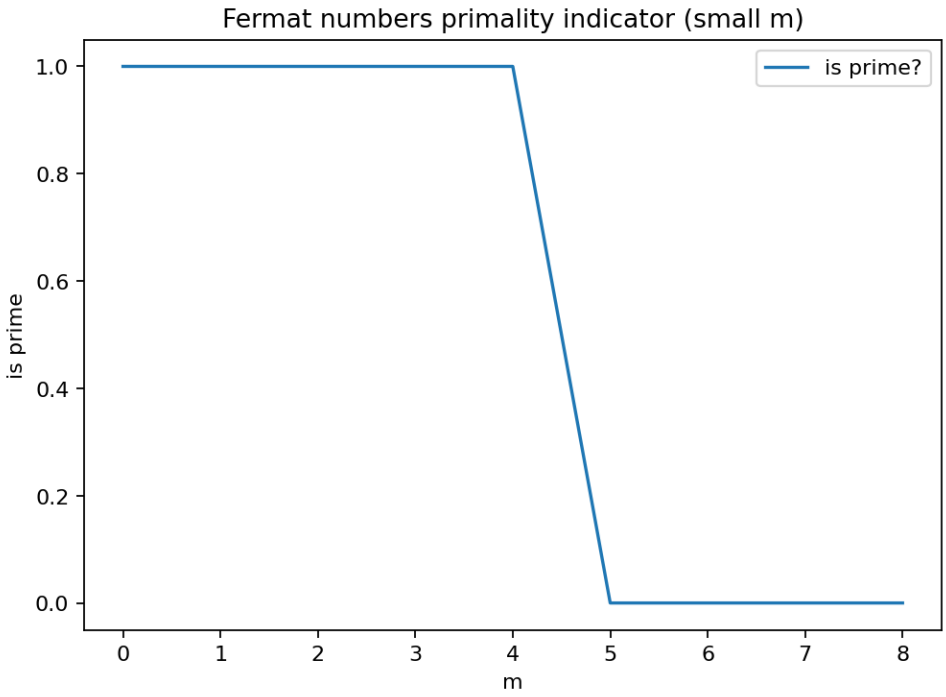
- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.41.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.41.3 Background (quick refresher)

- *Prime numbers refresher*



5.41.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.41.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e041/` with a parameter snapshot.

5.41.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.41.7 How to run

```
make run EXP=e041
```

or:

```
uv run python -m mathxlab.experiments.e041
```

5.41.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.41.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.41.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e041
```

Parameters

- m_max: 6

5.41.11 Table

m	F_m	prime?	factorization (if composite)
0	3	1	prime (in 64-bit test)
1	5	1	prime (in 64-bit test)
2	17	1	prime (in 64-bit test)
3	257	1	prime (in 64-bit test)
4	65537	1	prime (in 64-bit test)
5	4294967297	0	641 · 6700417
6	18446744073709551617	0	274177 · 67280421310721

Notes

- Fermat conjectured all F_m are prime; F5 is famously composite.
- This experiment provides a clean counterexample table and factors (via rho).

params.json (snapshot)

```
{
  "m_max": 6
}
```

5.41.12 References

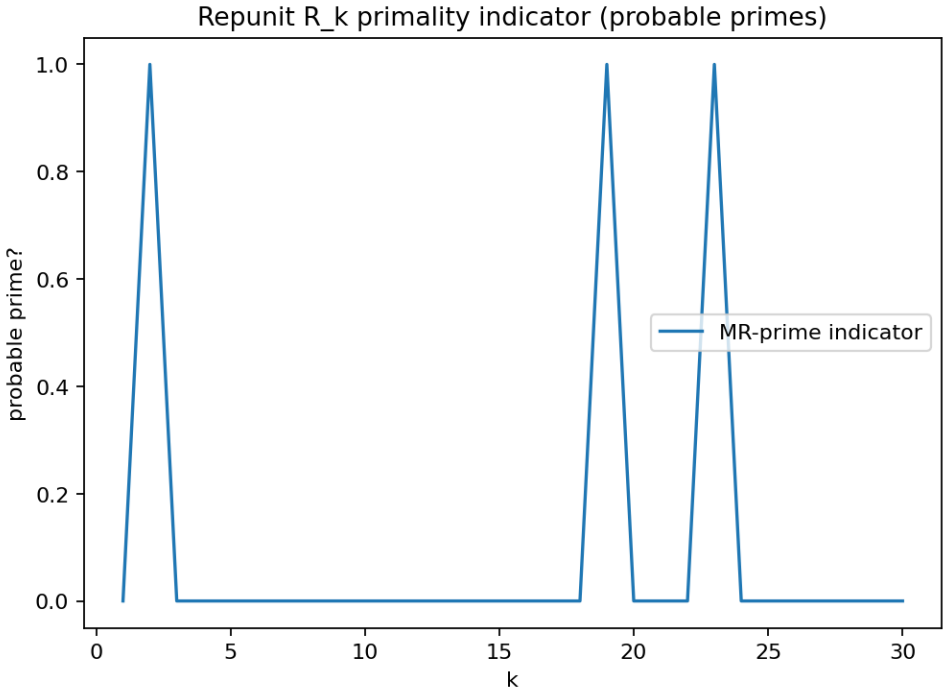
See ../references.

5.41.13 Related experiments

- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))

- *E047: Fermat numbers: Pépin test + factor witnesses* (Fermat numbers: Pépin test + factor witnesses)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)

5.42 E042: Repunit primes (small k scan)



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.42.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.42.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.42.3 Background (quick refresher)

- *Prime numbers refresher*

5.42.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.42.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e042/` with a parameter snapshot.

5.42.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.42.7 How to run

```
make run EXP=e042
```

or:

```
uv run python -m mathxlab.experiments.e042
```

5.42.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.42.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.42.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e042
```

Parameters

- k_max: `30`

Notes

- Repunit numbers grow fast; this experiment uses Miller–Rabin for a probable-prime indicator.
- Many k values produce composites; prime k is necessary (but not sufficient) for R_k to be prime.

params.json (snapshot)

```
{
  "k_max": 30
}
```

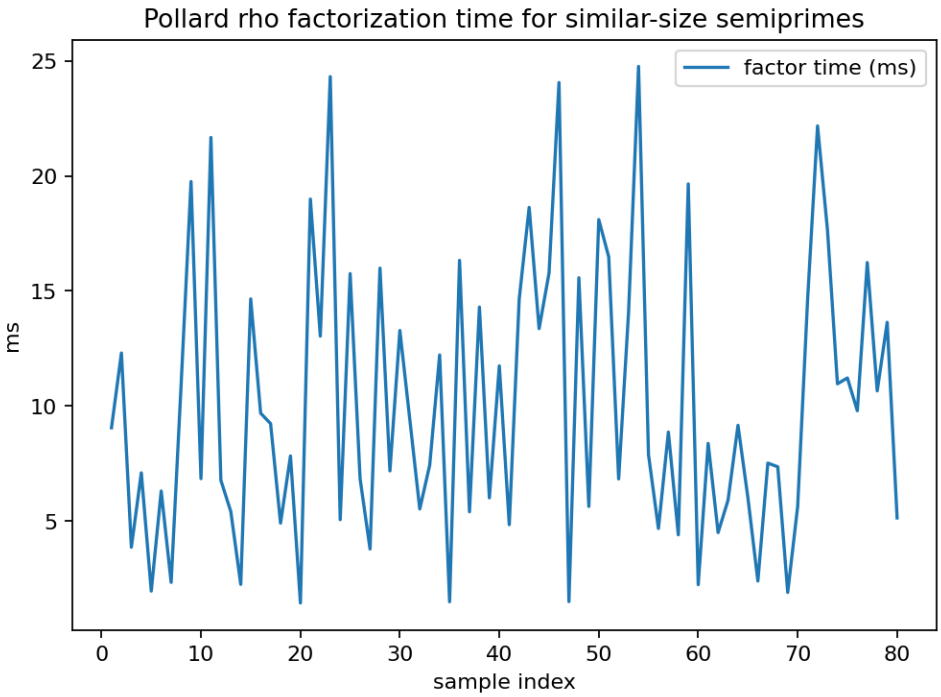
5.42.11 References

See ../references.

5.42.12 Related experiments

- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E036: Prime arithmetic progressions (small search)* (Prime arithmetic progressions (small search))
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E049: Wieferich primes (base 2): scan and quotient visualization* (Wieferich primes (base 2): scan and quotient visualization)

5.43 E043: Pollard rho runtime variability



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.43.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.43.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.43.3 Background (quick refresher)

- *Prime numbers refresher*

5.43.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.43.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e043/` with a parameter snapshot.

5.43.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.43.7 How to run

```
make run EXP=e043
```

or:

```
uv run python -m mathxlab.experiments.e043
```

5.43.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.43.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.43.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e043
```

Parameters

- samples: `80`
- bits: `28`

Notes

- Factoring difficulty depends on structure (e.g., closeness of factors), not just bit size.
- Rho is stochastic; variance is expected and is a useful experimental feature.

params.json (snapshot)


```
{
  "bits": 28,
  "samples": 80
}
```


5.43.11 References

See ../references.

5.43.12 Related experiments

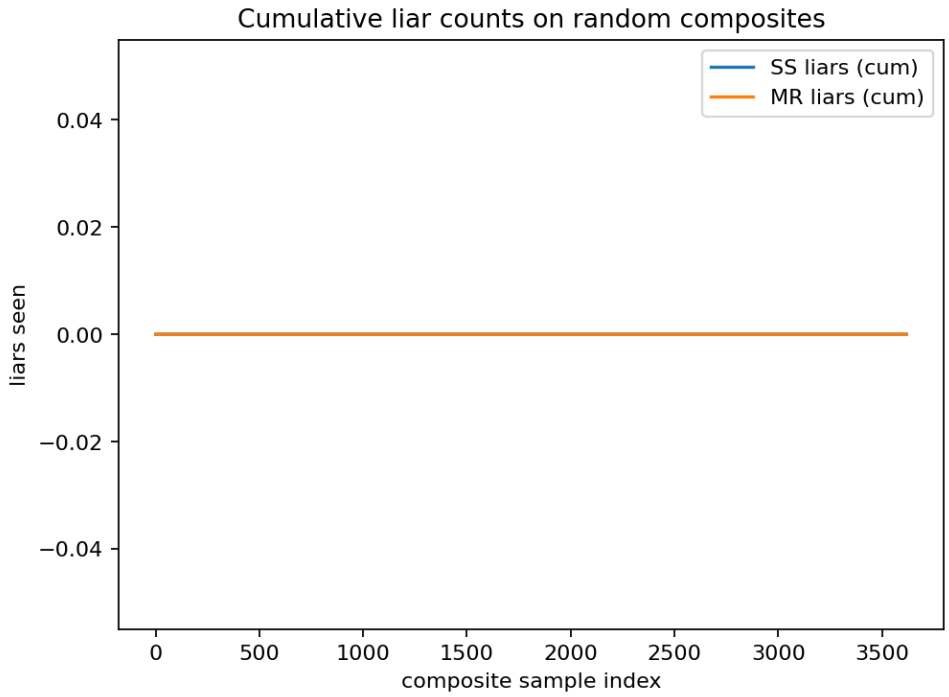
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)

5.44 E044: Solovay–Strassen vs. Miller–Rabin (liars)

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.44.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* “looks-true-until-it-breaks” behavior.



5.44.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.44.3 Background (quick refresher)

- *Prime numbers refresher*

5.44.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.44.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e044/` with a parameter snapshot.

5.44.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.44.7 How to run

```
make run EXP=e044
```

or:

```
uv run python -m mathxlab.experiments.e044
```

5.44.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.44.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.44.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e044
```

Parameters

- samples: `4000`
- bits: `32`
- bases: `[2, 3, 5, 7, 11]`

Notes

- Both tests are probabilistic; liar rates depend on bases and on the composite distribution.
- In practice, Miller–Rabin with strong base sets is often preferred.

params.json (snapshot)


```
{
  "bases": [
    2,
    3,
    5,
    7,
    11
  ],
  "bits": 32,
  "samples": 4000
}
```

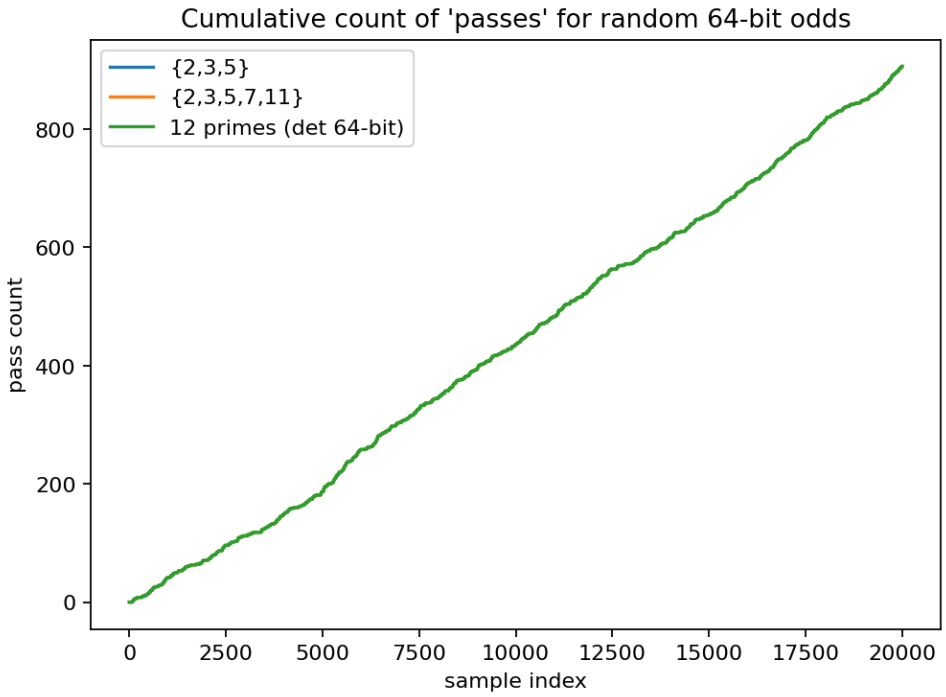
5.44.11 References

See ../references.

5.44.12 Related experiments

- *E016: Trial division vs. Miller–Rabin scaling* (Trial division vs. Miller–Rabin scaling)
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))

5.45 E045: Deterministic 64-bit MR base sets



Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.45.1 Highlights

- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.45.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.45.3 Background (quick refresher)

- *Prime numbers refresher*

5.45.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.45.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e045/` with a parameter snapshot.

5.45.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.45.7 How to run

```
make run EXP=e045
```

or:

```
uv run python -m mathxlab.experiments.e045
```

5.45.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.45.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.45.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e045
```

Parameters

- samples: `20000`
- bits: `64`

Notes

- For n < 2^64, the 12-base set (2..37 primes) is deterministic.
- Smaller base sets are faster but allow some composites through (rare, but real).

params.json (snapshot)

```
{
  "bits": 64,
  "samples": 20000
}
```

5.45.11 References

See ../references.

5.45.12 Related experiments

- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E049: Wieferich primes (base 2): scan and quotient visualization* (Wieferich primes (base 2): scan and quotient visualization)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))

5.46 E046: Prime-testing pipeline and tuning pitfalls

Tags: `number-theory`, `quantitative-exploration`, `visualization` See: *Valid Tags*.

5.46.1 Highlights

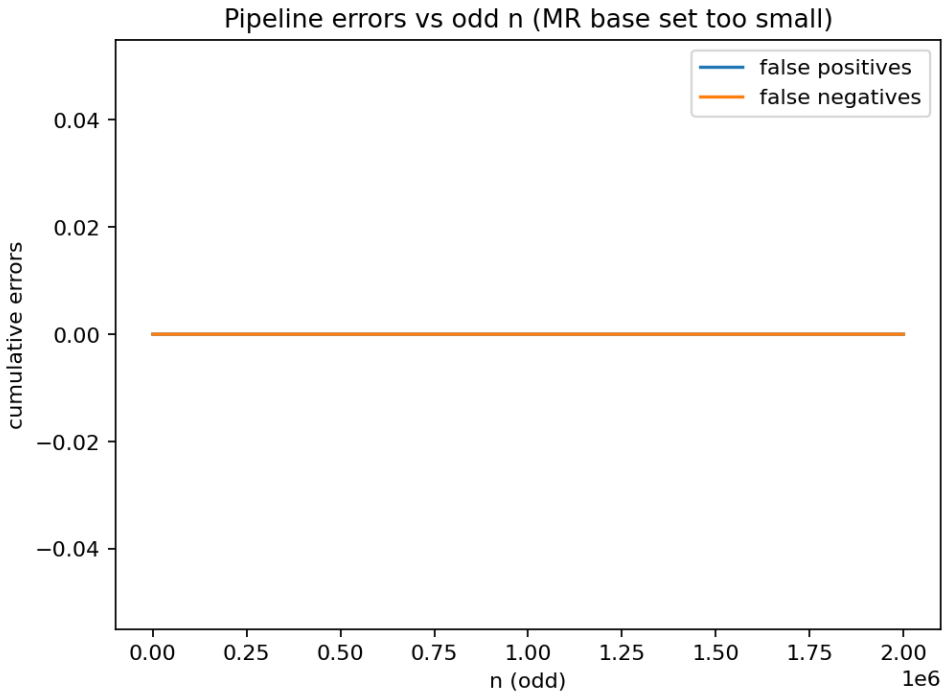
- This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.
- Writes reproducible artifacts (`params.json`, `report.md`, and figures).
- Designed to surface patterns *and* "looks-true-until-it-breaks" behavior.

5.46.2 Goal

This is a thin wrapper that follows the standard experiment template and delegates the actual computation to :mod:`mathxlab.experiments.prime_suite`.

5.46.3 Background (quick refresher)

- *Prime numbers refresher*



5.46.4 Research question

Which **prime-related claim, heuristic, or algorithm** breaks first under a clean, controlled computational sweep, and what does the smallest or clearest counterexample (or deviation) look like?

5.46.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** run a bounded search / sweep with recorded parameters.
- **Observable(s):** counts, gaps, residues, runtime scaling, or first counterexample witnesses.
- **Parameter space:** vary bounds (and sometimes algorithmic choices).
- **Outcome:** plots/tables + "witness objects" for failures.
- **Reproducibility:** outputs saved to `out/e046/` with a parameter snapshot.

5.46.6 Experiment design

- **Computation:** bounded enumeration / sampling with explicit limits.
- **Outputs:** figures and a short `report.md` summarizing what was found.
- **Artifacts written:**
- `figures/fig_*.png`
- `params.json`
- `report.md`

5.46.7 How to run

```
make run EXP=e046
```

or:

```
uv run python -m mathxlab.experiments.e046
```

5.46.8 Notes / pitfalls

- "No counterexample found" only means "none found within the configured bounds".
- For probabilistic tests (when used), treat outcomes as *evidence*, not proof.

5.46.9 Extensions

- Increase bounds and rerun (recording runtime and memory).
- Compare alternative heuristics or algorithms on the same parameter grid.
- Turn found deviations into new, tighter conjectures.

5.46.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e046
```

Parameters

- n_max: `2000000`
- mr_bases: `[2, 3, 5]`

5.46.11 First false positives (composites that passed MR stage)

none in this range

Notes

- Pipelines are great for speed, but correctness depends on parameters.
- This experiment intentionally uses too few MR bases to produce counterexamples.

params.json (snapshot)


```
{
  "mr_bases": [
    2,
    3,
    5
  ],
  "n_max": 2000000
}
```


5.46.12 References

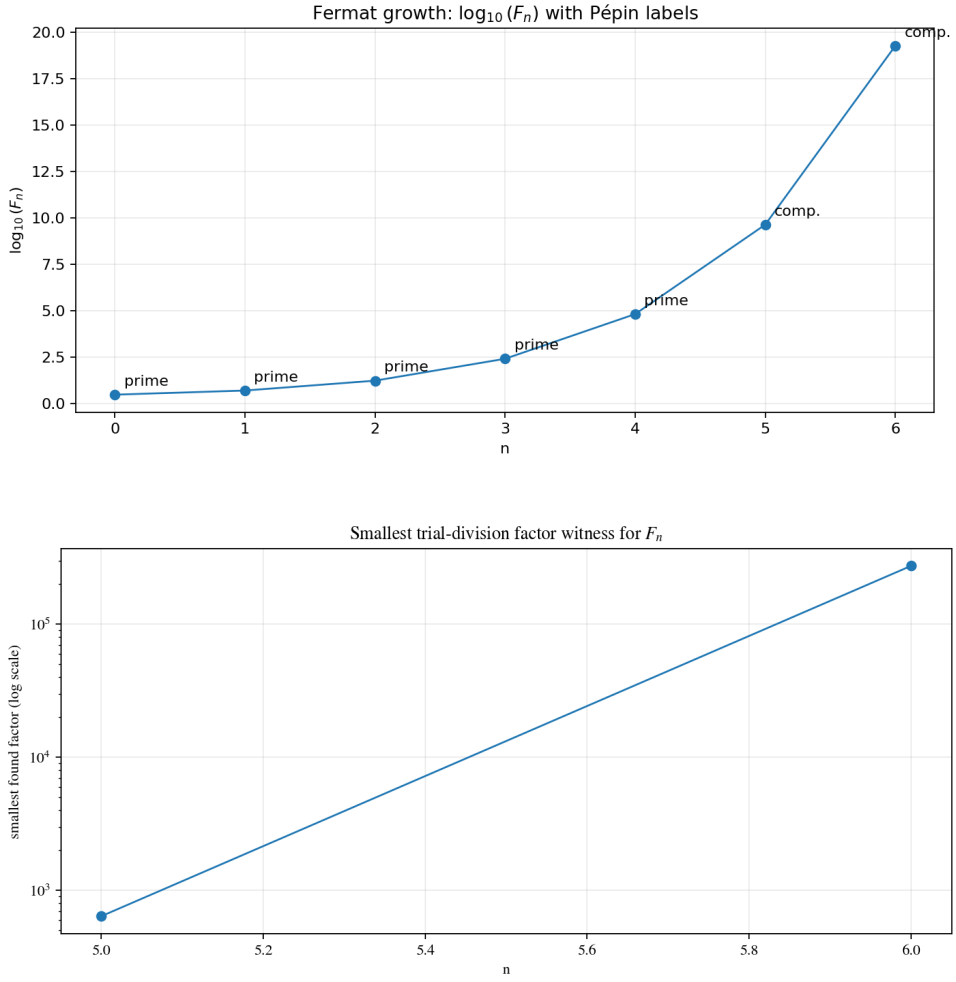
See ../references.

5.46.13 Related experiments

- *E005: Odd Perfect Numbers — Constraint Filter Pipeline* (Odd Perfect Numbers — Constraint Filter Pipeline)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)

- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))

5.47 E047: Fermat numbers: Pépin test + factor witnesses



Tags: `number-theory`, `counterexample-search`, `visualization`, `fermat` See: *Valid Tags*.

5.47.1 Highlights

- Runs Pépin's test for $F_n = 2^{2^n} + 1$ (definitive for Fermat numbers).
- Finds small, explicit factor witnesses by bounded trial division.
- Visualizes how explosively F_n grows and where the first counterexample appears.

5.47.2 Goal

Demonstrate (computationally) why the statement "all Fermat numbers are prime" fails, and produce concrete witnesses.

5.47.3 Background (quick refresher)

- *Fermat numbers*

5.47.4 Research question

How quickly do Fermat numbers become composite in practice, and how easy is it to exhibit a *witness* factor for the first failures?

5.47.5 Experiment design

- Compute F_n for $n \leq n_{max}$.
- Apply Pépin's test (for $n \geq 1$) to classify prime vs. composite.
- If composite, attempt to find a small factor by trial division up to a configurable bound.
- Plot $\log_{10}(F_n)$ and the smallest discovered factor (if any).

5.47.6 Reproducibility

- `params.json` records the run settings.
- `report.md` summarizes the key findings.
- `figures/*.png` contains the plots for the run.

5.47.7 Interpreting the results

- If Pépin says "composite", the number is definitively composite (not a probabilistic claim).
- The factor search is bounded: "no factor found" does not mean "prime".
- The classic factor for F_5 (and often for F_6) appears quickly under the default bounds.

5.47.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- n_max: 6
- factor_prime_bound: 1000000

Results (bounded)

n	F_n	Pépin says prime?	smallest factor found
0	3		—
1	5		—
2	17		—
3	257		—
4	65537		—
5	4294967297		641
6	18446744073709551617		274177

Notes

- Pépin's test is definitive for Fermat numbers (n ≥ 1).
- The factor search here is **bounded trial division**, meant to produce small, explicit witnesses (e.g., for F_5 and F_6).

params.json (snapshot)

```
{
  "factor_prime_bound": 1000000,
  "n_max": 6
}
```

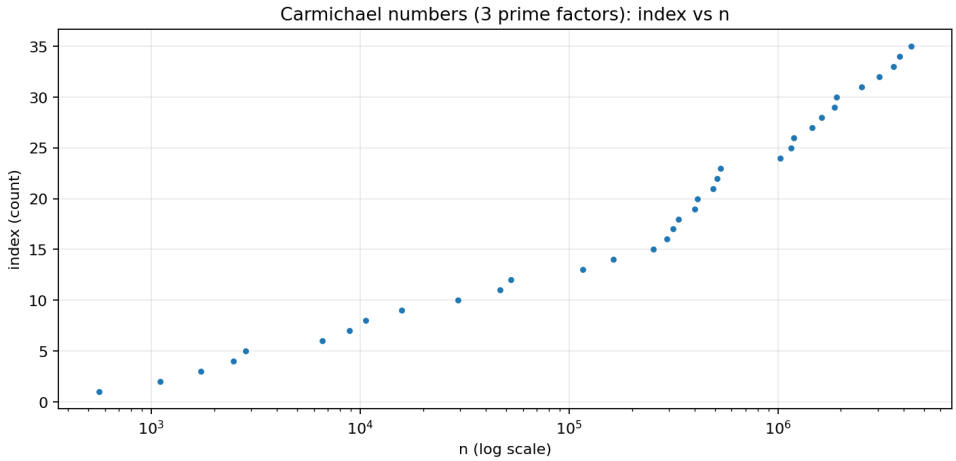
5.47.9 References

See Kř\'ižek *et al.* [2013], Wikipedia contributors [2025], The OEIS Foundation Inc. [2025].

5.47.10 Related experiments

- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E015: Wilson test infeasibility* (Wilson test infeasibility)
- *E041: Fermat numbers: not all prime* (Fermat numbers: not all prime)

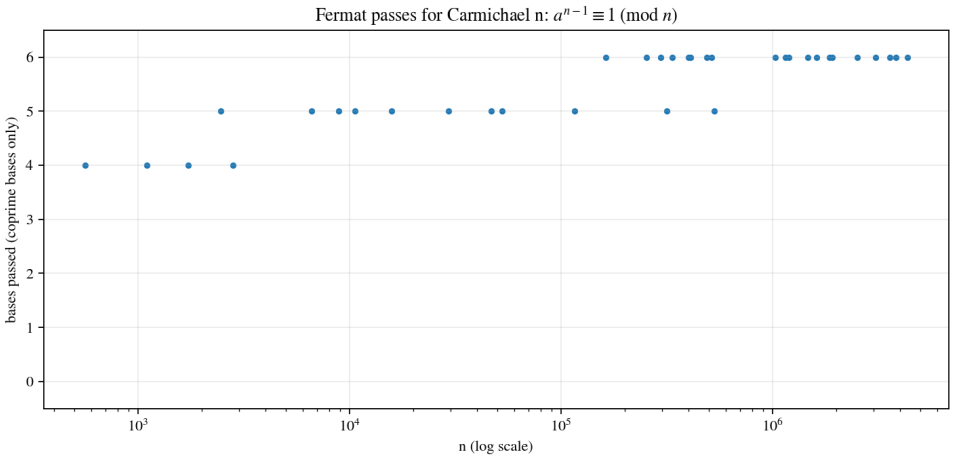
5.48 E048: Carmichael numbers: Korselt scan + Fermat counterexamples



Tags: `number-theory`, `counterexample-search`, `quantitative-exploration`, `visualization`, `carmichael`, `pseudoprime` See: *Valid Tags*.

5.48.1 Highlights

- Enumerates many small Carmichael numbers by scanning squarefree products of three primes.
- Verifies that these composites pass Fermat's test for several common bases.
- Visualizes the distribution of discovered Carmichael numbers on a log scale.



5.48.2 Goal

Show that Fermat's primality test can fail spectacularly, and produce a dataset of explicit counterexamples.

5.48.3 Background (quick refresher)

- *Carmichael numbers*
- *Prime numbers refresher*

5.48.4 Research question

Within a moderate bound N, how many Carmichael numbers can we find by a simple Korselt scan, and how many Fermat bases do they fool?

5.48.5 Experiment design

- Enumerate candidates $n = pqr \leq N$ with distinct primes $p < q < r$.
- Apply Korselt's criterion: $(p-1) \mid (n-1)$ for each prime factor $p \mid n$.
- For each Carmichael n, test Fermat congruence for a small base set (skipping non-coprime bases).
- Plot index vs. n and "bases passed" vs. n.

5.48.6 Reproducibility

- `params.json` records the run settings.
- `report.md` summarizes the key findings.
- `figures/*.png` contains the plots for the run.

5.48.7 Interpreting the results

- All numbers found are composite by construction, yet they pass Fermat congruences for *all* coprime bases (in theory).
- The scan is restricted to 3-prime-factor Carmichael numbers; larger-factor Carmichaels exist too.
- Seeing "all bases passed" in the plot is a strong reminder: Fermat alone is not a reliable primality test.

5.48.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- n_max: 5000000
- bases: 2, 3, 5, 7, 11, 13

Smallest Carmichael numbers found (3 prime factors)

n	factorization	Fermat bases passed
561	$3 \cdot 11 \cdot 17$	4
1105	$5 \cdot 13 \cdot 17$	4
1729	$7 \cdot 13 \cdot 19$	4
2465	$5 \cdot 17 \cdot 29$	5
2821	$7 \cdot 13 \cdot 31$	4
6601	$7 \cdot 23 \cdot 41$	5
8911	$7 \cdot 19 \cdot 67$	5
10585	$5 \cdot 29 \cdot 73$	5
15841	$7 \cdot 31 \cdot 73$	5
29341	$13 \cdot 37 \cdot 61$	5
46657	$13 \cdot 37 \cdot 97$	5
52633	$7 \cdot 73 \cdot 103$	5
115921	$13 \cdot 37 \cdot 241$	5
162401	$17 \cdot 41 \cdot 233$	6
252601	$41 \cdot 61 \cdot 101$	6
294409	$37 \cdot 73 \cdot 109$	6
314821	$13 \cdot 61 \cdot 397$	5
334153	$19 \cdot 43 \cdot 409$	6
399001	$31 \cdot 61 \cdot 211$	6
410041	$41 \cdot 73 \cdot 137$	6

Notes

- These n are composite yet pass Fermat's congruence for **all** coprime bases.
- The scan here is restricted to squarefree products of **three** primes, which already recovers many classical examples (e.g., $561 = 3 \cdot 11 \cdot 17$).

params.json (snapshot)


```
{
  "base_list": [
    2,
    3,
    5,
    7,
    11,
    13
  ],
  "max_results": 5000,
  "n_max": 5000000
}
```

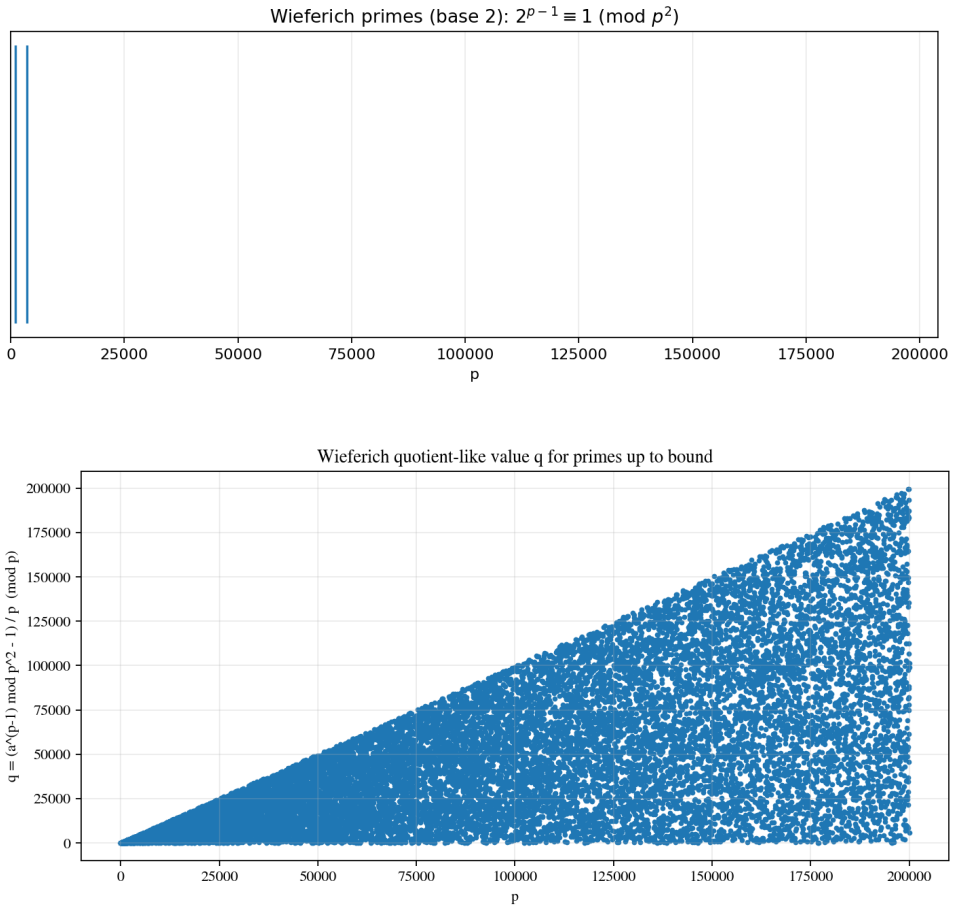
5.48.9 References

See Alford *et al.* [1994], Carmichael [1912], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

5.48.10 Related experiments

- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Prime-polynomial counterexamples (Euler's $n^2 + n + 41$))
- *E014: Primorial ± 1 counterexamples* (Primorial ± 1 counterexamples)
- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E047: Fermat numbers: Pépin test + factor witnesses* (Fermat numbers: Pépin test + factor witnesses)

5.49 E049: Wieferich primes (base 2): scan and quotient visualization



Tags: `number-theory`, `counterexample-search`, `visualization`, `wieferich` See: *Valid Tags*.

5.49.1 Highlights

- Scans primes up to a bound for the Wieferich condition $2^{p-1} \equiv 1 \pmod{p^2}$.
- Recovers the known small hits (1093 and 3511) under default settings.
- Visualizes a quotient-like value that is exactly zero for Wieferich primes.

5.49.2 Goal

Explore a rare strengthening of Fermat's congruence and visualize how exceptional Wieferich primes are.

5.49.3 Background (quick refresher)

- *Wieferich primes*
- *Prime numbers refresher*

5.49.4 Research question

Up to a bound B, which primes satisfy the Wieferich condition (base 2), and how does the quotient-like statistic vary across primes?

5.49.5 Experiment design

- Generate all primes $p \le B$ (excluding $p = 2$ for base 2).
- Compute $r = 2^{p-1} \bmod p^2$ and derive $q = (r - 1)/p \pmod{p}$.
- Mark primes with $q = 0$ as Wieferich hits.
- Plot hit positions and a scatter of q values.

5.49.6 Reproducibility

- `params.json` records the run settings.
- `report.md` summarizes the key findings.
- `figures/*.png` contains the plots for the run.

5.49.7 Interpreting the results

- Hits at 1093 and 3511 confirm the implementation.
- The quotient scatter is purely exploratory; it's a compact way to see "how far" a prime is from the Wieferich condition.
- Increasing the bound quickly becomes compute-heavy (but remains feasible for moderate bounds in pure Python).

5.49.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- p_max: 200000
- base: 2

Hits

Wieferich primes found within the scan bound:

1093, 3511

Notes

- For base 2, the smallest known Wieferich primes are 1093 and 3511.
- The scan here is purely computational and bounded.

params.json (snapshot)

```
{
  "base": 2,
  "p_max": 200000
}
```

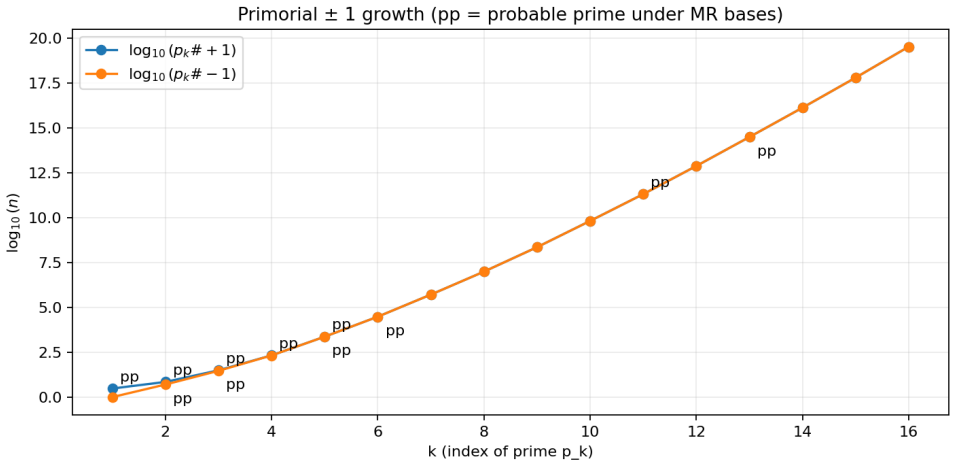
5.49.9 References

See Wieferich [1909], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

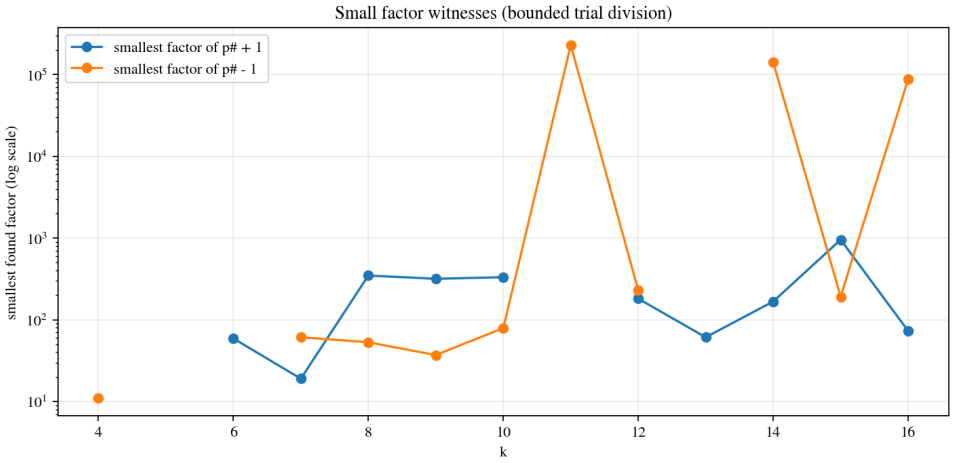
5.49.10 Related experiments

- *E018: Miller–Rabin base choice counterexamples* (Miller–Rabin base choice counterexamples)
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))
- *E009: Small-factor scan for Mersenne numbers* (Small-factor scan for Mersenne numbers)
- *E042: Repunit primes (small k scan)* (Repunit primes (small k scan))

5.50 E050: Primorials and Euclid numbers: $p\# \pm 1$ are usually composite



Tags: `number-theory`, `counterexample-search`, `quantitative-exploration`, `visualization`, `primorial` See: *Valid Tags*.



5.50.1 Highlights

- Computes primorials $p_k\# = \prod_{i \le k} p_i$ and Euclid-style numbers $p_k\# \pm 1$.
- Uses a probable-prime test to show "often composite" behavior early.
- Finds small factor witnesses via bounded trial division for many early k.

5.50.2 Goal

Demonstrate that Euclid-style numbers are coprime to small primes but are not reliably prime, and produce explicit factor witnesses.

5.50.3 Background (quick refresher)

- *Primorials*
- *Prime numbers refresher*

5.50.4 Research question

For small k, how often are $p_k\# \pm 1$ prime (or probable prime), and how easy is it to find a small factor when they are composite?

5.50.5 Experiment design

- Compute primorials for $k \le k_{max}$.
- Test $p_k\# \pm 1$ with Miller–Rabin (probable prime).
- If composite under MR, try to find a small trial-division factor as a concrete witness.
- Plot $\log_{10}(p_k\# \pm 1)$ and smallest factor witnesses (log scale).

5.50.6 Reproducibility

- `params.json` records the run settings.
- `report.md` summarizes the key findings.
- `figures/*.png` contains the plots for the run.

5.50.7 Interpreting the results

- "Probable prime" is not a proof; it's a fast filter for this exploratory experiment.
- Factor witnesses come from bounded trial division: not finding a factor is expected for some k.
- You should see early composite examples such as $p_6\# + 1 = 30031$ (with a small factor) under default bounds.

5.50.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- k_max: 16
- factor_prime_bound: 1000000
- mr_bases: 2, 3, 5, 7, 11, 13, 17

Results

k	p_k# + 1	pp?	factor	p_k# - 1	pp?	factor
1	3		—	1		—
2	7		—	5		—
3	31		—	29		—
4	211		—	209		11
5	2311		—	2309		—
6	30031		59	30029		—
7	510511		19	510509		61
8	9699691		347	9699689		53
9	223092871		317	223092869		37
10	6469693231		331	6469693229		79
11	200560490131		—	200560490129		228737
12	7420738134811		181	7420738134809		229
13	304250263527211		61	304250263527209		—
14	13082761331670031		167	13082761331670029		141269
15	614889782588491411		953	614889782588491409		191
16	32589158477190044731		73	32589158477190044729		87337

Notes

- `pp?` is *probable prime* under the chosen Miller–Rabin bases.
- Factor witnesses are from bounded trial division (not a full factorization).

params.json (snapshot)

```
{
  "factor_prime_bound": 1000000,
  "k_max": 16,
  "mr_bases": [
    2,
    3,
    5,
    7,
    11,
    13,
```

(continues on next page)

(continued from previous page)

```
    17
  ]
}
```

5.50.9 References

See Hardy and Wright [2008], The OEIS Foundation Inc. [2025], Prime Pages (UTM) [2025], Wikipedia contributors [2025].

5.50.10 Related experiments

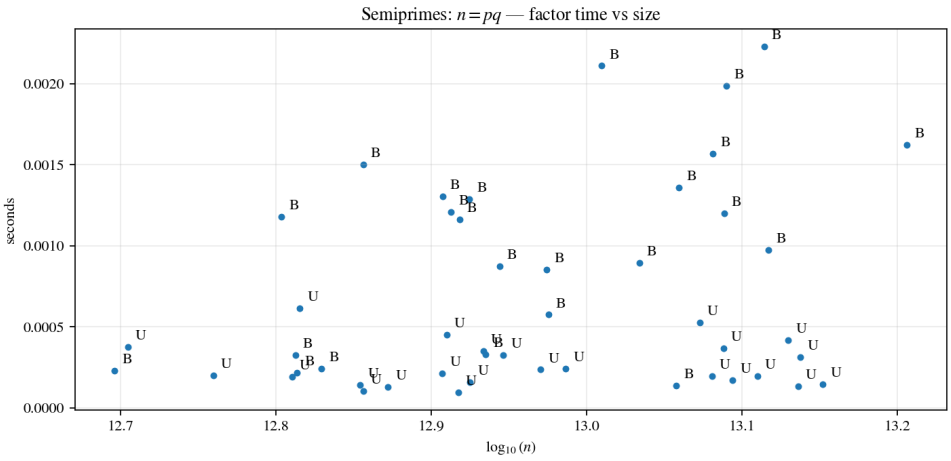
- *E048: Carmichael numbers: Korselt scan + Fermat counterexamples* (Carmichael numbers: Korselt scan + Fermat counterexamples)
- *E005: Odd Perfect Numbers — Constraint Filter Pipeline* (Odd Perfect Numbers — Constraint Filter Pipeline)
- *E012: Fermat pseudoprimes and Carmichael numbers (counterexamples)* (Fermat pseudoprimes and Carmichael numbers (counterexamples))
- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Prime-polynomial counterexamples (Euler's $n^2 + n + 41$))
- *E014: Primorial ± 1 counterexamples* (Primorial ± 1 counterexamples)

5.51 E051: Semiprimes: balanced vs. unbalanced factorization timing

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `factorization`, `semiprime`
See: *Valid Tags*.

5.51.1 Highlights

- Generates small semiprimes $n = pq$ in balanced and unbalanced regimes.
- Factors them using trial division + Pollard's rho and records timings.
- Visualizes time distributions to make "balanced is harder" tangible.



5.51.2 Goal

Compare how factorization difficulty changes when one prime factor is much smaller than the other.

5.51.3 Background (quick refresher)

- *Semiprimes*
- *Prime numbers refresher*

5.51.4 Research question

For semiprimes of similar overall size, how does Pollard-rho timing differ between balanced ($p \approx q$) and unbalanced ($p \ll q$) cases?

5.51.5 Experiment design

- Generate balanced semiprimes with roughly equal factor bit lengths.
- Generate unbalanced semiprimes with a fixed small factor size.
- Factor each using trial division pre-pass + Pollard's rho, measuring wall-clock time.
- Plot boxplots and a size-vs-time scatter.

5.51.6 Reproducibility

- `params.json` records the run settings.
- `report.md` summarizes the key findings.
- `figures/*.png` contains the plots for the run.

5.51.7 Interpreting the results

- Timings are noisy: random instance structure matters a lot at this size range.
- The trend should still show unbalanced instances often factoring faster (small factor is easier to find).
- This is an educational-scale experiment, not a cryptographic benchmark.

5.51.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- sample_count: 24
- balanced_bits: 44
- small_factor_bits: 16
- trial_division_bound: 2000

5.51.9 Summary

- balanced median time: 0.000874s
- unbalanced median time: 0.000167s

5.51.10 Samples (first 10)

category	n	p	q	time (s)
balanced	10224642949621	3203503	3191707	0.001547
balanced	12056190493199	3746843	3217693	0.001153
balanced	10809763013969	3214223	3363103	0.000652
balanced	8179752869357	3696163	2213039	0.000886
balanced	9425664304583	3074779	3065477	0.000645
balanced	8405168684519	2187287	3842737	0.000957
balanced	4970172113219	2333939	2129521	0.000170
balanced	11418890464403	2893439	3946477	0.000101
balanced	6511871939171	2301829	2828999	0.000162
balanced	6363658603031	2811649	2263319	0.000877
unbalanced	8129387573543	56681	143423503	0.000348
unbalanced	12416116674211	58771	211262641	0.000130
unbalanced	8269233337793	33721	245225033	0.000070
unbalanced	7449408569203	45083	165237641	0.000094
unbalanced	6463784298977	44623	144853199	0.000140
unbalanced	5067033575867	33457	151449131	0.000276
unbalanced	9340248965111	49157	190008523	0.000177
unbalanced	7151136875947	45823	156059989	0.000106
unbalanced	8838375880759	51169	172729111	0.000247
unbalanced	12038405385443	61651	195266993	0.000152

params.json (snapshot)


```
{
  "balanced_bits": 44,
  "mr_bases": [
    2,
    3,
    5,
    7,
    11,
    13,
    17
  ],
  "sample_count": 24,
```


(continues on next page)

(continued from previous page)

```
  "small_factor_bits": 16,
  "trial_division_bound": 2000
}
```

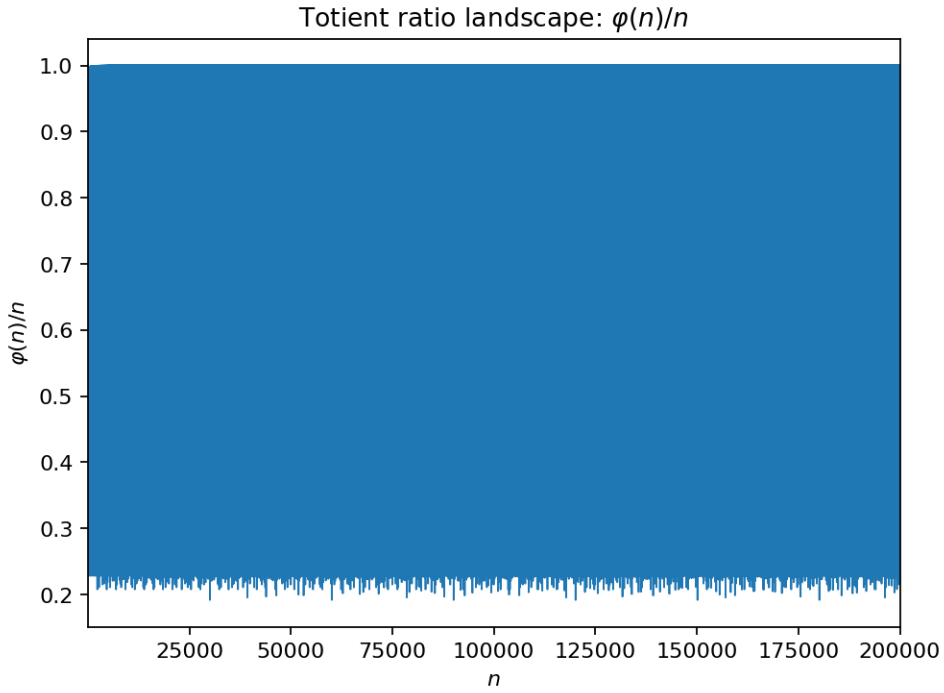
5.51.11 References

See Rivest *et al.* [1978], The OEIS Foundation Inc. [2025].

5.51.12 Related experiments

- *E004: Computing \sigma(n) at Scale — Sieve vs. Factorization* (Computing $\sigma(n)$ at Scale — Sieve vs. Factorization)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E007: Mersenne growth (bits and digits)* (Mersenne growth (bits and digits))
- *E008: Lucas–Lehmer scan (prime exponents)* (Lucas–Lehmer scan (prime exponents))

5.52 E052: Totient ratio landscape



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `totient`, `multiplicative` See: *Valid Tags*.

5.52.1 Highlights

- Plot $\varphi(n)/n$ for a range of n and annotate record lows.
- Relate deep dips to integers with many small prime factors (primorial-like structure).

Goal

Make the multiplicative structure of the totient function visible, especially the role of small primes.

Background (quick refresher)

- *Arithmetic functions refresher*

- *Euler's totient function \varphi(n) refresher*
- *Primorials*

Research question

Which n minimize $\varphi(n)/n$ up to a bound, and how do they relate to primorial products?

Method

- Compute $\varphi(n)$ for $n \le N$ (sieve / SPF-based).

- Plot the ratio $\varphi(n)/n$ and track record minima.
- Optionally compare record minima to primorials and nearby multiples.

How to run

- `make run EXP=e052`
- `uv run python -m mathxlab.experiments.e052`

Outputs

This experiment follows the standard output contract:

- `out/e052/figures/` — generated figures (PNG)
- `out/e052/report.md` — short narrative report
- `out/e052/manifest.json` — snapshot metadata for the gallery

5.52.2 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 200000
- min (n)/n in range: 0.191808 at n=30030

Figures:

- fig_01_phi_over_n.png
- fig_02_record_lows.png

params.json (snapshot)

```
{
  "n_max": 200000
}
```

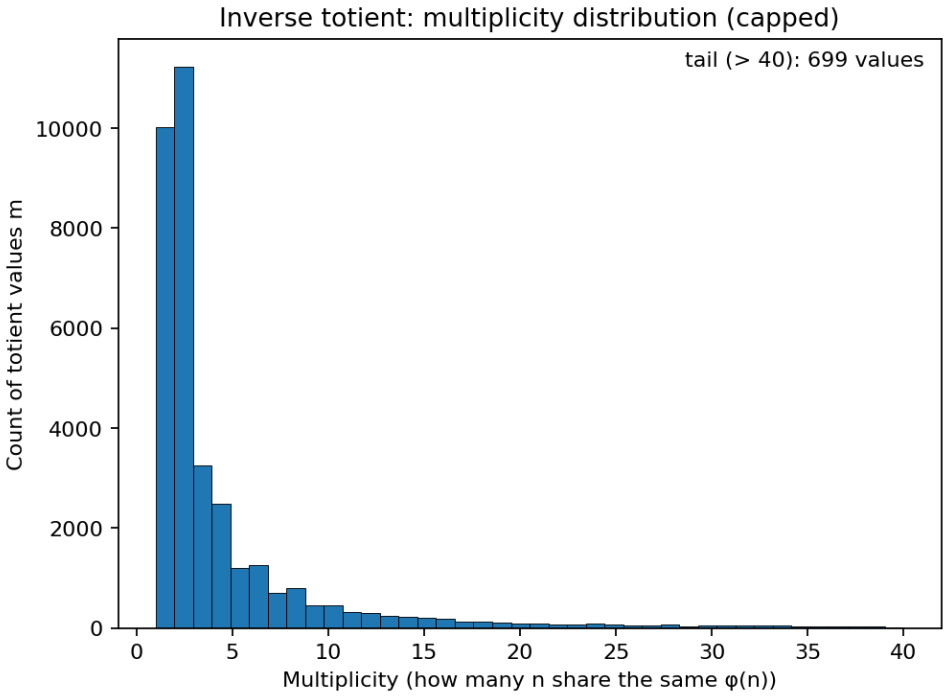
5.52.3 References

See Apostol [1976], Tenenbaum [2015], Niven *et al.* [1991].

5.52.4 Related experiments

- *E060: Jordan totients* (Jordan totients)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)
- *E102: Dirichlet convolution identity zoo* (E102: Dirichlet convolution identity zoo)
- *E062: Carmichael (n) vs. (n)* (Carmichael (n) vs. (n))

5.53 E053: Inverse totient multiplicities



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `totient`, `search` See: *Valid Tags*.

5.53.1 Highlights

- Compute $\varphi(n)$ for $n \le N$ and count how many preimages each value has.
- Plot the histogram of multiplicities and list the top collisions.

5.53.2 Goal

Empirically study the many-to-one nature of the totient function.

5.53.3 Background (quick refresher)

- *Euler's totient function \varphi(n) refresher*
- *Arithmetic functions refresher*

5.53.4 Research question

For $n \leq N$, how large can the multiplicity of a totient value get, and what do the maximizers look like?

5.53.5 Method

- Compute $\varphi(n)$ on a range and build a frequency table for values.
- Visualize multiplicities (histogram) and inspect the most frequent values.

5.53.6 How to run

- `make run EXP=e053`
- `uv run python -m mathxlab.experiments.e053`

5.53.7 Outputs

This experiment follows the standard output contract:

- `out/e053/figures/` — generated figures (PNG)
- `out/e053/report.md` — short narrative report
- `out/e053/manifest.json` — snapshot metadata for the gallery

5.53.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 200000

Top 10 totient values by multiplicity (count, m):

- 357 x : m = 40320
- 332 x : m = 51840
- 331 x : m = 34560
- 320 x : m = 60480
- 307 x : m = 43200
- 291 x : m = 48384
- 281 x : m = 30240
- 280 x : m = 47520
- 280 x : m = 46080
- 276 x : m = 25920

Figure:

- fig_01_multiplicity_hist.png

params.json (snapshot)

```
{
  "hist_max": 40,
  "n_max": 200000
}
```

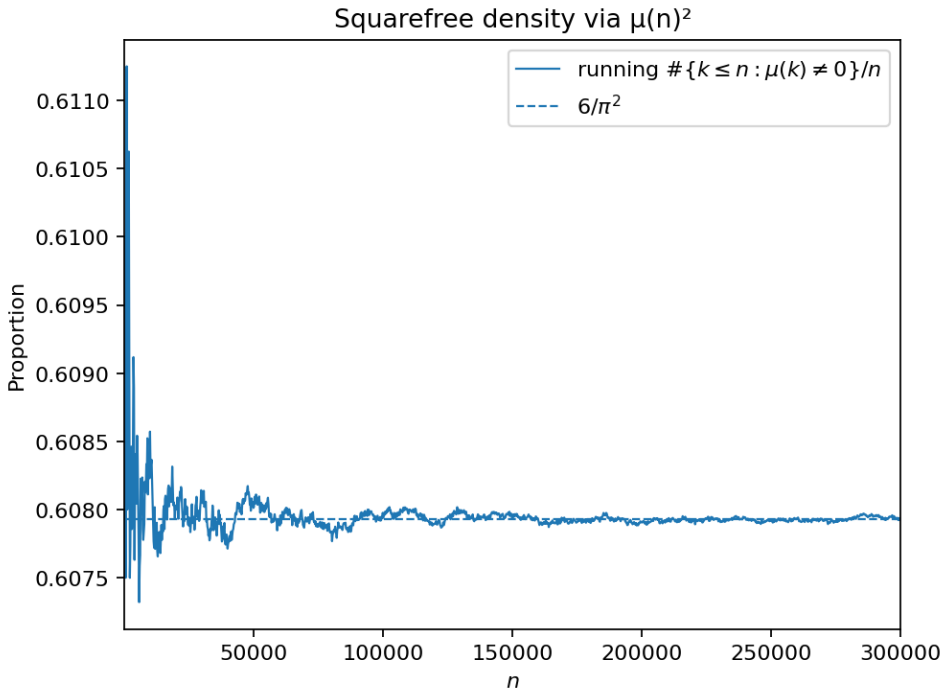
5.53.9 References

See Apostol [1976], Niven *et al.* [1991].

5.53.10 Related experiments

- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)
- *E005: Odd Perfect Numbers — Constraint Filter Pipeline* (Odd Perfect Numbers — Constraint Filter Pipeline)
- *E060: Jordan totients* (Jordan totients)
- *E062: Carmichael (n) vs. (n)* (Carmichael (n) vs. (n))

5.54 E054: Squarefree density via Möbius



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `mobius`, `summatory` See: *Valid Tags*.

5.54.1 Highlights

- Use $\mu(n)^2$ as an indicator for squarefree integers.
- Plot the running density and compare to $6/\pi^2$.

5.54.2 Goal

Show how an arithmetic function encodes a classic density result.

5.54.3 Background (quick refresher)

- *Möbius function \mu(n) and Mertens function M(x) refresher*
- *Arithmetic functions refresher*

5.54.4 Research question

How quickly does the empirical density of squarefree numbers approach $6/\pi^2$?

5.54.5 Method

- Compute $\mu(n)$ up to N and accumulate $\sum_{n \le x} \mu(n)^2$.
- Plot the ratio $\frac{1}{x} \sum_{n \le x} \mu(n)^2$ with a reference line at $6/\pi^2$.

5.54.6 How to run

- `make run EXP=e054`
- `uv run python -m mathxlab.experiments.e054`

5.54.7 Outputs

This experiment follows the standard output contract:

- `out/e054/figures/` — generated figures (PNG)
- `out/e054/report.md` — short narrative report
- `out/e054/manifest.json` — snapshot metadata for the gallery

5.54.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 300000
- observed squarefree proportion: 0.607927
- theoretical density 6/pi^2: 0.607927

Figure:

- fig_01_squarefree_density.png

params.json (snapshot)

```
{
  "n_max": 300000
}
```

5.54.9 References

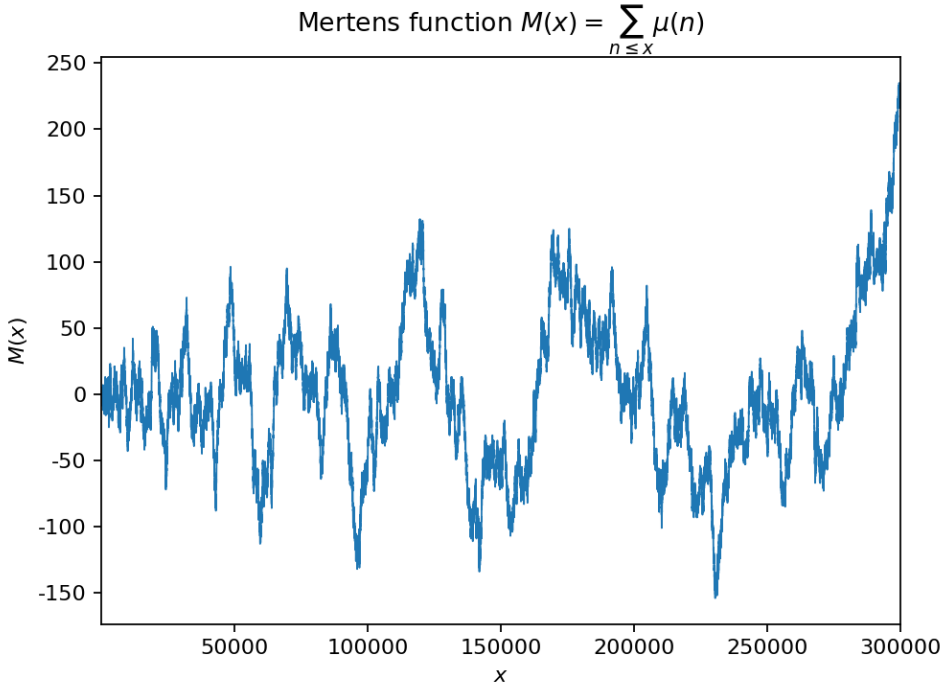
See Apostol [1976], Tenenbaum [2015].

5.54.10 Related experiments

- *E056: Liouville vs. Möbius walks* (Liouville vs. Möbius walks)
- *E055: Mertens function walk* (Mertens function walk)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)
- *E105: Mertens M(x): scaling views* (E105: Mertens M(x): scaling views)

- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)

5.55 E055: Mertens function walk



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `mobius`, `summatory`, `model-checking` See: *Valid Tags*.

5.55.1 Highlights

- Plot $M(x)$ and scaled variants such as $M(x)/\sqrt{x}$.
- Compare qualitative behavior to a simple random-walk heuristic (without overclaiming).

5.55.2 Goal

Build intuition for summatory Möbius behavior and typical fluctuation scales.

5.55.3 Background (quick refresher)

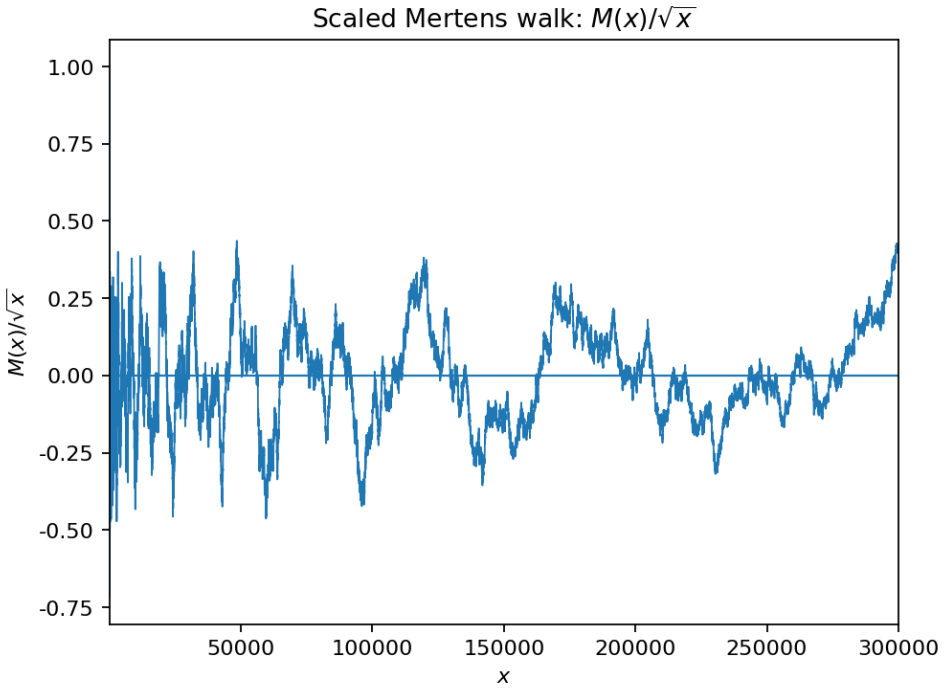
- *Möbius function \mu(n) and Mertens function M(x) refresher*
- *Average orders and the Erdős–Kac viewpoint*

5.55.4 Research question

What does $M(x)/\sqrt{x}$ look like numerically over moderate ranges?

5.55.5 Method

- Compute $\mu(n)$ up to N and cumulative sums $M(x)$.
- Plot $M(x)$ and $M(x)/\sqrt{x}$; optionally add moving-window statistics.



5.55.6 How to run

- `make run EXP=e055`
- `uv run python -m mathxlab.experiments.e055`

5.55.7 Outputs

This experiment follows the standard output contract:

- `out/e055/figures/` — generated figures (PNG)
- `out/e055/report.md` — short narrative report
- `out/e055/manifest.json` — snapshot metadata for the gallery

5.55.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 300000
- max |M(x)| in range: 235

Figures:

- fig_01_mertens.png
- fig_02_mertens_scaled.png

params.json (snapshot)

```
{
  "n_max": 300000
}
```

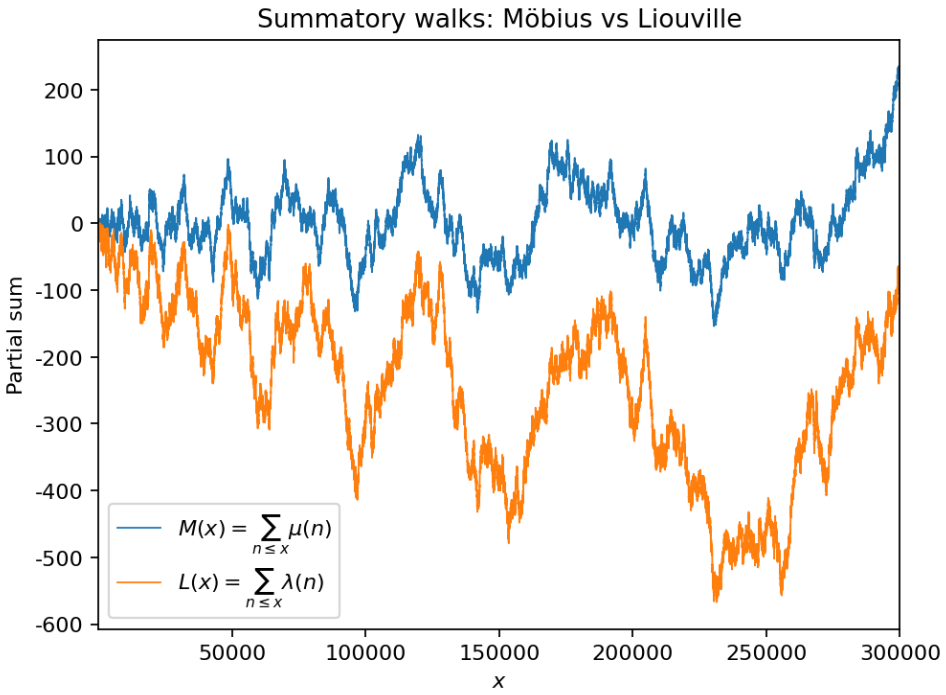
5.55.9 References

See Tenenbaum [2015], Odlyzko and te Riele [1985].

5.55.10 Related experiments

- *E105: Mertens M(x): scaling views* (E105: Mertens M(x): scaling views)
- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E056: Liouville vs. Möbius walks* (Liouville vs. Möbius walks)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)
- *E121: Möbius inversion as convolution undo* (E121: Möbius inversion as convolution undo)

5.56 E056: Liouville vs. Möbius walks



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `summatory`, `liouville`, `mobius` See: *Valid Tags*.

5.56.1 Highlights

- Plot partial sums of $\lambda(n)$ and $\mu(n)$ side by side.
- Compare scaled versions to see how the inclusion/exclusion of squareful terms changes the walk.

5.56.2 Goal

Contrast two closely related $\pm 1/0$ sequences arising from prime factorizations.

5.56.3 Background (quick refresher)

- *Liouville function \lambda(n) refresher*
- *Möbius function \mu(n) and Mertens function M(x) refresher*
- *Prime-factor counting: \omega(n) and \Omega(n) refresher*

5.56.4 Research question

How do the fluctuations of $\sum_{n \le x} \lambda(n)$ compare to $\sum_{n \le x} \mu(n)$?

5.56.5 Method

- Compute $\Omega(n)$, then $\lambda(n) = (-1)^{\Omega(n)}$; compute $\mu(n)$.
- Plot partial sums and scaled partial sums (e.g., divide by $\sqrt{x}$).

5.56.6 How to run

- `make run EXP=e056`
- `uv run python -m mathxlab.experiments.e056`

5.56.7 Outputs

This experiment follows the standard output contract:

- `out/e056/figures/` — generated figures (PNG)
- `out/e056/report.md` — short narrative report
- `out/e056/manifest.json` — snapshot metadata for the gallery

5.56.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 300000
- M(n_max) = 220
- L(n_max) = -98

Figure:

- fig_01_walks.png

params.json (snapshot)

```
{
  "n_max": 300000
}
```

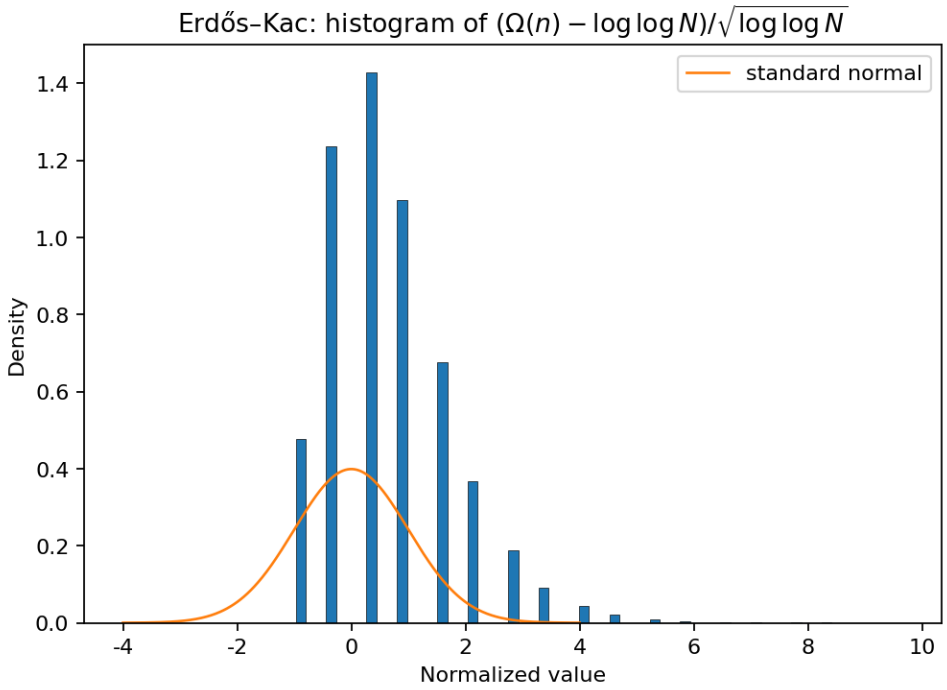
5.56.9 References

See Tenenbaum [2015].

5.56.10 Related experiments

- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)
- *E055: Mertens function walk* (Mertens function walk)
- *E105: Mertens M(x): scaling views* (E105: Mertens M(x): scaling views)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)

5.57 E057: Erdős–Kac in practice



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `omega`, `heuristics` See: *Valid Tags*.

5.57.1 Highlights

- Compute $\Omega(n)$ for $n \le N$ and plot its histogram.
- Optionally normalize to compare with a Gaussian curve (Erdős–Kac heuristic).

5.57.2 Goal

Visualize the probabilistic number-theory flavor of prime-factor counts.

5.57.3 Background (quick refresher)

- *Prime-factor counting: \omega(n) and \Omega(n) refresher*
- *Average orders and the Erdős–Kac viewpoint*

5.57.4 Research question

How close does the distribution of $\Omega(n)$ (after normalization) look to a normal curve for moderate N?

5.57.5 Method

- Compute $\Omega(n)$ using an SPF sieve.
- Plot histogram; optionally overlay a normal density with matching mean/variance approximations.

5.57.6 How to run

- `make run EXP=e057`
- `uv run python -m mathxlab.experiments.e057`

5.57.7 Outputs

This experiment follows the standard output contract:

- `out/e057/figures/` — generated figures (PNG)
- `out/e057/report.md` — short narrative report
- `out/e057/manifest.json` — snapshot metadata for the gallery

5.57.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 400000
- bins: 60

Figure:

- fig_01_erdos_kac_hist.png

params.json (snapshot)


```
{
  "bins": 60,
  "n_max": 400000
}
```


5.57.9 References

See Erdős and Kac [1940], Tenenbaum [2015].

5.57.10 Related experiments

- *E094: (n) vs. Ω(n): Erdős–Kac normalization* (E094: (n) vs. Ω(n): Erdős–Kac normalization)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)
- *E118: Chebyshev bias: lead-time statistics* (E118: Chebyshev bias: lead-time statistics)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)
- *E052: Totient ratio landscape* (Totient ratio landscape)

5.58 E058: Divisor-count record highs

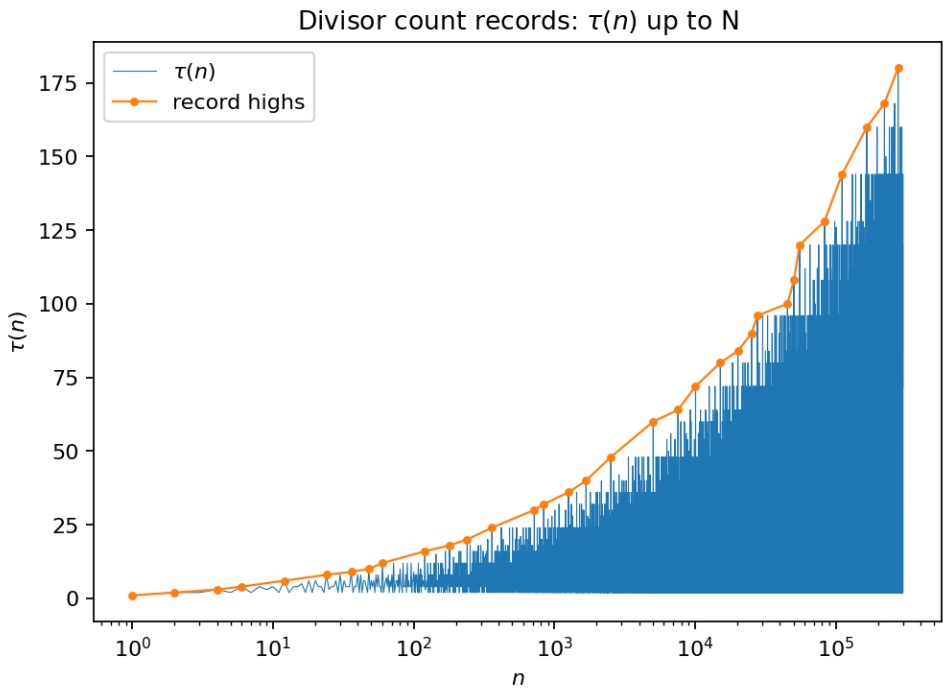
Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `divisor-function`, `bounds` See: *Valid Tags*.

5.58.1 Highlights

- Track record highs of $\tau(n)$ as n grows.
- Inspect factorizations of record-holders (many small primes with decreasing exponents).

5.58.2 Goal

Make the record phenomenon behind highly composite numbers tangible.



5.58.3 Background (quick refresher)

- *Divisor functions d(n) and \sigma_k(n) refresher*
- *Arithmetic functions refresher*

5.58.4 Research question

How do record values of $\tau(n)$ grow, and what structure do record-holders share?

5.58.5 Method

- Compute $\tau(n)$ up to N and track record positions.
- Plot record curve; list factorizations of top records.

5.58.6 How to run

- `make run EXP=e058`
- `uv run python -m mathxlab.experiments.e058`

5.58.7 Outputs

This experiment follows the standard output contract:

- `out/e058/figures/` — generated figures (PNG)
- `out/e058/report.md` — short narrative report
- `out/e058/manifest.json` — snapshot metadata for the gallery

5.58.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 300000
- max (n) in range: 180 at n=277200

Figure:

- fig_01_tau_records.png

params.json (snapshot)


```
{
  "n_max": 300000
}
```

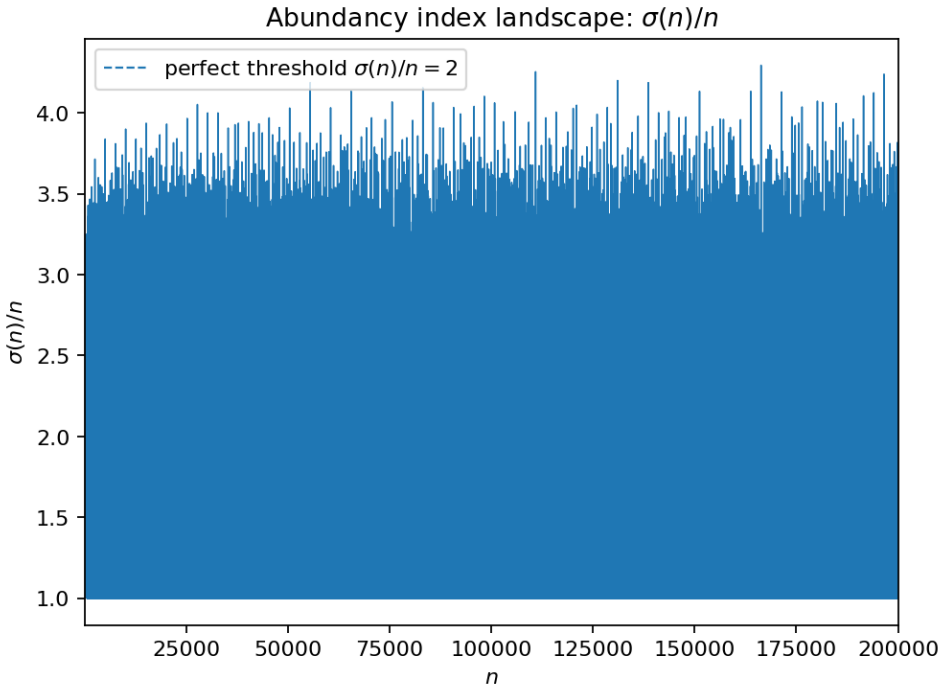

5.58.9 References

See Ramanujan [1915], Tenenbaum [2015].

5.58.10 Related experiments

- *E096: Record-holders for (n)* (E096: Record-holders for (n))
- *E059: Abundancy index landscape* (Abundancy index landscape)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)
- *E117: Prime-counting approximations: li(x) and friends* (E117: Prime-counting approximations: li(x) and friends)
- *E123: (x;q,a) vs. a simple baseline* (E123: (x;q,a) vs. a simple baseline)

5.59 E059: Abundancy index landscape



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `sigma`, `divisor-function`, `perfect` See: *Valid Tags*.

5.59.1 Highlights

- Plot $\sigma(n)/n$ and mark the threshold 2 (perfect numbers).
- Highlight spikes caused by many small prime factors.

5.59.2 Goal

Connect a divisor-sum function to the perfect/abundant classification visually.

5.59.3 Background (quick refresher)

- *Divisor functions d(n) and \sigma_k(n) refresher*
- *Perfect numbers refresher*

5.59.4 Research question

What shapes and spikes appear in $\sigma(n)/n$ over moderate ranges, and where do perfect numbers sit?

5.59.5 Method

- Compute $\sigma(n)$ up to N using factorization via SPF.
- Plot $\sigma(n)/n$ and annotate known perfect numbers in range.

5.59.6 How to run

- `make run EXP=e059`
- `uv run python -m mathxlab.experiments.e059`

5.59.7 Outputs

This experiment follows the standard output contract:

- `out/e059/figures/` — generated figures (PNG)
- `out/e059/report.md` — short narrative report
- `out/e059/manifest.json` — snapshot metadata for the gallery

5.59.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 200000

Top 12 values of (n)/n (value, n):

- 4.29437 at n=166320
- 4.25455 at n=110880
- 4.23932 at n=196560
- 4.20000 at n=131040
- 4.18701 at n=138600
- 4.18701 at n=55440
- 4.15584 at n=83160
- 4.13333 at n=163800
- 4.13333 at n=151200

- 4.13333 at n=65520
- 4.12941 at n=171360
- 4.12430 at n=194040

Figure:

- fig_01_sigma_over_n.png

params.json (snapshot)


```
{
  "n_max": 200000,
  "top_k": 12
}
```

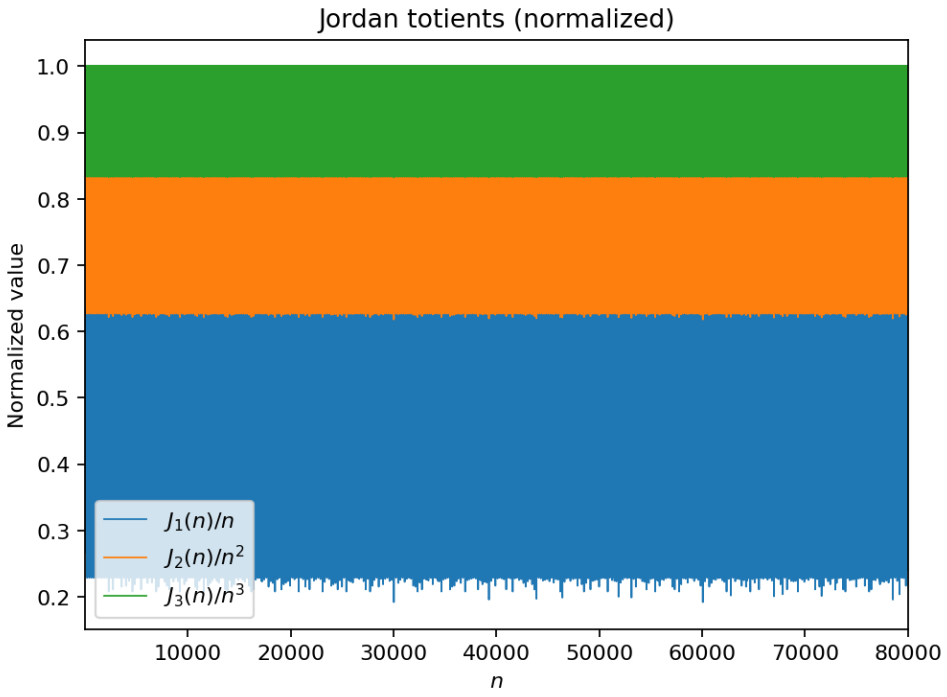

5.59.9 References

See Apostol [1976], Niven *et al.* [1991].

5.59.10 Related experiments

- *E097: (n)/n landscape: deficient, perfect, abundant* (E097: (n)/n landscape: deficient, perfect, abundant)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E058: Divisor-count record highs* (Divisor-count record highs)
- *E096: Record-holders for (n)* (E096: Record-holders for (n))
- *E052: Totient ratio landscape* (Totient ratio landscape)

5.60 E060: Jordan totients



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `totient`, `multiplicative` See: *Valid Tags*.

5.60.1 Highlights

- Compute $J_k(n)$ for k=1,2,3 and plot $J_k(n)/n^k$.
- Compare how the small-prime structure changes as k increases.

5.60.2 Goal

Show a clean generalization of (n) and how normalization reveals multiplicative structure.

5.60.3 Background (quick refresher)

- *Jordan totient J_k(n) refresher*
- *Euler's totient function \varphi(n) refresher*

5.60.4 Research question

How do the normalized Jordan totients $J_k(n)/n^k$ behave for k=1..3?

5.60.5 Method

- Compute $J_k(n) = n^k \prod_{p|n}(1 - p^{-k})$ via prime factors.
- Plot normalized values for k=1..3.

5.60.6 How to run

- `make run EXP=e060`
- `uv run python -m mathxlab.experiments.e060`

5.60.7 Outputs

This experiment follows the standard output contract:

- `out/e060/figures/` — generated figures (PNG)
- `out/e060/report.md` — short narrative report
- `out/e060/manifest.json` — snapshot metadata for the gallery

5.60.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 80000

Figure:

- fig_01_jordan_normalized.png

params.json (snapshot)

```
{
  "n_max": 80000
}
```

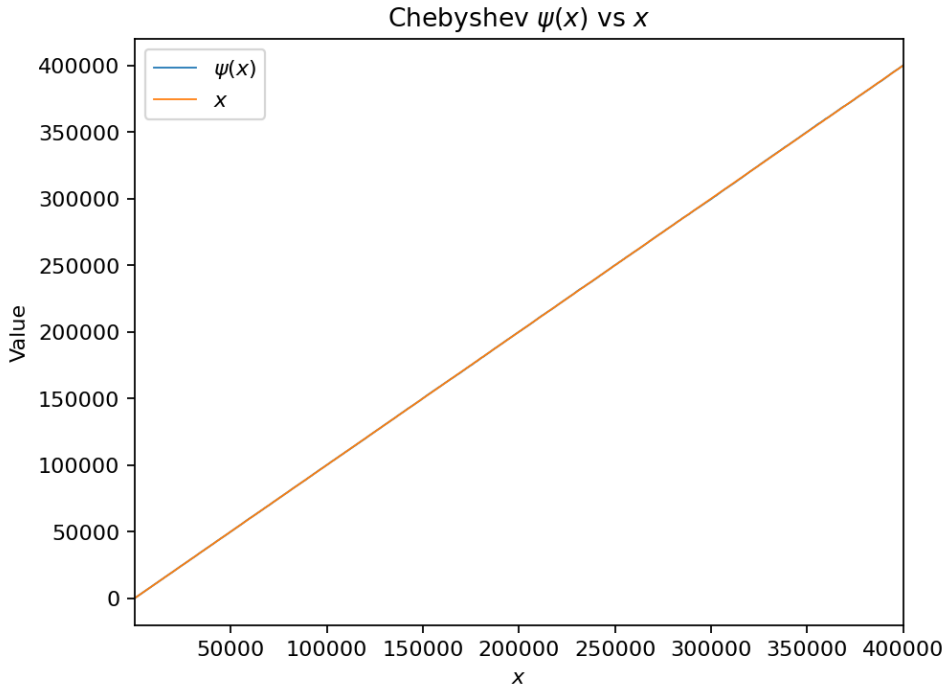
5.60.9 References

See Apostol [1976].

5.60.10 Related experiments

- *E099: Jordan totients J_k: identities and ratios* (E099: Jordan totients J_k: identities and ratios)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E102: Dirichlet convolution identity zoo* (E102: Dirichlet convolution identity zoo)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E062: Carmichael (n) vs. (n)* (Carmichael (n) vs. (n))

5.61 E061: Chebyshev ψ(x) and prime powers



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `mangoldt`, `pnt`, `summatory` See: *Valid Tags*.

5.61.1 Highlights

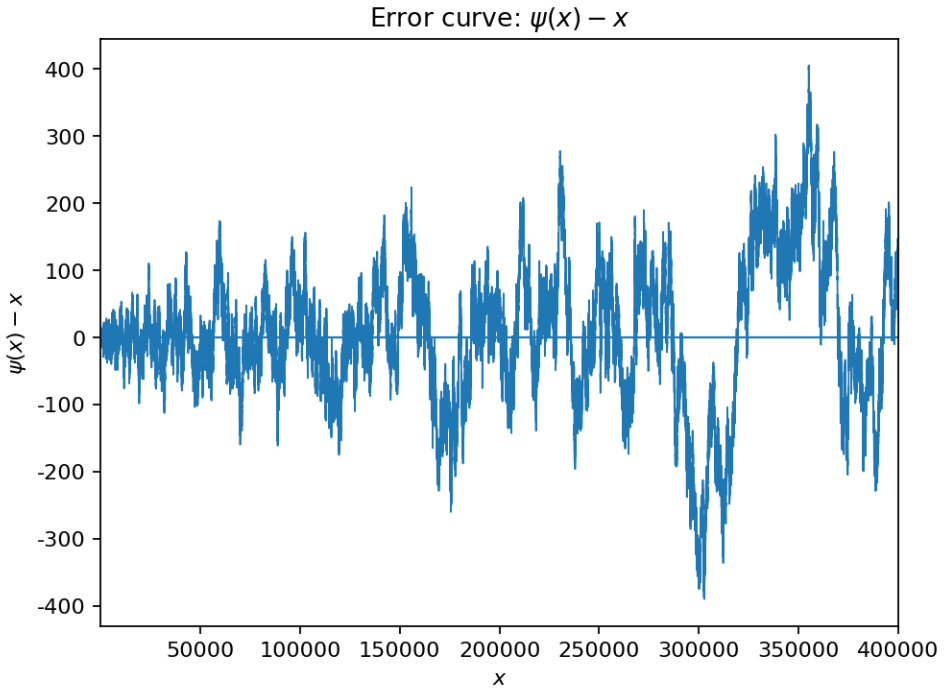
- Plot $\psi(x)$ and the error $\psi(x) - x$.
- Make the role of prime powers visible (jumps at p^k).

5.61.2 Goal

Build intuition for Chebyshev functions as "smoothed" prime counters.

5.61.3 Background (quick refresher)

- *von Mangoldt \Lambda(n) and Chebyshev functions refresher*
- *Prime numbers refresher*



5.61.4 Research question

Over a moderate range, what does $\psi(x) - x$ look like and where do the main jumps occur?

5.61.5 Method

- Compute Λ(n) (log p on prime powers) and cumulative (x).
- Plot (x) and (x)-x; optionally annotate largest jumps.

5.61.6 How to run

- `make run EXP=e061`
- `uv run python -m mathxlab.experiments.e061`

5.61.7 Outputs

This experiment follows the standard output contract:

- `out/e061/figures/` — generated figures (PNG)
- `out/e061/report.md` — short narrative report
- `out/e061/manifest.json` — snapshot metadata for the gallery

5.61.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 400000
- (n_max) - n_max: +108.161

Figures:

- fig_01_psi_vs_x.png
- fig_02_psi_minus_x.png

params.json (snapshot)


```
{
  "n_max": 400000
}
```

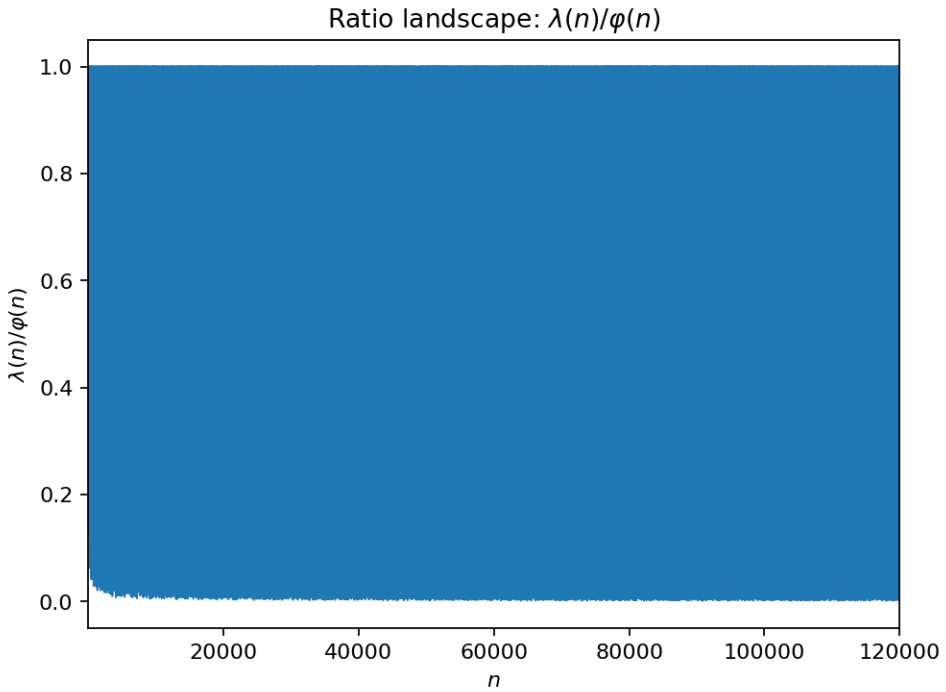

5.61.9 References

See Montgomery and Vaughan [2006], Apostol [1976].

5.61.10 Related experiments

- *E103: Chebyshev (x): prime powers drive jumps* (E103: Chebyshev (x): prime powers drive jumps)
- *E104: von Mangoldt Λ(n): support and statistics* (E104: von Mangoldt Λ(n): support and statistics)
- *E117: Prime-counting approximations: li(x) and friends* (E117: Prime-counting approximations: li(x) and friends)
- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E055: Mertens function walk* (Mertens function walk)

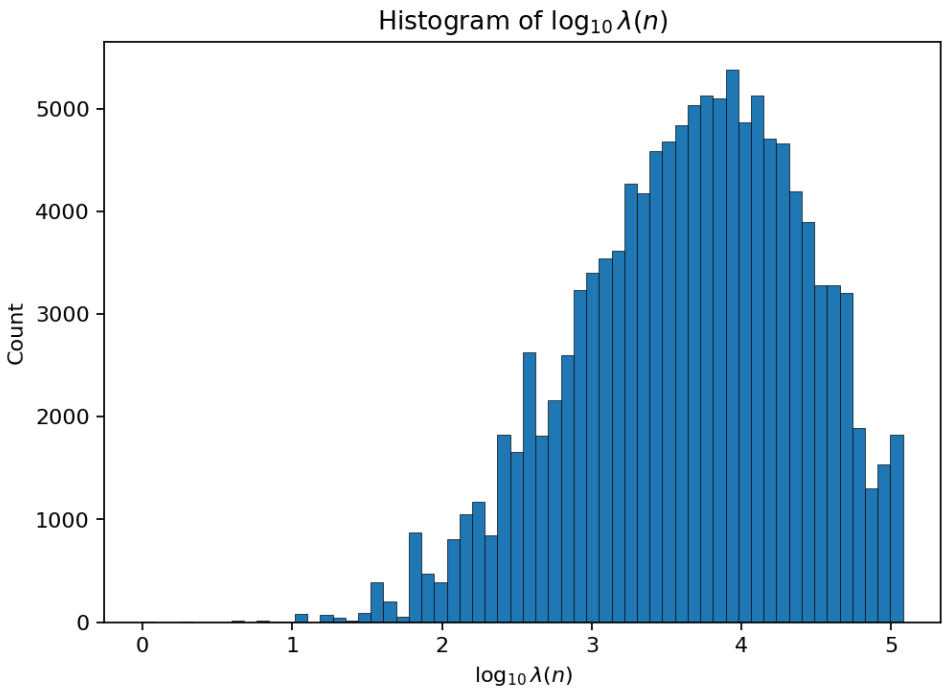
5.62 E062: Carmichael λ(n) vs. φ(n)



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `carmichael-lambda`, `totient` See: *Valid Tags*.

5.62.1 Highlights

- Plot or histogram the ratio $\varphi(n)/\lambda(n)$.
- Show how 2-adic structure affects (n) (notably for powers of 2).



5.62.2 Goal

See how the exponent of the multiplicative group mod n differs from its size.

5.62.3 Background (quick refresher)

- *Carmichael's \lambda(n) function refresher*
- *Euler's totient function \varphi(n) refresher*

5.62.4 Research question

How large can $\varphi(n)/\lambda(n)$ get for $n \leq N$, and what kinds of n cause large gaps?

5.62.5 Method

- Compute (n) and (n) from prime-power formulas and lcm composition.
- Visualize ratios (histograms, quantiles) and list top outliers.

5.62.6 How to run

- `make run EXP=e062`
- `uv run python -m mathxlab.experiments.e062`

5.62.7 Outputs

This experiment follows the standard output contract:

- `out/e062/figures/` — generated figures (PNG)
- `out/e062/report.md` — short narrative report
- `out/e062/manifest.json` — snapshot metadata for the gallery

5.62.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 120000
- min / : 0.000772
- max / : 1.000000

Figures:

- fig_01_ratio_lambda_over_phi.png
- fig_02_log10_lambda_hist.png

params.json (snapshot)


```
{
  "hist_bins": 60,
  "n_max": 120000
}
```


5.62.9 References

See Erdős *et al.* [1991], Montgomery and Vaughan [2006].

5.62.10 Related experiments

- *E100: Carmichael (n) vs. Euler (n)* (E100: Carmichael (n) vs. Euler (n))
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E060: Jordan totients* (Jordan totients)
- *E099: Jordan totients J_k: identities and ratios* (E099: Jordan totients J_k: identities and ratios)

5.63 E063: Dirichlet convolution playground

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `arithmetic-functions`, `dirichlet-convolution`, `model-checking` See: *Valid Tags*.

5.63.1 Highlights

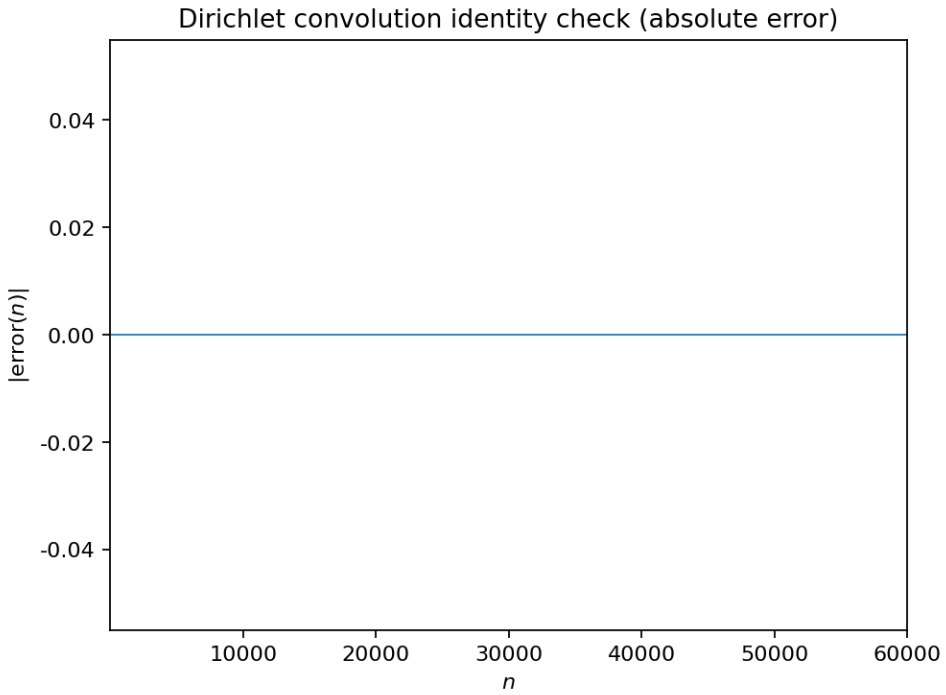
- Compute Dirichlet convolutions numerically on a finite range.
- Check exact identities and report any mismatches (should be zero).

5.63.2 Goal

Turn algebraic identities of arithmetic functions into concrete computed checks.

5.63.3 Background (quick refresher)

- *Dirichlet convolution refresher*
- *Arithmetic functions refresher*
- *Euler's totient function \varphi(n) refresher*



5.63.4 Research question

Can we numerically verify standard Dirichlet-convolution identities up to a bound without mistakes?

5.63.5 Method

- Implement divisor-sum convolution on [1..N].
- Compute *1 and* id and compare to and respectively.

5.63.6 How to run

- `make run EXP=e063`
- `uv run python -m mathxlab.experiments.e063`

5.63.7 Outputs

This experiment follows the standard output contract:

- `out/e063/figures/` — generated figures (PNG)
- `out/e063/report.md` — short narrative report
- `out/e063/manifest.json` — snapshot metadata for the gallery

5.63.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- n_max: 60000

Checks:

- * 1 = : OK
- * id = : OK

Figure:

- fig_01_abs_error.png

params.json (snapshot)

```
{
  "n_max": 60000
}
```

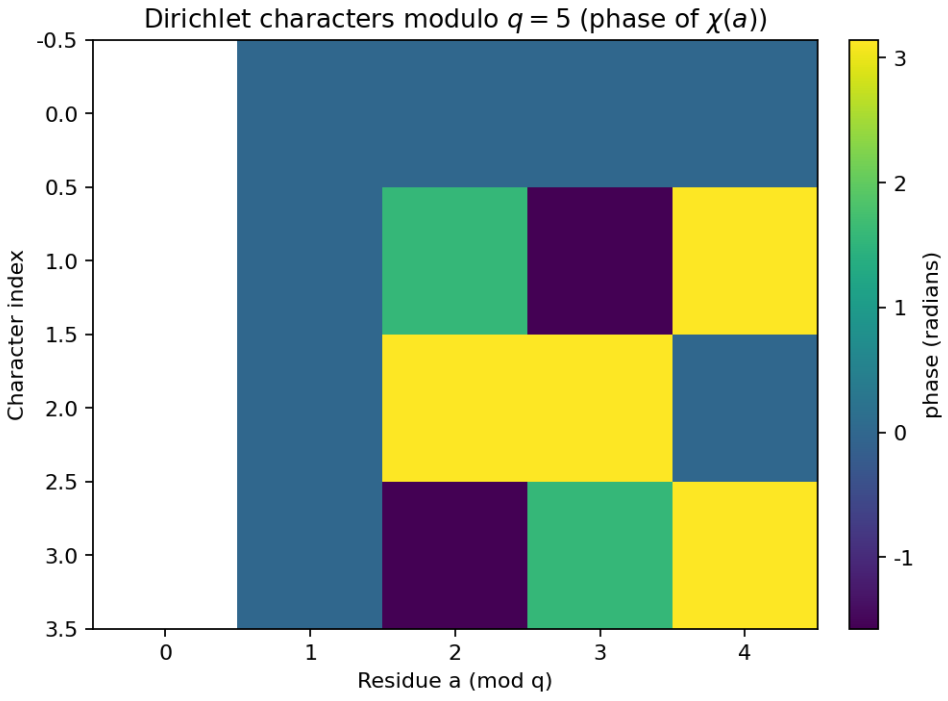
5.63.9 References

See Apostol [1976].

5.63.10 Related experiments

- *E102: Dirichlet convolution identity zoo* (E102: Dirichlet convolution identity zoo)
- *E121: Möbius inversion as convolution undo* (E121: Möbius inversion as convolution undo)
- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E055: Mertens function walk* (Mertens function walk)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)

5.64 E064: Dirichlet character tables (phase view).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.64.1 Highlights

- Constructs full character tables for a small modulus q.
- Visualizes values as phases / real parts to spot structure and symmetries.
- Checks orthogonality and basic sanity constraints numerically.

5.64.2 What this experiment does

This experiment enumerates all Dirichlet characters modulo a small modulus q and visualizes the values (a) for residues a=0..q-1.

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.64.3 Outputs

This experiment writes into `out/e064/`:

- `figures/fig_01_character_phases.png`

5.64.4 How to run

```
make run EXP=e064
```

5.64.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.64.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 5
- phi(q): 4

Figure:

- fig_01_character_phases.png

Notes:

- Non-units (gcd(a,q)>1) are shown as missing values.
- The principal character appears as the first row (all phases 0 on units).

params.json (snapshot)

```
{
  "q": 5
}
```

5.64.7 References

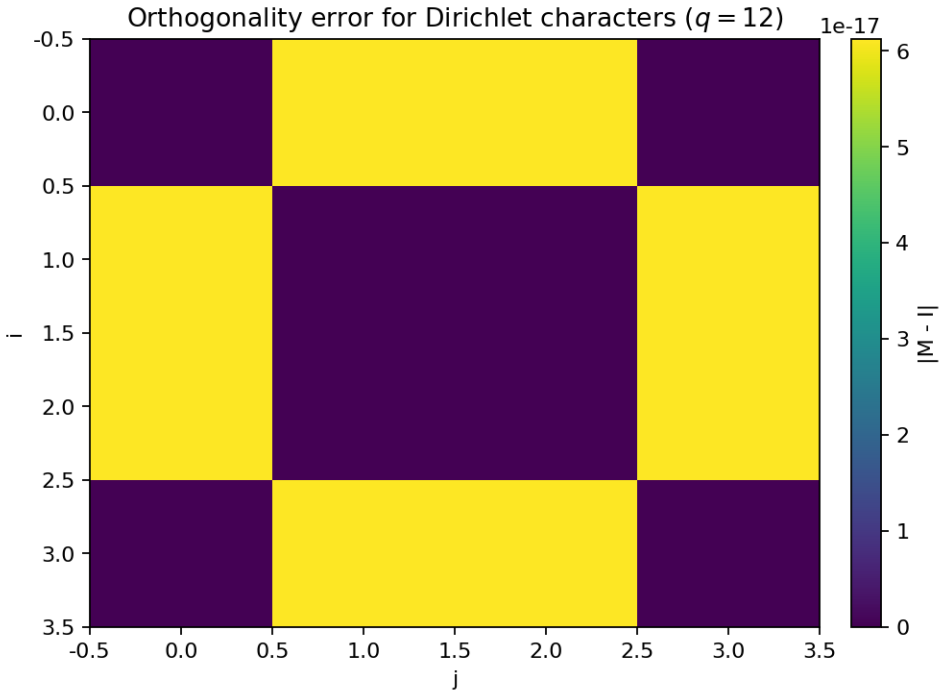
Davenport [2000], Niven *et al.* [1991]

5.64.8 Related experiments

- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)

- *E077: Indicator via character orthogonality (sanity check).* (Indicator via character orthogonality (sanity check).)
- *E122: Character averages over primes* (E122: Character averages over primes)

5.65 E065: Orthogonality matrix for Dirichlet characters.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.65.1 Highlights

- Constructs full character tables for a small modulus q.
- Visualizes values as phases / real parts to spot structure and symmetries.
- Checks orthogonality and basic sanity constraints numerically.

5.65.2 What this experiment does

For characters _i modulo q, orthogonality says:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.65.3 Outputs

This experiment writes into `out/e065/`:

- `figures/fig_01_orthogonality_error.png`

5.65.4 How to run

```
make run EXP=e065
```

5.65.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.65.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 12
- phi(q): 4
- max |M - I|: 6.123e-17

Figure:

- fig_01_orthogonality_error.png

params.json (snapshot)


```
{
  "q": 12
}
```


5.65.7 References

Davenport [2000], Niven *et al.* [1991]

5.65.8 Related experiments

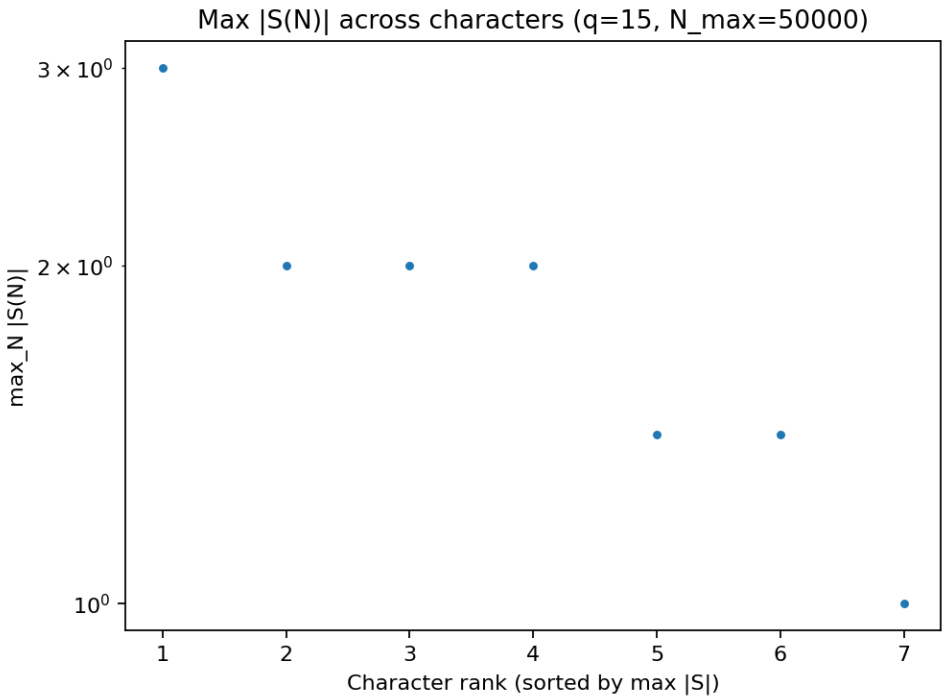
- *E108: Orthogonality heatmap for characters* (E108: Orthogonality heatmap for characters)
- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)
- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E077: Indicator via character orthogonality (sanity check).* (Indicator via character orthogonality (sanity check).)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)

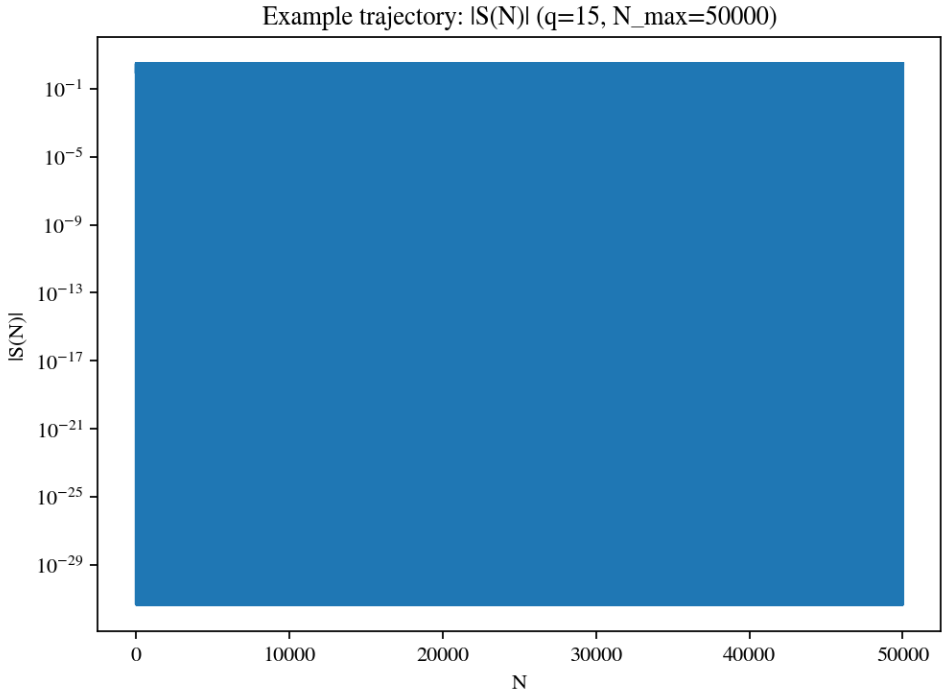
5.66 E066: Character partial sums: cancellation profiles.

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.66.1 Highlights

- Constructs full character tables for a small modulus q.
- Visualizes values as phases / real parts to spot structure and symmetries.
- Checks orthogonality and basic sanity constraints numerically.


Max |S(N)| across characters (q=15, N_max=50000)
3 × 10^0
2 × 10^0
10^0
max_N |S(N)|
1
2
3
4
5
6
7
Character rank (sorted by max |S|)


Example trajectory: |S(N)| (q=15, N_max=50000)
10^-1
10^-5
10^-9
10^-13
10^-17
10^-21
10^-25
10^-29
|S(N)|
0
10000
20000
30000
40000
50000
N

5.66.2 What this experiment does

For a Dirichlet character modulo q, consider partial sums:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.66.3 Outputs

This experiment writes into `out/e066/`:

- `figures/fig_01_max_partial_sums.png`
- `figures/fig_02_example_partial_sum.png`

5.66.4 How to run

```
make run EXP=e066
```

5.66.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.66.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Parameters

- q: 15
- N_max: 50000
- include_principal: False

5.66.7 Summary

Computed partial sums S(N)=sum_{n<=N} chi(n) for 7 Dirichlet characters modulo q.

Top characters by max |S(N)|

| rank | index | max |S(N)| | conductor | principal | |—:|—:|—:|—:|:—:| | 1 | 5 | 3 | 15 | False | | 2 | 0 | 2 | 5 | False | | 3 | 3 | 2 | 3 | False | | 4 | 2 | 2 | 5 | False | | 5 | 6 | 1.41421 | 15 | False | | 6 | 4 | 1.41421 | 15 | False | | 7 | 1 | 1 | 5 | False |

params.json (snapshot)


```
{
  "include_principal": false,
  "n_max": 50000,
  "q": 15,
  "top_k": 8
}
```

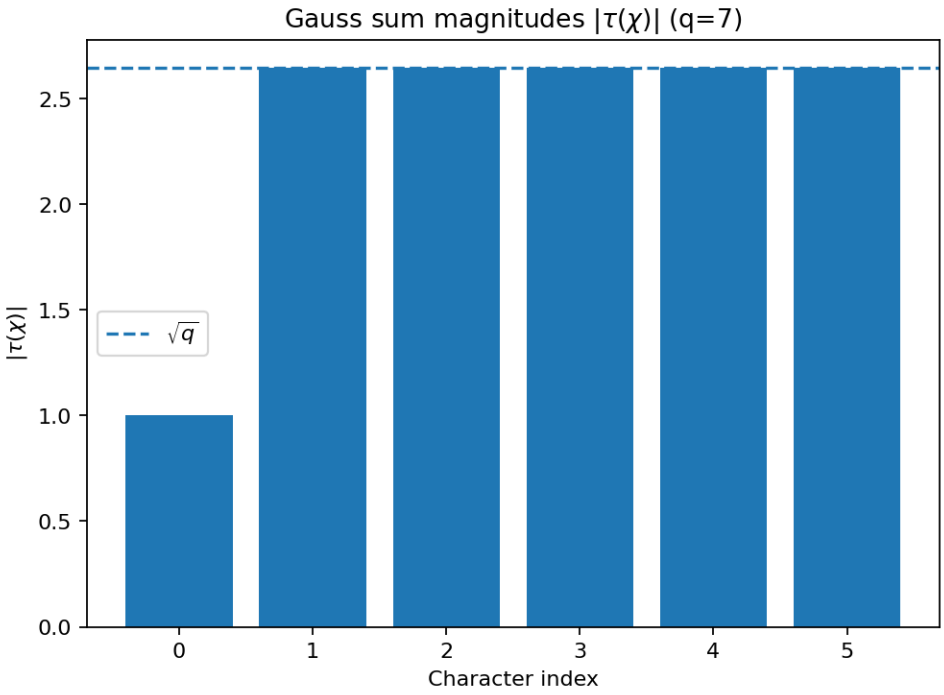
5.66.8 References

Davenport [2000], Niven *et al.* [1991]

5.66.9 Related experiments

- *E078: Max partial sums across characters.* (Max partial sums across characters.)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)
- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)
- *E067: Gauss sums: magnitude vs. sqrt(q).* (Gauss sums: magnitude vs. sqrt(q).)

5.67 E067: Gauss sums: magnitude vs. sqrt(q).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.67.1 Highlights

- Constructs full character tables for a small modulus q.
- Visualizes values as phases / real parts to spot structure and symmetries.
- Checks orthogonality and basic sanity constraints numerically.

5.67.2 What this experiment does

For a Dirichlet character modulo q, define the Gauss sum

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.67.3 Outputs

This experiment writes into `out/e067/`:

- `figures/fig_01_gauss_sum_magnitudes.png`

5.67.4 How to run

```
make run EXP=e067
```

5.67.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.67.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 7
- phi(q): 6
- sqrt(q): 2.645751
- mean |tau| (nontrivial): 2.645751
- min/max |tau| (nontrivial): 2.645751 / 2.645751

Figure:

- fig_01_gauss_sum_magnitudes.png

Notes:

- For prime q, nonprincipal characters are primitive, and |tau(chi)| clusters near sqrt(q).

params.json (snapshot)

```
{
  "q": 7
}
```

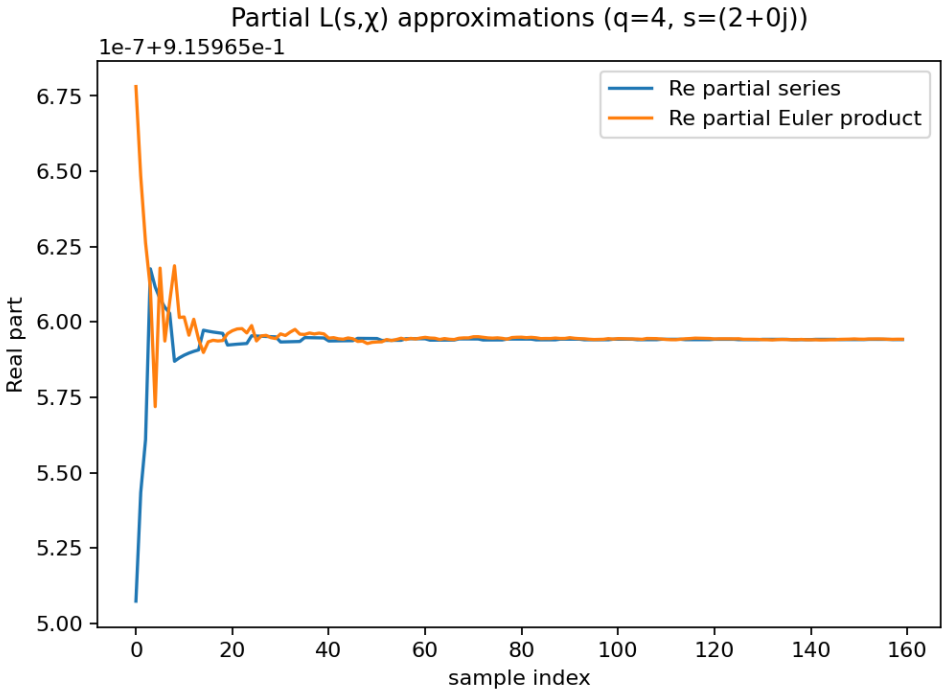
5.67.7 References

Davenport [2000], Niven *et al.* [1991]

5.67.8 Related experiments

- *E109: Gauss sums: magnitude patterns* (E109: Gauss sums: magnitude patterns)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)

5.68 E068: Dirichlet L(s,χ): series vs. Euler product (partial approximations).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`, `l-functions`

5.68.1 Highlights

- Computes truncated Dirichlet L-series for a nontrivial character.
- Compares series truncation against Euler-product truncation (where meaningful).
- Shows convergence behavior (fast for s>1, slow at s=1).

5.68.2 What this experiment does

For Re(s) > 1, Dirichlet L-functions admit both:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.68.3 Outputs

This experiment writes into `out/e068/`:

- `figures/fig_01_series_vs_euler.png`

5.68.4 How to run

```
make run EXP=e068
```

5.68.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.68.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 4
- s: (2+0j)
- n_max: 120000
- p_max: 400000
- |series - euler| (last sample): 1.170e-10

Figure:

- fig_01_series_vs_euler.png

Notes:

- Both approximations converge to the same limit for Re(s)>1.
- The Euler product is only over primes p not dividing q.

params.json (snapshot)


```
{
  "n_max": 120000,
  "n_points": 160,
  "p_max": 400000,
  "q": 4,
  "s_im": 0.0,
  "s_re": 2.0
}
```

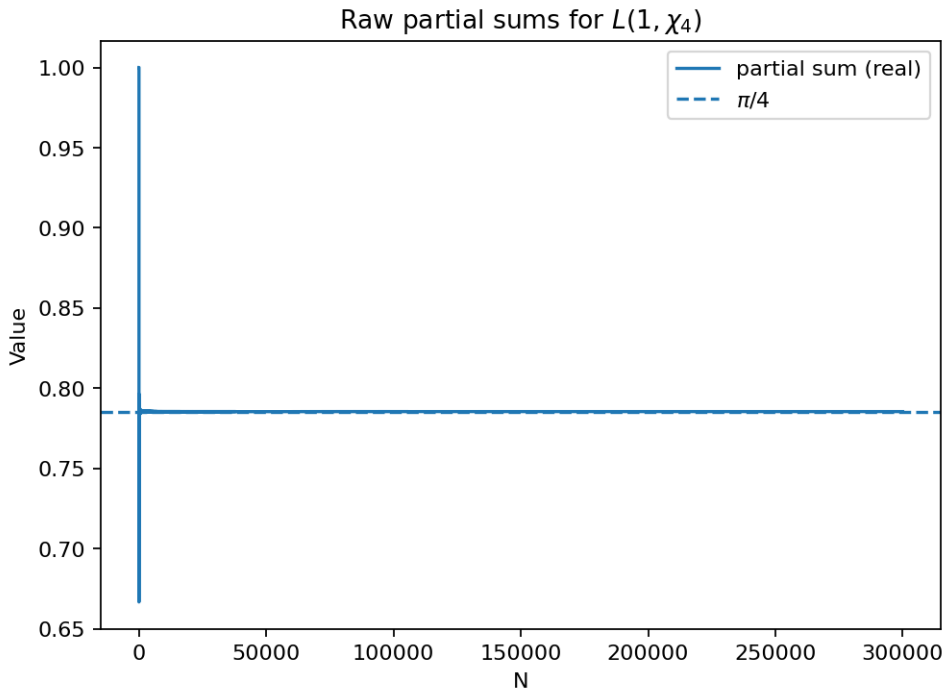

5.68.7 References

Apostol [1976], Montgomery and Vaughan [2006]

5.68.8 Related experiments

- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)

5.69 E069: L(1,χ): slow convergence and smoothing.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`, `l-functions`

5.69.1 Highlights

- Computes truncated Dirichlet L-series for a nontrivial character.
- Compares series truncation against Euler-product truncation (where meaningful).
- Shows convergence behavior (fast for s>1, slow at s=1).

5.69.2 What this experiment does

At s=1, the Dirichlet series for L(1,) converges very slowly. For the nontrivial character modulo 4:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

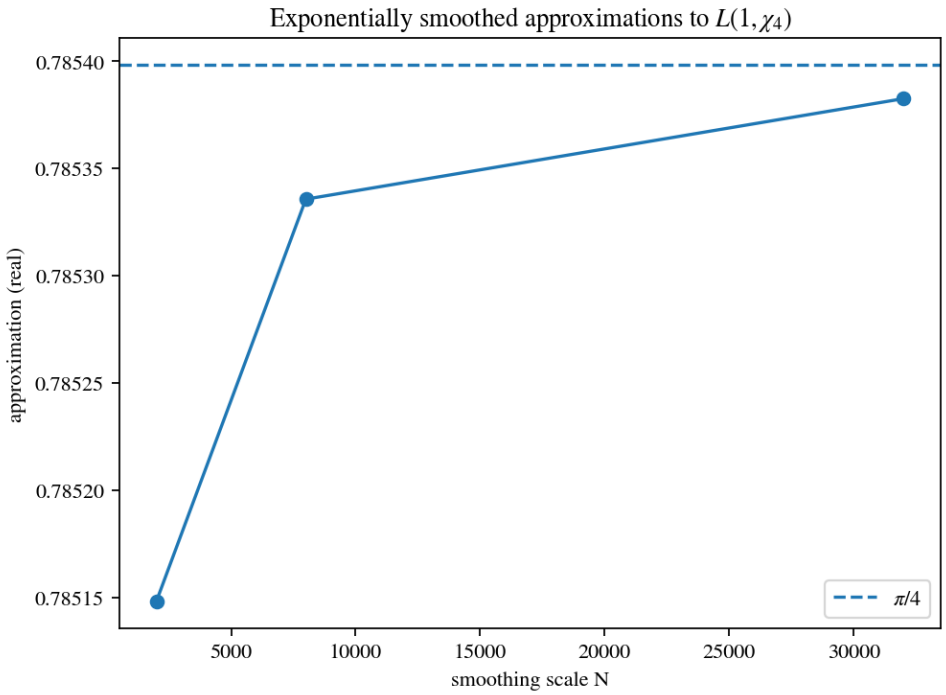
5.69.3 Outputs

This experiment writes into `out/e069/`:

- `figures/fig_01_l1_partial_sums.png`
- `figures/fig_02_l1_smoothed.png`

5.69.4 How to run

```
make run EXP=e069
```



5.69.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.69.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 4
- n_max: 300000
- last raw partial sum error vs pi/4: 1.667e-06
- smoothed scales: [2000, 8000, 32000]

Figures:

- fig_01_l1_partial_sums.png
- fig_02_l1_smoothed.png

params.json (snapshot)

```
{
  "n_max": 300000,
  "q": 4,
  "smooth_scales": [
    2000,
    8000,
    32000
  ]
}
```

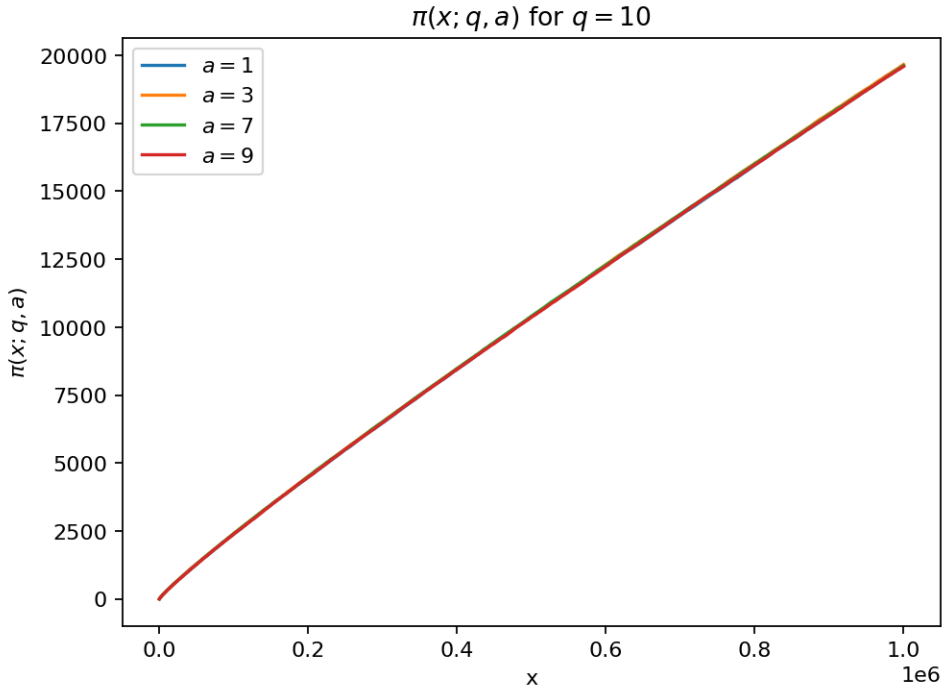
5.69.7 References

Apostol [1976], Montgomery and Vaughan [2006]

5.69.8 Related experiments

- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E107: Conductor: primitive vs. induced characters* (E107: Conductor: primitive vs. induced characters)
- *E122: Character averages over primes* (E122: Character averages over primes)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))

5.70 E070: Primes in residue classes: pi(x; q, a).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `aps`

5.70.1 Highlights

- Counts primes in residue classes a mod q for several a.
- Compares empirical counts to the first-order prediction li(x)/ (q).
- Tracks a simple error proxy across x to visualize deviation patterns.

5.70.2 What this experiment does

This experiment counts primes in selected reduced residue classes modulo q:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.70.3 Outputs

This experiment writes into `out/e070/`:

- `figures/fig_01_pi_x_q_a.png`

5.70.4 How to run

```
make run EXP=e070
```

5.70.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.70.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 10
- phi(q): 4
- residues: [1, 3, 7, 9]
- x_max: 1000000

Figure:

- fig_01_pi_x_q_a.png

Notes:

- Curves should be close for large x (equidistribution among reduced residue classes).

params.json (snapshot)


```
{
  "n_points": 700,
  "q": 10,
  "residues": [
    1,
    3,
    7,
    9
  ],
  "x_max": 1000000
}
```


5.70.7 References

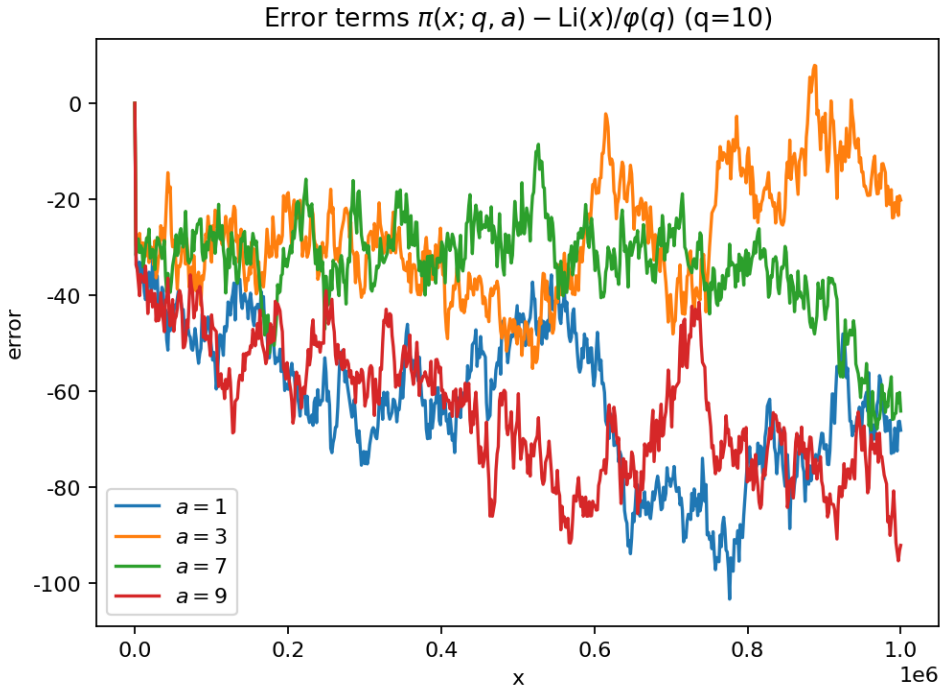
Apostol [1976], Davenport [2000]

5.70.8 Related experiments

- *E113: First prime in each residue class* (E113: First prime in each residue class)
- *E023: Residue class distribution mod q* (Residue class distribution mod q)

- *E071: PNT(AP) numerics: pi(x;q,a) - Li(x)/phi(q).* (PNT(AP) numerics: pi(x;q,a) - Li(x)/phi(q).)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)

5.71 E071: PNT(AP) numerics: pi(x;q,a) - Li(x)/phi(q).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `aps`

5.71.1 Highlights

- Counts primes in residue classes a mod q for several a.
- Compares empirical counts to the first-order prediction li(x)/ (q).
- Tracks a simple error proxy across x to visualize deviation patterns.

5.71.2 What this experiment does

The prime number theorem in arithmetic progressions suggests:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.71.3 Outputs

This experiment writes into `out/e071/`:

- `figures/fig_01_error_terms.png`

5.71.4 How to run

```
make run EXP=e071
```

5.71.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.71.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 10
- residues: [1, 3, 7, 9]
- x_max: 1000000
- li_step: 200

Figure:

- fig_01_error_terms.png

Notes:

- The errors oscillate; their fine behavior is linked to zeros of Dirichlet L-functions.

params.json (snapshot)


```
{
  "li_step": 200,
  "n_points": 650,
  "q": 10,
  "residues": [
    1,
    3,
    7,
    9
  ],
  "x_max": 1000000
}
```


5.71.7 References

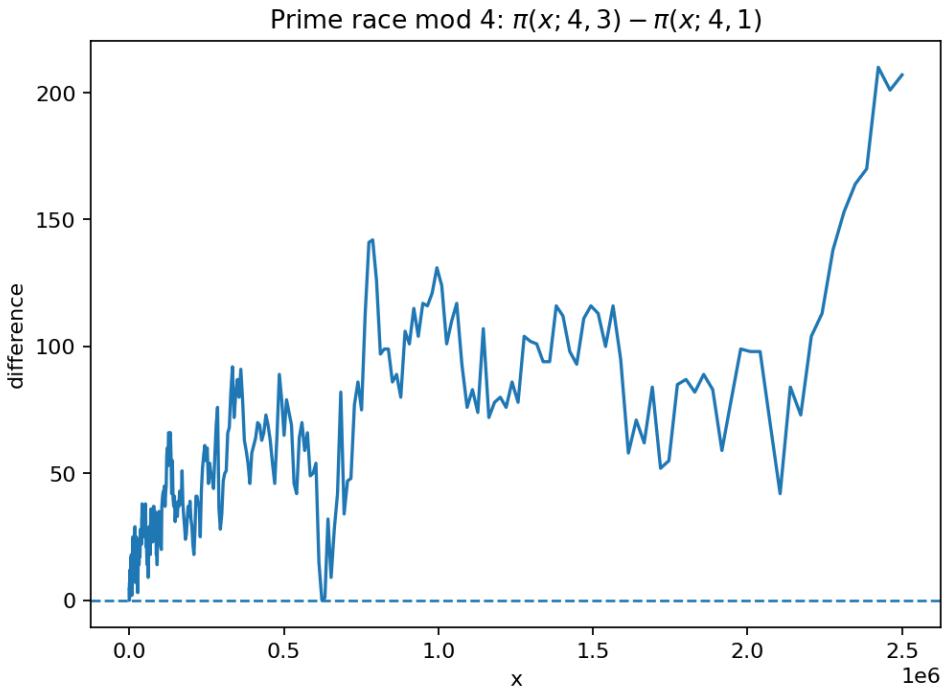
Apostol [1976], Davenport [2000]

5.71.8 Related experiments

- *E070: Primes in residue classes: pi(x; q, a).* (Primes in residue classes: pi(x; q, a).)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)

- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)

5.72 E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`, `aps`

5.72.1 Highlights

- Computes prime-race differences (x;q,a) − (x;q,b) on a grid of x values.
- Visualizes lead changes and the size of fluctuations as x grows.
- Adds a derived statistic (normalization / -variant) to compare behaviors.

5.72.2 What this experiment does

A classical “prime race” compares how often one residue class leads another in prime counts. The most famous is mod 4:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.72.3 Outputs

This experiment writes into `out/e072/`:

- `figures/fig_01_race_mod4_diff.png`

5.72.4 How to run

```
make run EXP=e072
```

5.72.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.72.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max: 2500000
- n_points: 900
- log_grid: True
- sign changes on sample grid: 0

Figure:

- fig_01_race_mod4_diff.png

Notes:

- The sample grid is not dense enough to capture every sign change; it gives a qualitative picture.

params.json (snapshot)


```
{
  "log_grid": true,
  "n_points": 900,
  "x_max": 2500000
}
```


5.72.7 References

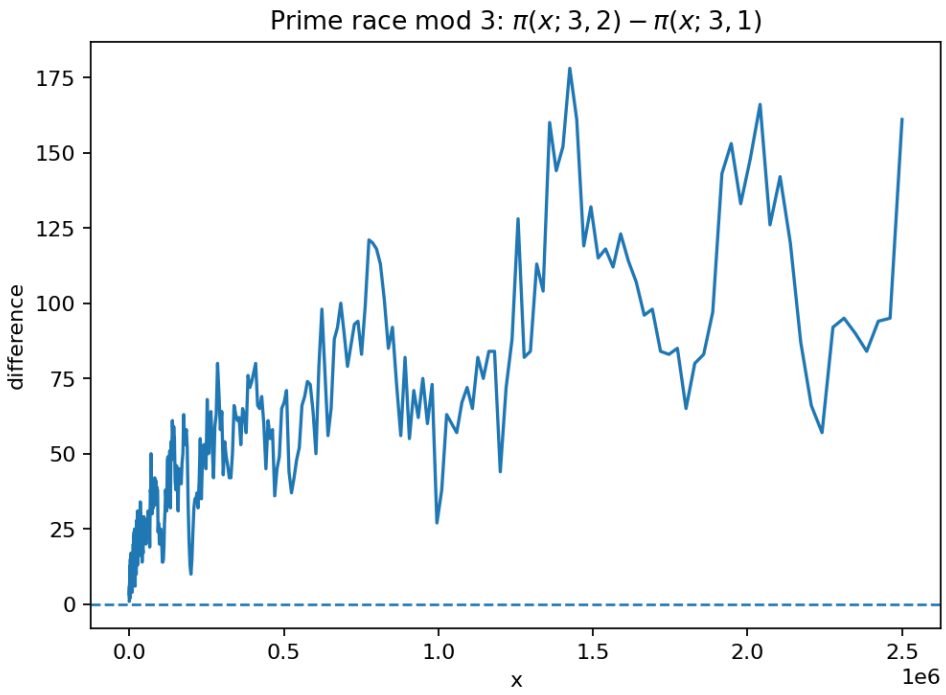
Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.72.8 Related experiments

- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)
- *E112: Prime race: (x;q,a) − (x;q,b)* (E112: Prime race: (x;q,a) − (x;q,b))
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.73 E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`, `aps`



5.73.1 Highlights

- Computes prime-race differences (x;q,a) − (x;q,b) on a grid of x values.
- Visualizes lead changes and the size of fluctuations as x grows.
- Adds a derived statistic (normalization / -variant) to compare behaviors.

5.73.2 What this experiment does

Another small prime race compares the two reduced residue classes modulo 3:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.73.3 Outputs

This experiment writes into `out/e073/`:

- `figures/fig_01_race_mod3_diff.png`

5.73.4 How to run

```
make run EXP=e073
```

5.73.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.73.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max: 2500000
- n_points: 900
- log_grid: True

Figure:

- fig_01_race_mod3_diff.png

params.json (snapshot)


```
{
  "log_grid": true,
  "n_points": 900,
  "x_max": 2500000
}
```


5.73.7 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.73.8 Related experiments

- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)
- *E112: Prime race: (x;q,a) − (x;q,b)* (E112: Prime race: (x;q,a) − (x;q,b))
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.74 E074: Prime race mod 8: leaderboard among 1,3,5,7.

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`, `aps`

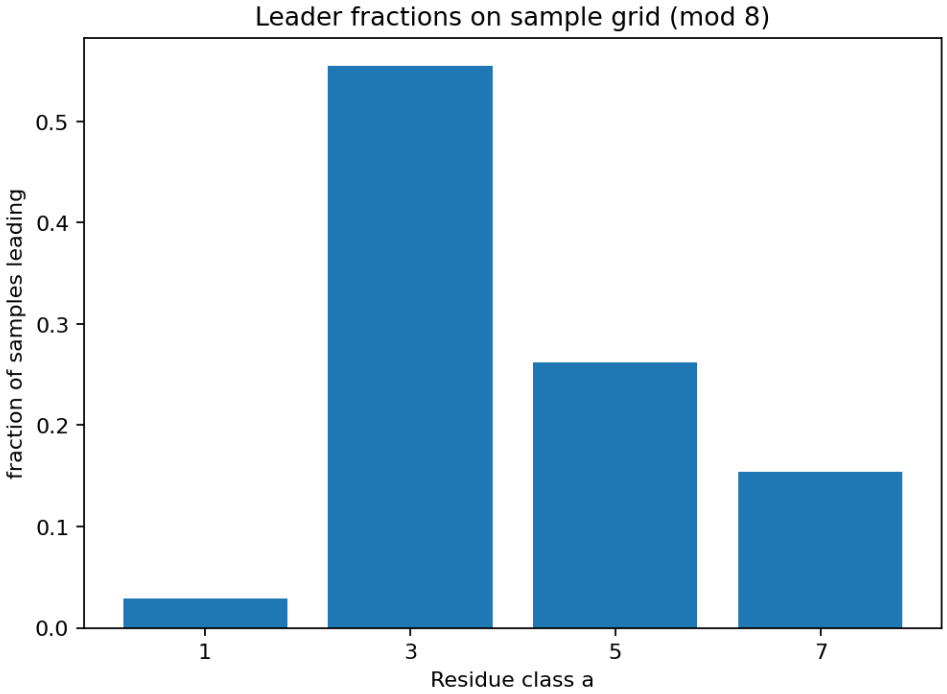
5.74.1 Highlights

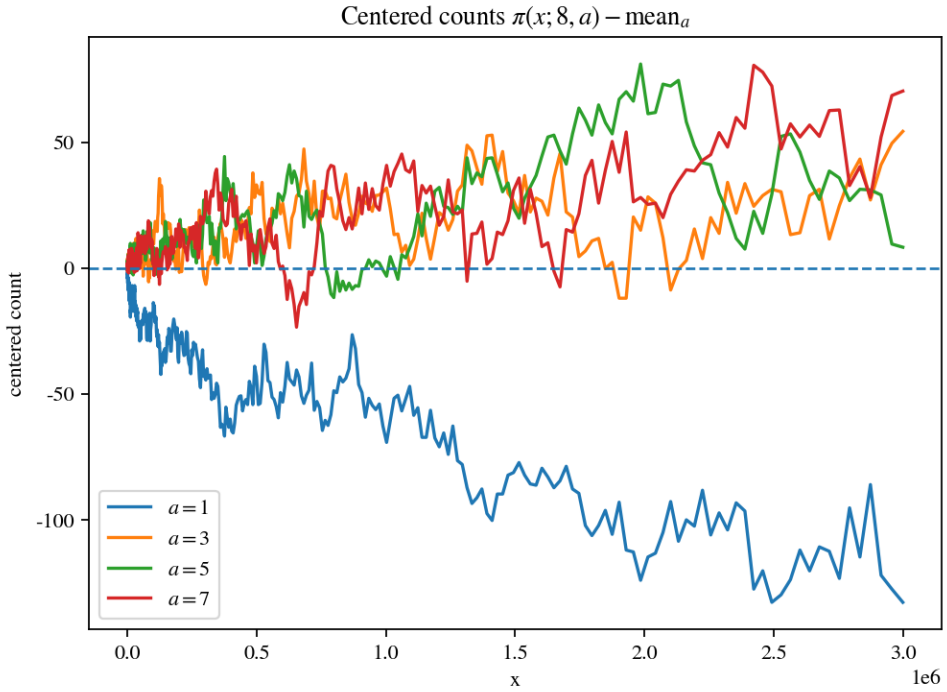
- Computes prime-race differences (x;q,a) − (x;q,b) on a grid of x values.
- Visualizes lead changes and the size of fluctuations as x grows.
- Adds a derived statistic (normalization / -variant) to compare behaviors.

5.74.2 What this experiment does

This experiment tracks four residue classes modulo 8:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.


Leader fractions on sample grid (mod 8)
fraction of samples leading
0.0
0.1
0.2
0.3
0.4
0.5
1
3
5
7
Residue class a


Centered counts $\pi(x;8,a) - \text{mean}_a$
centered count
50
0
-50
-100
$a=1$
$a=3$
$a=5$
$a=7$
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
1e6

5.74.3 Outputs

This experiment writes into `out/e074/`:

- `figures/fig_01_leader_fractions.png`
- `figures/fig_02_counts_minus_mean.png`

5.74.4 How to run

```
make run EXP=e074
```

5.74.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.74.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max: 3000000
- residues: [1, 3, 5, 7]
- n_points: 1000
- log_grid: True

Leader fractions on the sample grid:

- a=1: 0.029
- a=3: 0.555
- a=5: 0.262
- a=7: 0.154

Figures:

- fig_01_leader_fractions.png
- fig_02_counts_minus_mean.png

Notes:

- This uses a coarse sample grid, so fractions are approximate and depend on sampling.

params.json (snapshot)


```
{
  "log_grid": true,
  "n_points": 1000,
  "residues": [
    1,
    3,
    5,
    7
  ],
  "x_max": 3000000
}
```

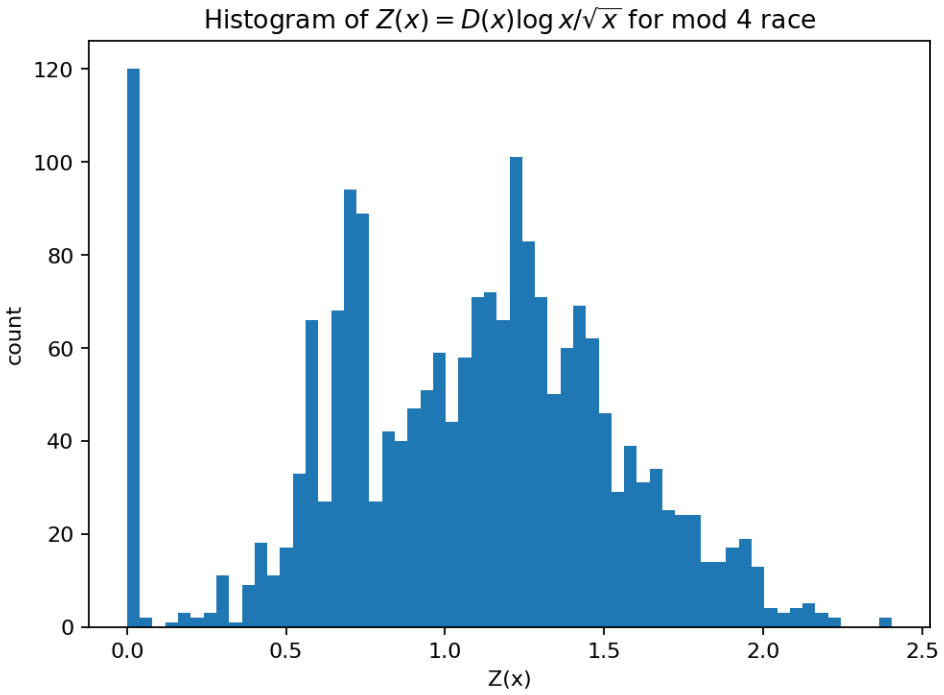
5.74.7 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.74.8 Related experiments

- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)
- *E112: Prime race: (x;q,a) − (x;q,b)* (E112: Prime race: (x;q,a) − (x;q,b))
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.75 E075: Prime race statistic: distribution on a log-grid.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`, `aps`

5.75.1 Highlights

- Computes prime-race differences (x;q,a) − (x;q,b) on a grid of x values.
- Visualizes lead changes and the size of fluctuations as x grows.
- Adds a derived statistic (normalization / -variant) to compare behaviors.

5.75.2 What this experiment does

For a prime race difference D(x), a common heuristic normalization is:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.75.3 Outputs

This experiment writes into `out/e075/`:

- `figures/fig_01_statistic_hist.png`

5.75.4 How to run

```
make run EXP=e075
```

5.75.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.75.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max: 8000000
- n_points: 2000
- bins: 60
- mean(Z): 1.064
- std(Z): 0.473

Figure:

- fig_01_statistic_hist.png

params.json (snapshot)


```
{
  "bins": 60,
  "n_points": 2000,
  "x_max": 8000000
}
```

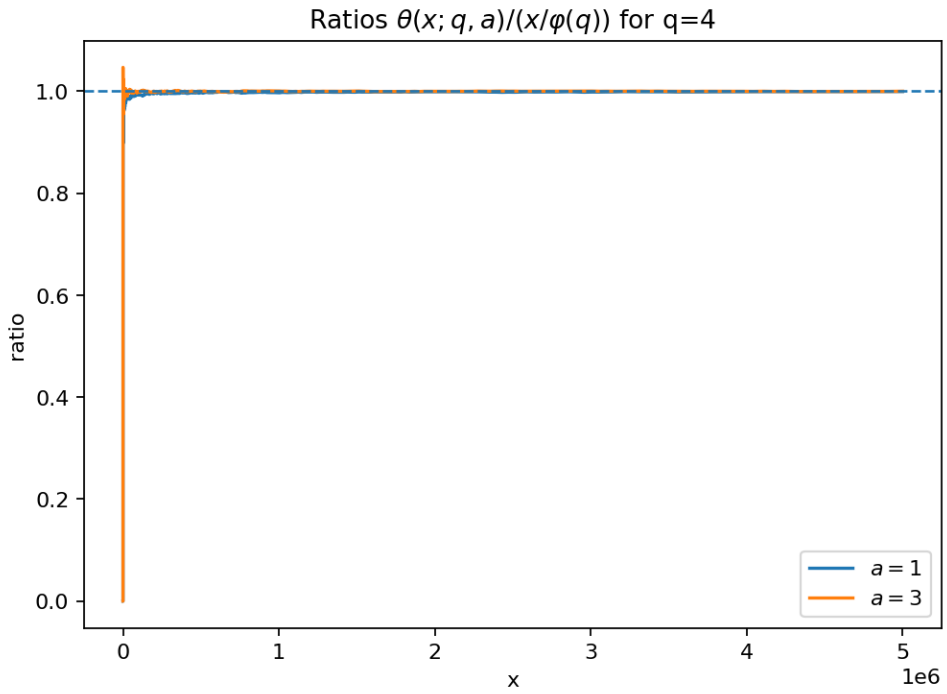

5.75.7 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.75.8 Related experiments

- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E112: Prime race: (x;q,a) − (x;q,b)* (E112: Prime race: (x;q,a) − (x;q,b))
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.76 E076: Chebyshev θ(x;q,a): weighted prime counts in progressions.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`, `aps`

5.76.1 Highlights

- Computes prime-race differences (x;q,a) − (x;q,b) on a grid of x values.
- Visualizes lead changes and the size of fluctuations as x grows.
- Adds a derived statistic (normalization / -variant) to compare behaviors.

5.76.2 What this experiment does

Define the Chebyshev theta function in a residue class:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.76.3 Outputs

This experiment writes into `out/e076/`:

- `figures/fig_01_theta_ratios.png`

5.76.4 How to run

```
make run EXP=e076
```

5.76.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.

- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.76.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 4
- residues: [1, 3]
- x_max: 5000000

Figure:

- fig_01_theta_ratios.png

Notes:

- Ratios fluctuate around 1 and encode distribution information beyond plain counts.

params.json (snapshot)


```
{
  "log_grid": true,
  "n_points": 900,
  "q": 4,
  "residues": [
    1,
    3
  ],
  "x_max": 5000000
}
```


5.76.7 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.76.8 Related experiments

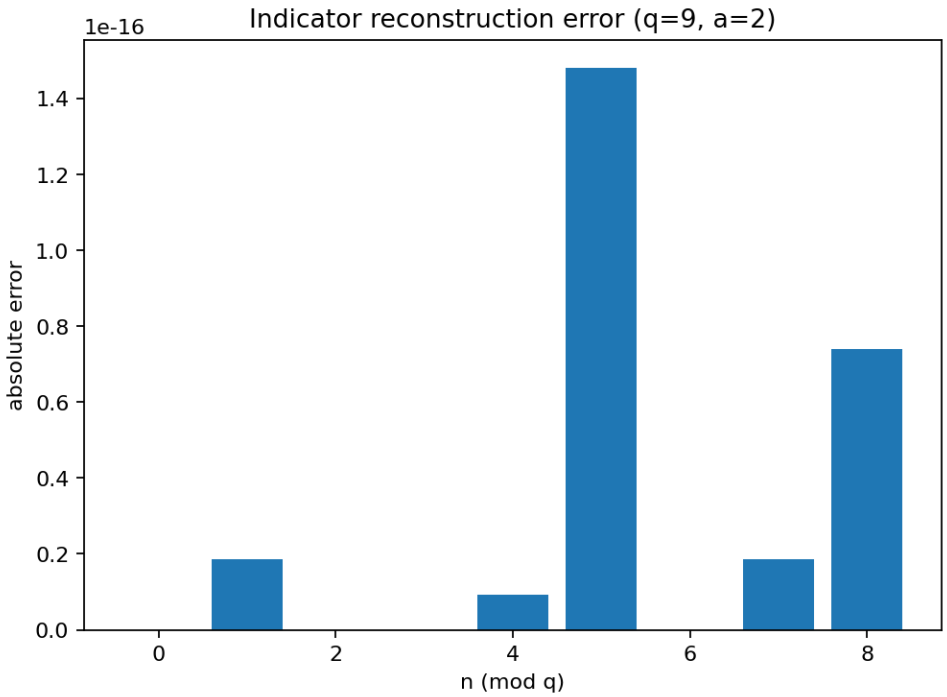
- *E080: Chebyshev bias: leader fraction vs. x.* (Chebyshev bias: leader fraction vs. x.)
- *E118: Chebyshev bias: lead-time statistics* (E118: Chebyshev bias: lead-time statistics)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)

5.77 E077: Indicator via character orthogonality (sanity check).

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.77.1 Highlights

- Uses character orthogonality to reconstruct residue indicators.
- Measures reconstruction error / maximal partial sums across characters.
- Explores how primitive characters vary with the modulus.



5.77.2 What this experiment does

A standard identity expresses an indicator of a residue class using Dirichlet characters. For gcd(a,q)=1 and gcd(n,q)=1:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.77.3 Outputs

This experiment writes into `out/e077/`:

- `figures/fig_01_indicator_error.png`

5.77.4 How to run

```
make run EXP=e077
```

5.77.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.77.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 9
- a_target: 2
- phi(q): 6
- max abs error: 1.665e-16

Figure:

- fig_01_indicator_error.png

Notes:

- The identity holds on units (gcd(n,q)=1). Non-units are outside the unit group, and (n)=0 there.

params.json (snapshot)


```
{
  "a_target": 2,
  "q": 9
}
```

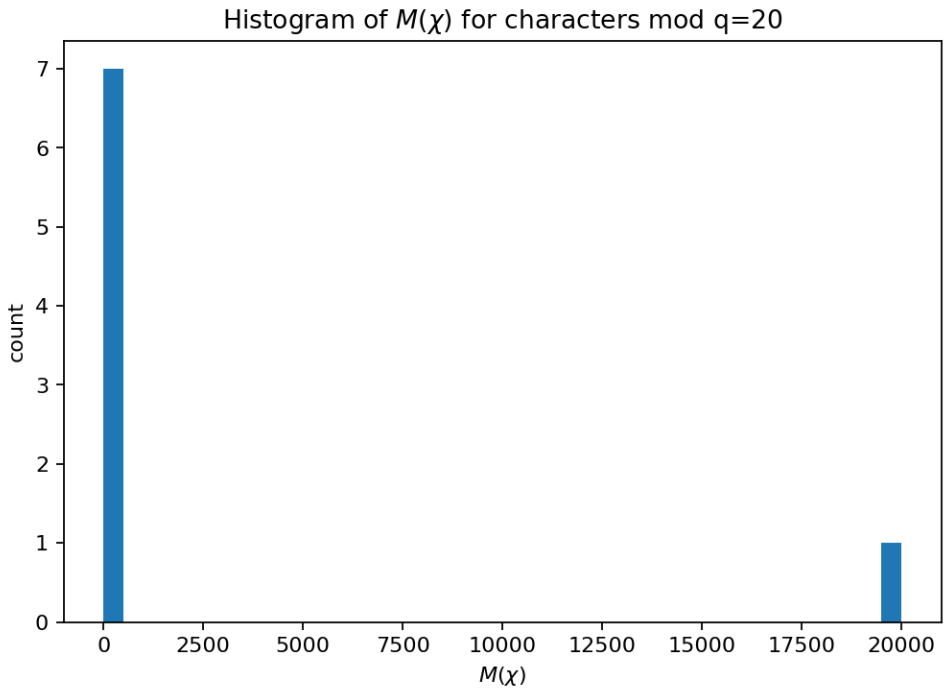

5.77.7 References

Davenport [2000], Niven *et al.* [1991]

5.77.8 Related experiments

- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)
- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E122: Character averages over primes* (E122: Character averages over primes)
- *E106: Character gallery: real vs. complex* (E106: Character gallery: real vs. complex)

5.78 E078: Max partial sums across characters.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.78.1 Highlights

- Uses character orthogonality to reconstruct residue indicators.
- Measures reconstruction error / maximal partial sums across characters.
- Explores how primitive characters vary with the modulus.

5.78.2 What this experiment does

For each Dirichlet character modulo q, define

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.78.3 Outputs

This experiment writes into `out/e078/`:

- `figures/fig_01_max_sums_hist.png`

5.78.4 How to run

```
make run EXP=e078
```

5.78.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.78.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 20
- phi(q): 8
- n_max: 50000
- mean M(chi): 2501.46
- max M(chi): 20000.00

Figure:

- fig_01_max_sums_hist.png

params.json (snapshot)

```
{
  "bins": 40,
  "n_max": 50000,
  "q": 20
}
```

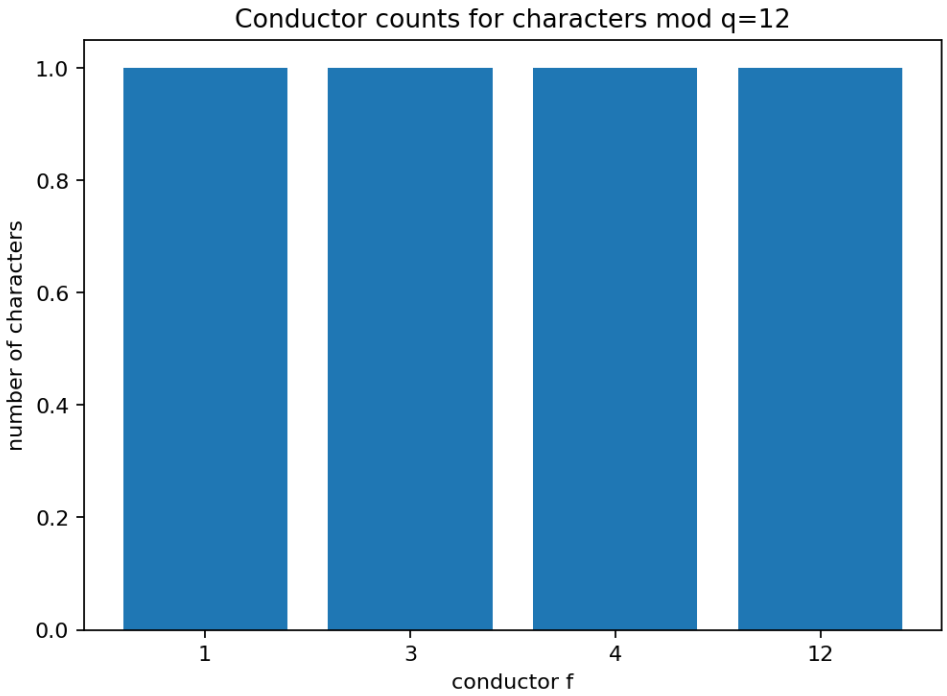
5.78.7 References

Davenport [2000], Niven *et al.* [1991]

5.78.8 Related experiments

- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)
- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)
- *E067: Gauss sums: magnitude vs. sqrt(q).* (Gauss sums: magnitude vs. sqrt(q).)

5.79 E079: Primitive vs. imprimitive characters: conductors.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `dirichlet-characters`

5.79.1 Highlights

- Uses character orthogonality to reconstruct residue indicators.
- Measures reconstruction error / maximal partial sums across characters.
- Explores how primitive characters vary with the modulus.

5.79.2 What this experiment does

A character modulo q may factor through a smaller modulus f | q. The smallest such modulus is the **conductor** of the character.

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.79.3 Outputs

This experiment writes into `out/e079/`:

- `figures/fig_01_conductor_counts.png`

5.79.4 How to run

```
make run EXP=e079
```

5.79.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.79.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- q: 12
- phi(q): 4
- number of primitive characters (conductor = q): 1

Conductor breakdown:

- f=1: 1
- f=3: 1
- f=4: 1
- f=12: 1

Figure:

- fig_01_conductor_counts.png

Notes:

- Conductor computation here is a brute-force check intended for small q.

params.json (snapshot)

```
{
  "q": 12
}
```

5.79.7 References

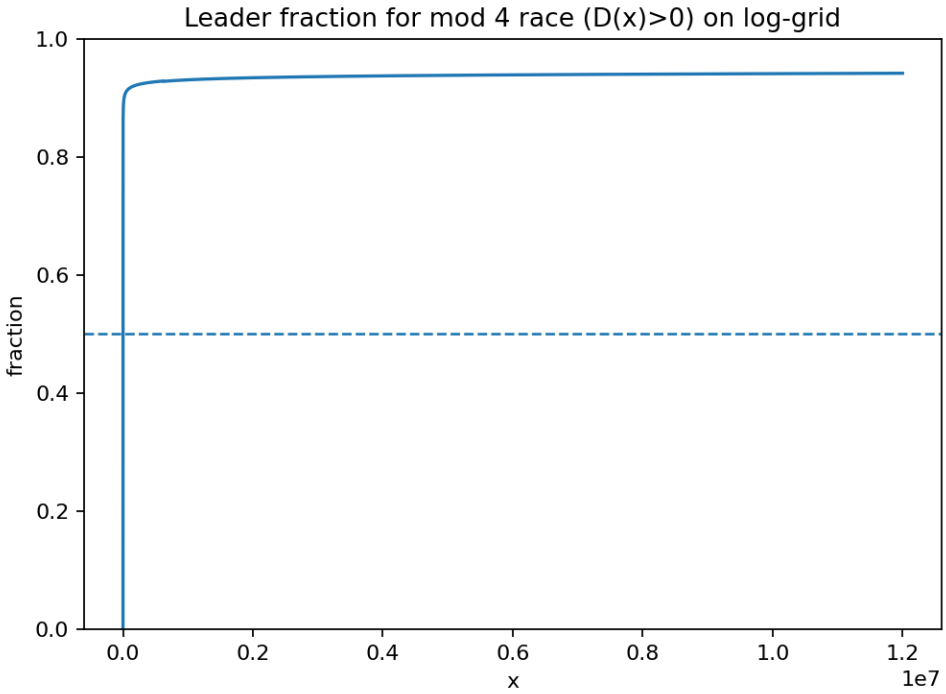
Davenport [2000], Niven *et al.* [1991]

5.79.8 Related experiments

- *E107: Conductor: primitive vs. induced characters* (E107: Conductor: primitive vs. induced characters)
- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)

- *E108: Orthogonality heatmap for characters* (E108: Orthogonality heatmap for characters)
- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)

5.80 E080: Chebyshev bias: leader fraction vs. x.



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`

5.80.1 Highlights

- Estimates the fraction of x where one residue class leads another.
- Plots a running bias estimate over log-sampled x values.
- Detects coarse sign-change neighborhoods for the race difference.

5.80.2 What this experiment does

For the mod 4 race D(x)= (x;4,3)- (x;4,1), define the empirical leader fraction:

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.80.3 Outputs

This experiment writes into `out/e080/`:

- `figures/fig_01_bias_fraction.png`

5.80.4 How to run

```
make run EXP=e080
```

5.80.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.80.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max: 12000000
- n_points: 4000
- final fraction (D>0): 0.941

Figure:

- fig_01_bias_fraction.png

Notes:

- This is sample-grid dependent; it is a qualitative bias indicator, not a rigorous density.

params.json (snapshot)


```
{
  "n_points": 4000,
  "x_max": 12000000
}
```


5.80.7 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.80.8 Related experiments

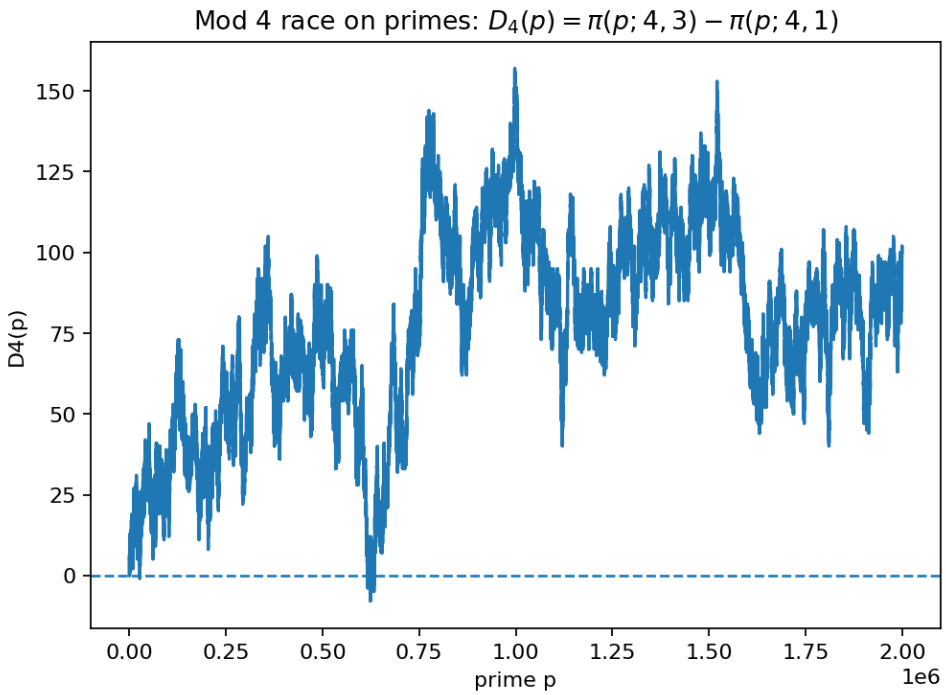
- *E118: Chebyshev bias: lead-time statistics* (E118: Chebyshev bias: lead-time statistics)
- *E076: Chebyshev (x;q,a): weighted prime counts in progressions.* (Chebyshev (x;q,a): weighted prime counts in progressions.)
- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)
- *E103: Chebyshev (x): prime powers drive jumps* (E103: Chebyshev (x): prime powers drive jumps)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)

5.81 E081: Prime race sign changes: first crossings table.

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `prime-races`

5.81.1 Highlights

- Estimates the fraction of x where one residue class leads another.
- Plots a running bias estimate over log-sampled x values.
- Detects coarse sign-change neighborhoods for the race difference.



5.81.2 What this experiment does

A prime race difference D(x) only changes when x passes a prime. We can track sign changes by iterating primes in order.

The implementation focuses on a compact, reproducible numerical workflow: deterministic parameter defaults, structured output folders, and one or more figures saved for the gallery.

5.81.3 Outputs

This experiment writes into `out/e081/`:

- `figures/fig_01_mod4_diff_on_primes.png`

5.81.4 How to run

```
make run EXP=e081
```

5.81.5 Notes

- The gallery preview figure shipped with the documentation uses conservative cutoffs so builds stay fast. If you run the experiment locally, increase the cutoffs to see the asymptotic regime more clearly.
- Prime-race plots depend on the chosen sampling of `x` (linear vs. log grid). The qualitative "who leads" picture can change when you zoom in.

5.81.6 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

- x_max (search): 25000000
- max_changes: 20

5.81.7 Mod 4: D4(x)=pi(x;4,3)-pi(x;4,1)

#	prime p	D after
1	26861	-1
2	26879	1
3	616841	-1
4	617039	1
5	617269	-1
6	617471	1
7	617521	-1
8	617587	1
9	617689	-1
10	617723	1
11	622813	-1
12	623387	1
13	623401	-1
14	623851	1
15	623933	-1
16	624031	1
17	624097	-1
18	624191	1
19	624241	-1
20	624259	1

5.81.8 Mod 3: D3(x)=pi(x;3,2)-pi(x;3,1)

(No sign change found in range.)

Figure:

- fig_01_mod4_diff_on_primes.png

Notes:

- Sign changes are detected at prime steps where D jumps.
- If no sign change appears, increase x_max.

params.json (snapshot)


```
{
  "max_changes": 20,
  "plot_max": 2000000,
  "x_max": 25000000
}
```


5.81.9 References

Granville and Martin [2006], Rubinstein and Sarnak [1994]

5.81.10 Related experiments

- *E115: Hardy Z: sign changes and zero bracketing* (E115: Hardy Z: sign changes and zero bracketing)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)

- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)

5.82 E082: Zeta(s) series convergence

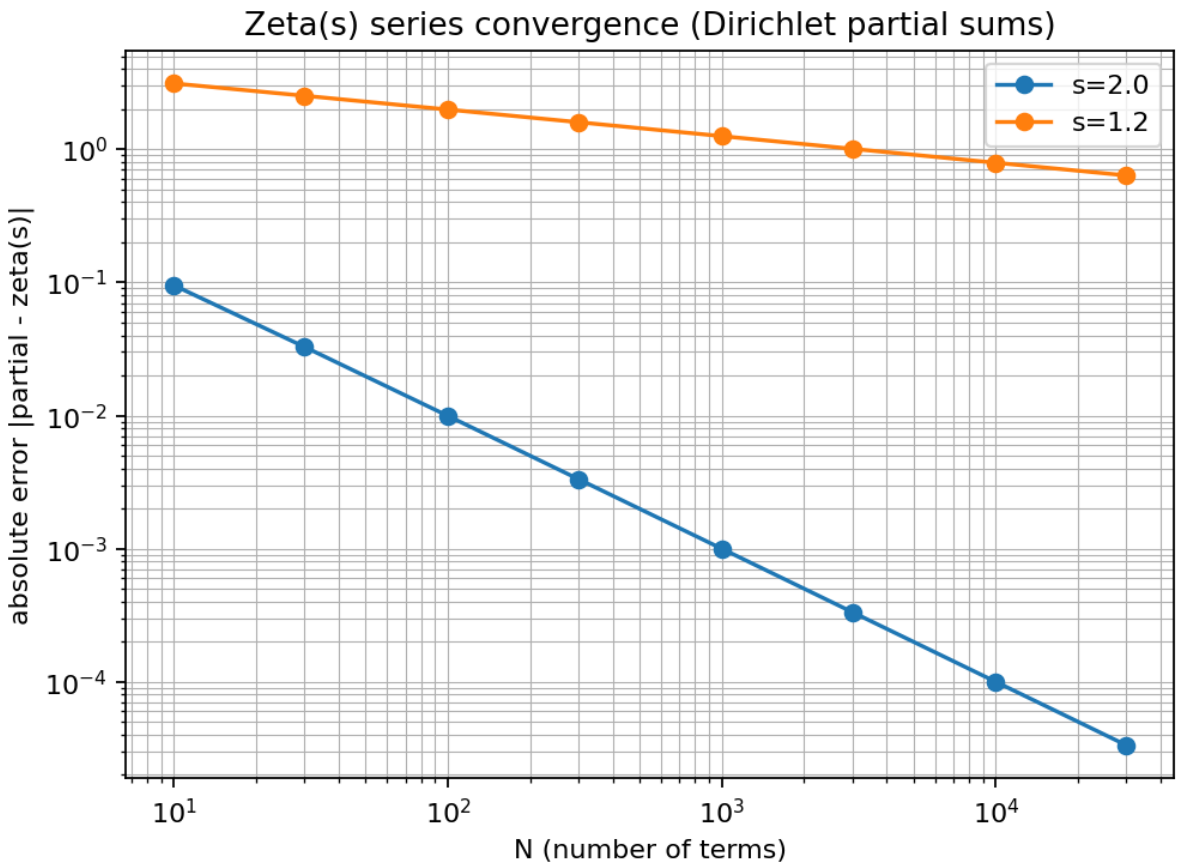


Fig. 1: E082: Zeta(s) series convergence

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `numerics`

5.82.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI "slow" suite (small defaults).

5.82.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.82.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.82.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_series_convergence.png` — main figure

5.82.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We approximate zeta(s) by partial sums of the Dirichlet series for two real s values.

- As s approaches 1, convergence becomes much slower.
- The plot shows the absolute error relative to a high-precision mpmath zeta(s).

params.json (snapshot)


```
{
  "mp_dps": 60,
  "n_values": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000,
    30000
  ],
  "s_values": [
    2.0,
    1.2
  ]
}
```


5.82.6 References

Edwards [1974], Titchmarsh [1986]

5.82.7 Related experiments

- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E085: Dirichlet eta acceleration for (s)* (E085: Dirichlet eta acceleration for (s))

5.83 E083: Series vs. Euler product (ζ)

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `numerics`

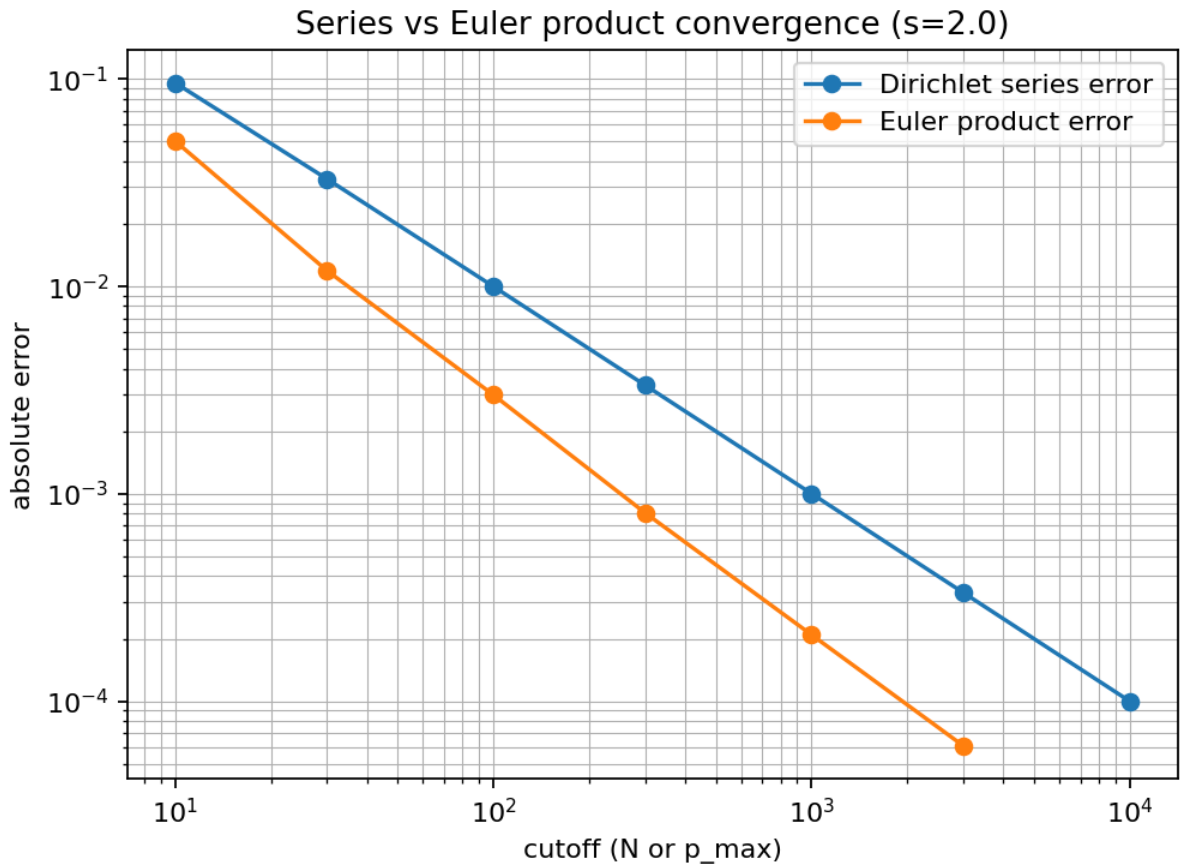


Fig. 2: E083: Series vs. Euler product ( )

5.83.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI "slow" suite (small defaults).

5.83.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.83.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.83.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_series_vs_euler_product.png` — main figure

5.83.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We compare partial approximations to zeta(s) coming from the series and the Euler product.

Note: both approximations converge for Re(s) $>$ 1, but their practical behavior depends on the cutoff choices.

params.json (snapshot)


```
{
  "mp_dps": 60,
  "n_values": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000
  ],
  "prime_cutoffs": [
    10,
    30,
    100,
    300,
    1000,
    3000
  ],
  "s": 2.0
}
```

5.83.6 References

Edwards [1974], Titchmarsh [1986]

5.83.7 Related experiments

- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)

5.84 E084: |ζ(1/2+it)| growth snapshots

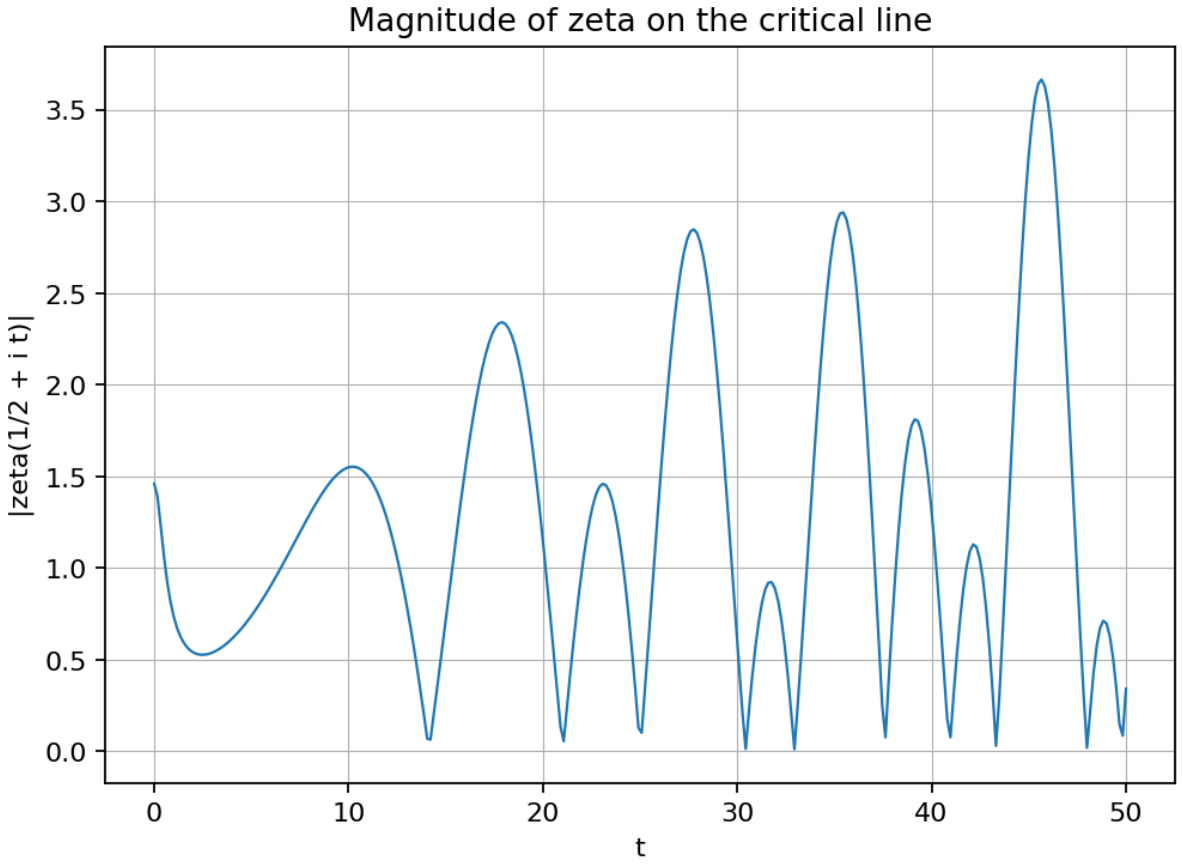


Fig. 3: E084: | (1/2+it)| growth snapshots

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `critical-line`, `numerics`

5.84.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI “slow” suite (small defaults).

5.84.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.84.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.84.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_critical_line_logabs.png` — main figure

5.84.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We sample the magnitude of zeta(s) along the critical line s = 1/2 + i t on a moderate t-range.

params.json (snapshot)

```
{
  "mp_dps": 50,
  "n_points": 300,
  "t_max": 50.0
}
```

5.84.6 References

Edwards [1974], Titchmarsh [1986]

5.84.7 Related experiments

- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E114: via : stability map on the critical line* (E114: via : stability map on the critical line)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E085: Dirichlet eta acceleration for (s)* (E085: Dirichlet eta acceleration for (s))

5.85 E085: Dirichlet eta acceleration for ζ(s)

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `numerics`

5.85.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI “slow” suite (small defaults).

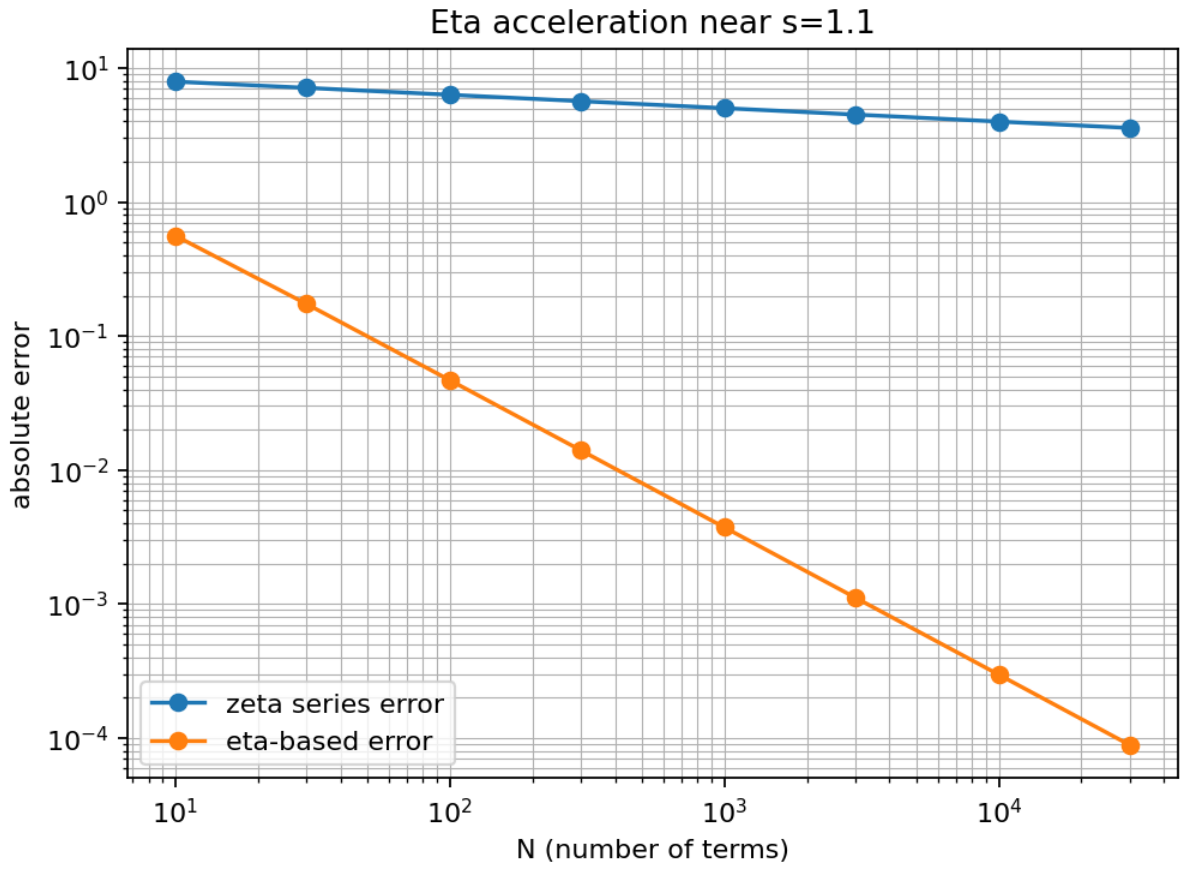


Fig. 4: E085: Dirichlet eta acceleration for (s)

5.85.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.85.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.85.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_eta_acceleration.png` — main figure

5.85.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We compare:

- naive partial sums of zeta(s)
- partial sums of eta(s), mapped back to zeta(s)

for s close to 1. The eta-based approach typically reduces cancellation issues and improves convergence.

params.json (snapshot)

```
{
  "mp_dps": 60,
  "n_values": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000,
    30000
  ],
  "s": 1.1
}
```

5.85.6 References

Edwards [1974], Titchmarsh [1986]

5.85.7 Related experiments

- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)

- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)

5.86 E086: Hardy Z(t) near zeros

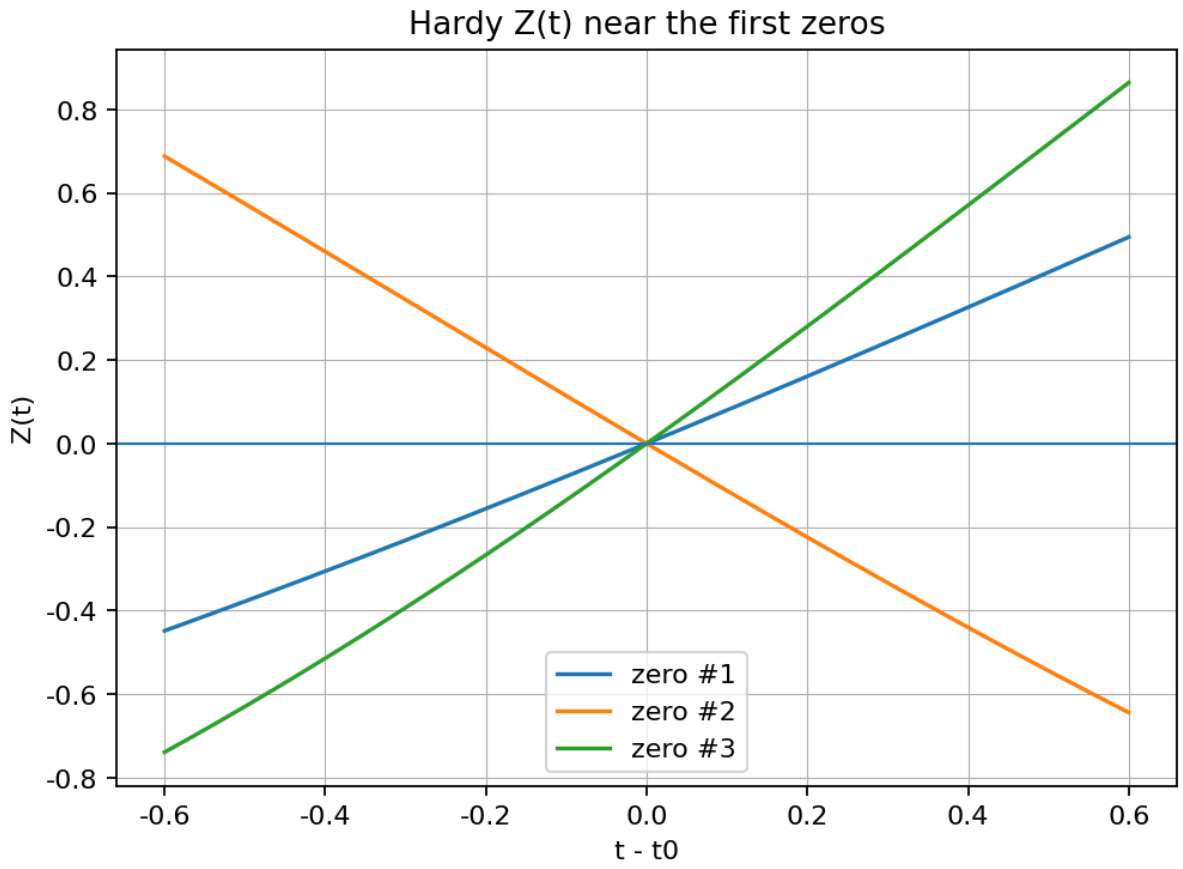


Fig. 5: E086: Hardy Z(t) near zeros

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `zeta-zeros`, `critical-line`, `numerics`

5.86.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI "slow" suite (small defaults).

5.86.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.86.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.86.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_hardy_Z_near_zeros.png` — main figure

5.86.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We locate the first few imaginary parts of non-trivial zeros (via mpmath) and sample Hardy's Z(t) in a small window around each zero.

params.json (snapshot)

```
{
  "mp_dps": 60,
  "n_points": 400,
  "n_zeros": 3,
  "window": 0.6
}
```

5.86.6 References

Edwards [1974], Titchmarsh [1986]

5.86.7 Related experiments

- *E115: Hardy Z: sign changes and zero bracketing* (E115: Hardy Z: sign changes and zero bracketing)
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E114: via : stability map on the critical line* (E114: via : stability map on the critical line)
- *E087: Gram points and spacing* (E087: Gram points and spacing)
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)

5.87 E087: Gram points and spacing

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `gram-points`, `zeta-zeros`, `numerics`

5.87.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI "slow" suite (small defaults).

5.87.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

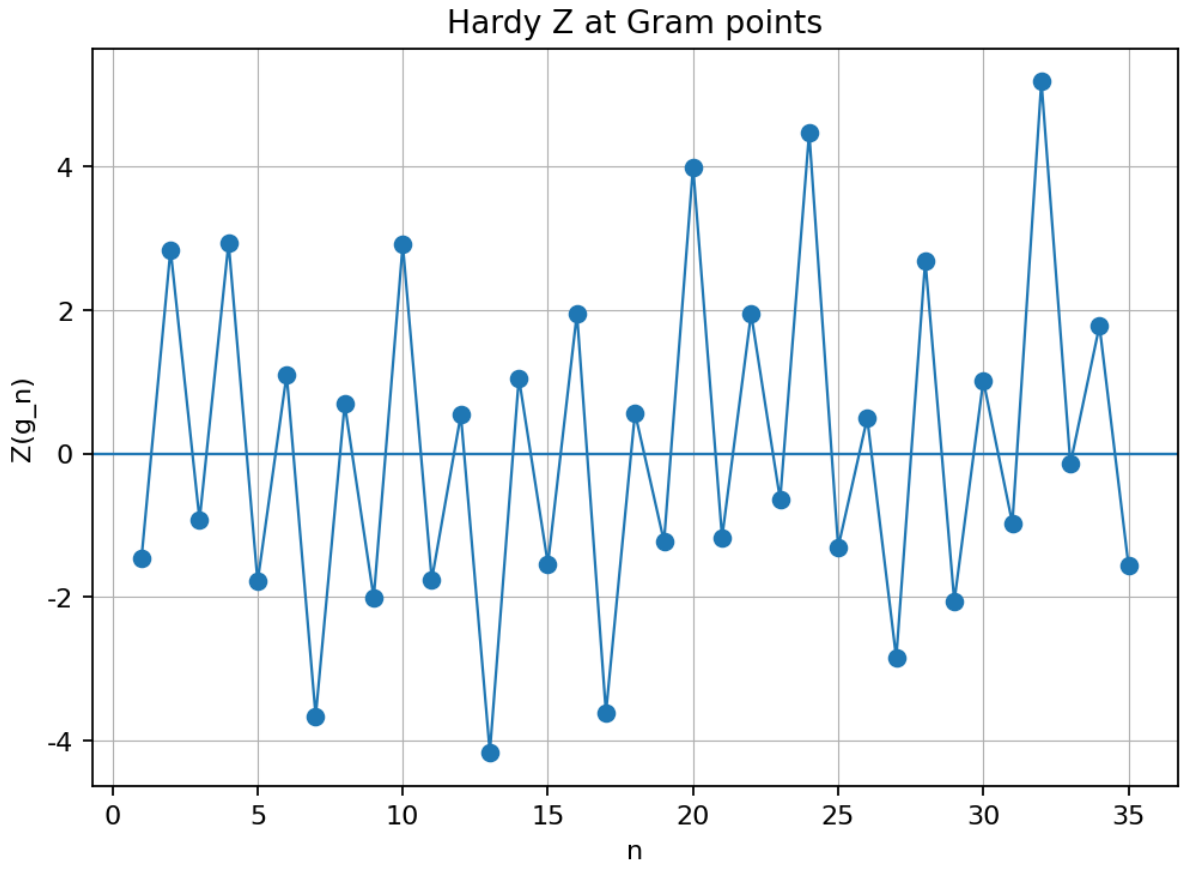


Fig. 6: E087: Gram points and spacing

5.87.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.87.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_gram_points_spacing.png` — main figure

5.87.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We computed Gram points g_n for n in [1, 35] and sampled Hardy Z(g_n).

Observed sign changes between consecutive Gram points (simple count): **34**.

params.json (snapshot)

```
{
  "mp_dps": 60,
  "n_end": 35,
  "n_start": 1
}
```

5.87.6 References

Edwards [1974], Montgomery [1973], Titchmarsh [1986]

5.87.7 Related experiments

- *E116: Gram points and zero-counting heuristics* (E116: Gram points and zero-counting heuristics)
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)
- *E115: Hardy Z: sign changes and zero bracketing* (E115: Hardy Z: sign changes and zero bracketing)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)

5.88 E088: Zero counting via Riemann–von Mangoldt

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `zeta-zeros`, `numerics`

5.88.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI “slow” suite (small defaults).

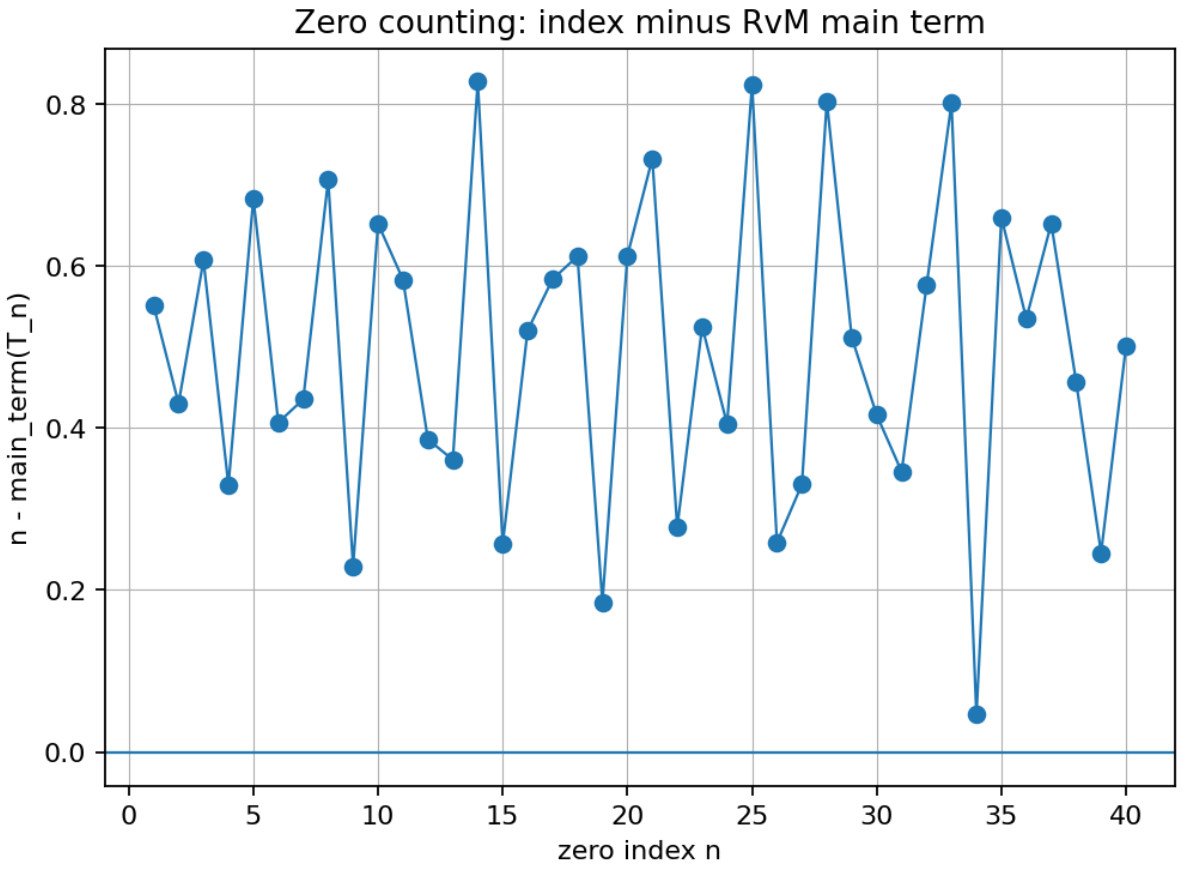


Fig. 7: E088: Zero counting via Riemann–von Mangoldt

5.88.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.88.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.88.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_rvm_zero_counting.png` — main figure

5.88.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We compare the index n of the nth non-trivial zero with the Riemann–von Mangoldt main-term approximation. The difference should remain relatively small compared to n and reflects the lower-order terms and fluctuations.

params.json (snapshot)

```
{
  "mp_dps": 70,
  "n_zeros": 40
}
```

5.88.6 References

Edwards [1974], Montgomery [1973], Titchmarsh [1986]

5.88.7 Related experiments

- *E116: Gram points and zero-counting heuristics* (E116: Gram points and zero-counting heuristics)
- *E115: Hardy Z: sign changes and zero bracketing* (E115: Hardy Z: sign changes and zero bracketing)
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E087: Gram points and spacing* (E087: Gram points and spacing)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)

5.89 E089: log|ζ(s)| heatmap

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `numerics`

5.89.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Lightweight computation suitable for CI “slow” suite (small defaults).

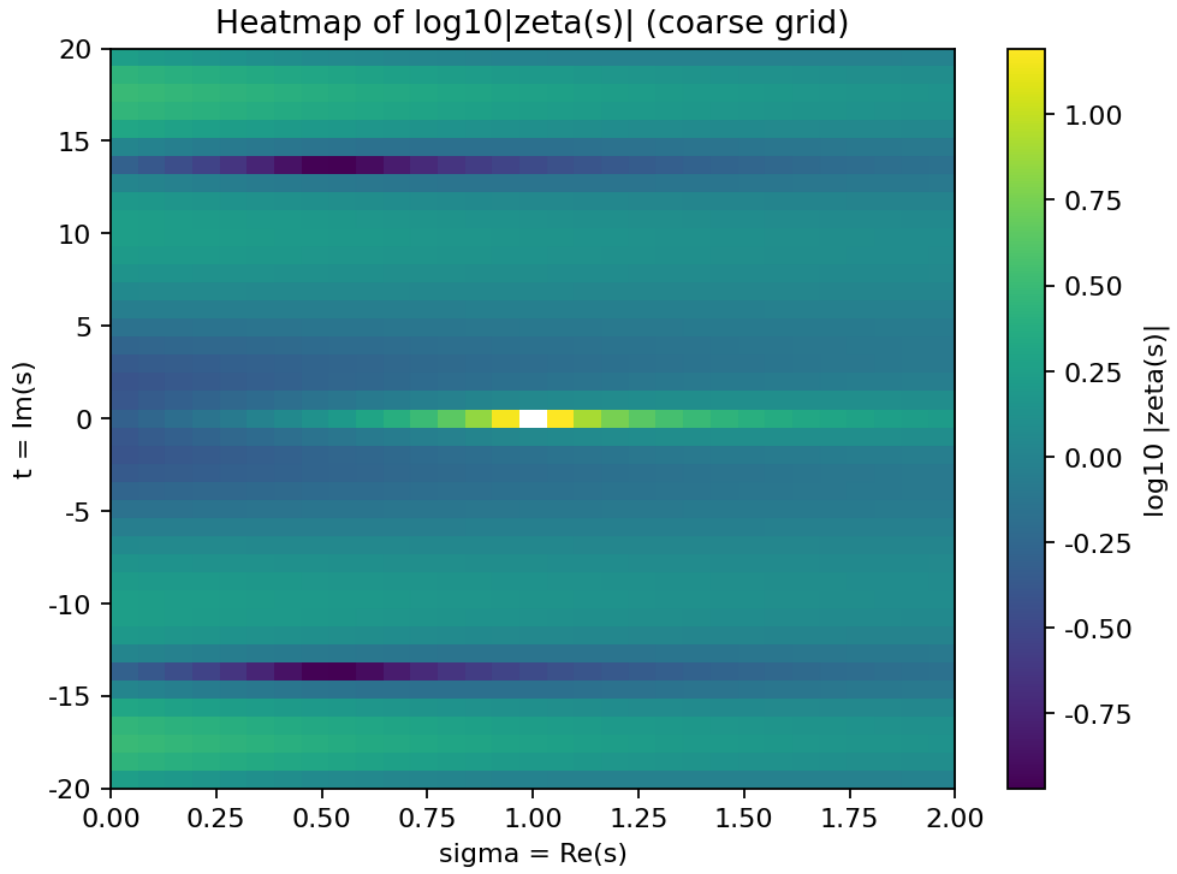


Fig. 8: E089: log| (s)| heatmap

5.89.2 What is computed

- A parameterized numeric evaluation related to the Riemann zeta function.
- A visualization summarizing the main phenomenon for the chosen parameter range.

5.89.3 Algorithm sketch

1. Build the numeric grid / sampling points.
2. Evaluate the target quantity with controlled truncation.
3. Render the figure and write a short report.

5.89.4 Outputs

- `report.md` — short narrative summary
- `params.json` — experiment parameters snapshot
- `figures/fig_01_zeta_heatmap_logabs.png` — main figure

5.89.5 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We plot log10|zeta(s)| on a coarse grid in the (sigma, t) plane. The pole at s=1 manifests as a bright region near sigma=1, t=0.

params.json (snapshot)


```
{
  "mp_dps": 50,
  "n_sigma": 31,
  "n_t": 41,
  "sigma_max": 2.0,
  "sigma_min": 0.0,
  "t_max": 20.0,
  "t_min": -20.0
}
```


5.89.6 References

Edwards [1974], Titchmarsh [1986]

5.89.7 Related experiments

- *E090: Functional equation residual heatmap* (E090: Functional equation residual heatmap)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E085: Dirichlet eta acceleration for (s)* (E085: Dirichlet eta acceleration for (s))

5.90 E090: Functional equation residual heatmap

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `numerics`

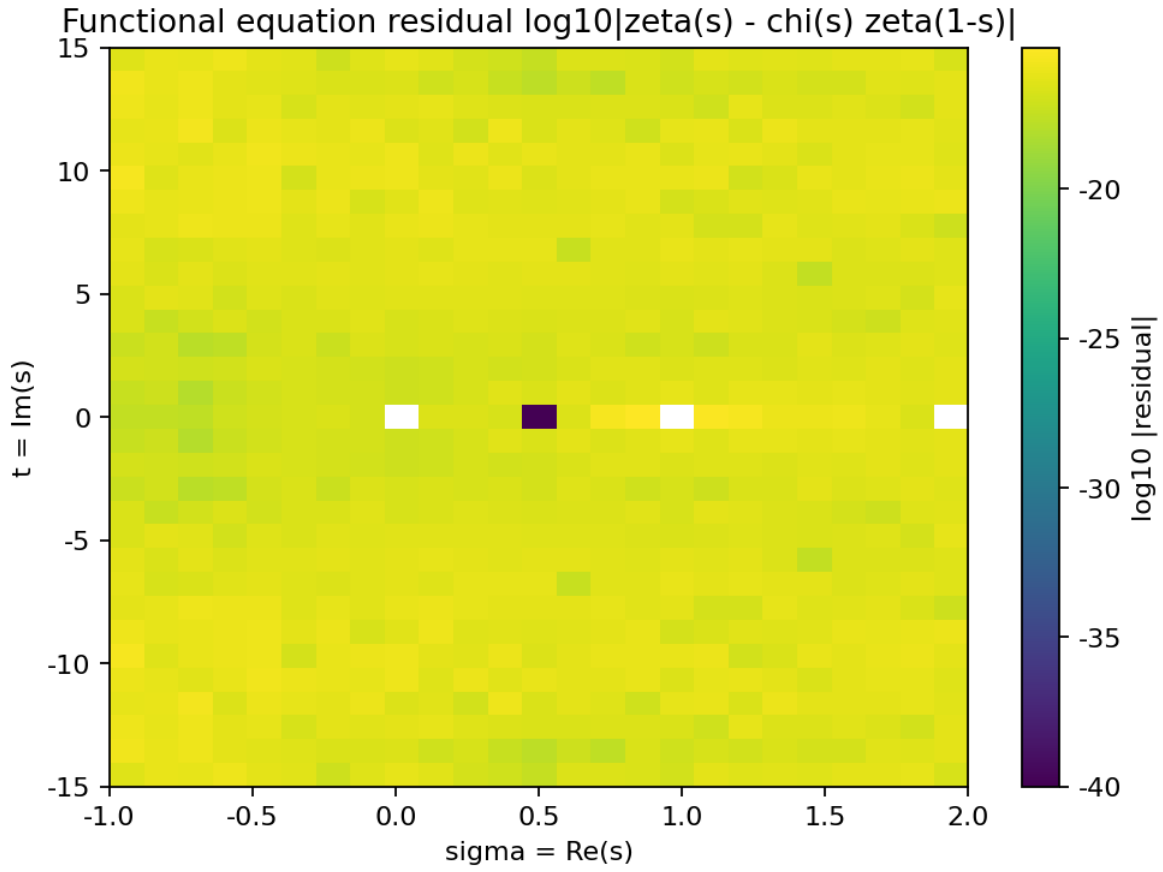


Fig. 9: E090: Functional equation residual heatmap

5.90.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.90.2 What is computed

- Evaluate the functional-equation residual (R(s)=\zeta(s)-\chi(s),\zeta(1-s)) on a small rectangle on the complex plane.
- Visualize `log10(|R(s)| + eps)` as a heatmap.

5.90.3 Notes

- This is a sanity-check style plot: away from poles/zeros and with sufficient precision, the residual should be small.
- Parameters control precision and grid size.

5.90.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We visualize log10|zeta(s) - chi(s) zeta(1-s)| on a coarse grid. Away from the pole at s=1, the residual should be close to numerical precision.

params.json (snapshot)


```
{
  "mp_dps": 70,
  "n_sigma": 25,
  "n_t": 31,
  "sigma_max": 2.0,
  "sigma_min": -1.0,
  "t_max": 15.0,
  "t_min": -15.0
}
```


5.90.5 References

- See the zeta / Dirichlet-series references in `refs.bib`.

5.90.6 Related experiments

- *E089: log| (s)| heatmap* (E089: log| (s)| heatmap)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E085: Dirichlet eta acceleration for (s)* (E085: Dirichlet eta acceleration for (s))

5.91 E091: Partial Euler products on the critical line

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `dirichlet-series`, `numerics`

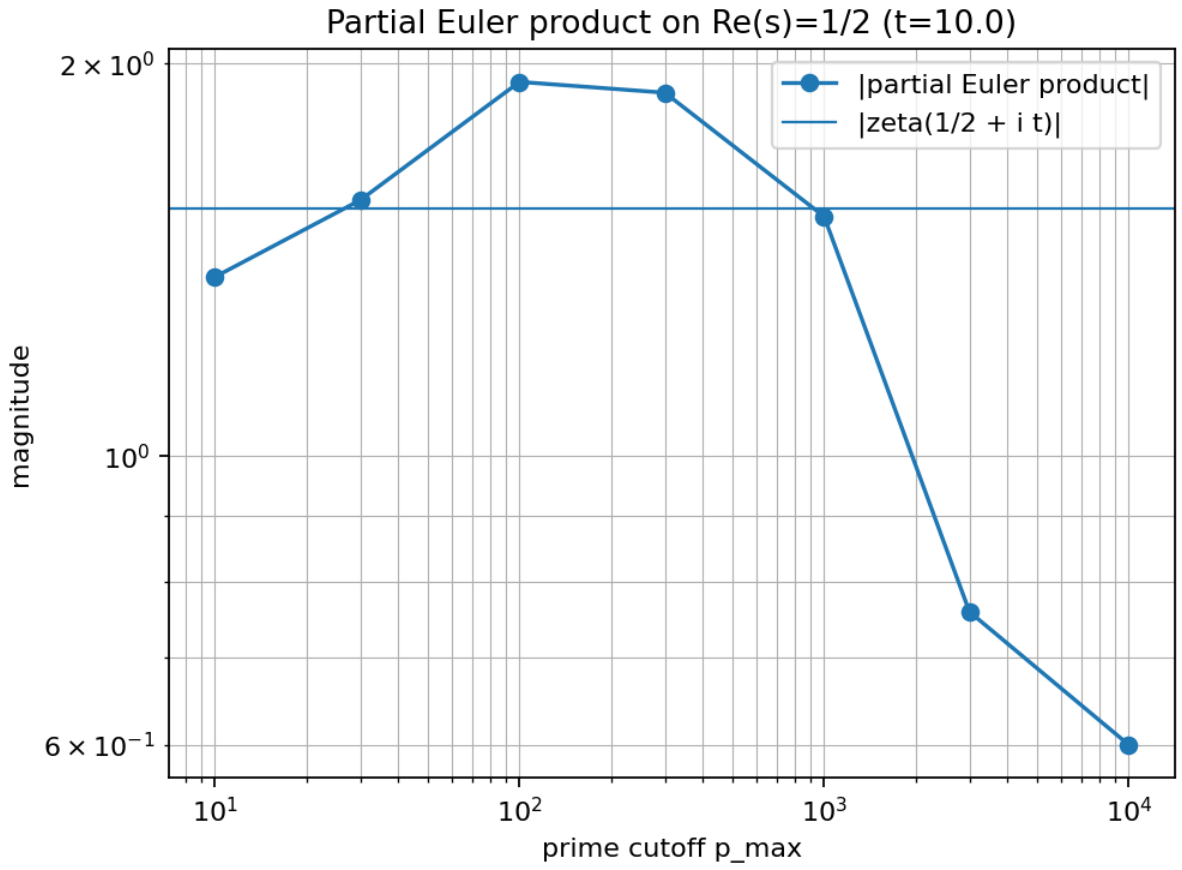


Fig. 10: E091: Partial Euler products on the critical line

5.91.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.91.2 What is computed

- Compare (\zeta(1/2+it)) to partial Euler products (\prod_{p\le P} (1-p^{-s})^{-1}) at a fixed `t` while increasing the prime cutoff `P`.
- Plot the mismatch to illustrate non-convergence on the critical line.

5.91.3 Notes

- The Euler product is only convergent for (\Re(s)>1); this experiment visualizes the failure mode at (\Re(s)=1/2).

5.91.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We compare zeta(1/2 + i t) (t=10.0) to partial Euler products truncated at primes <= p_max.

The Euler product does not converge on the critical line, so the partial products do not stabilize as p_max grows (in contrast to the Re(s) > 1 case).

params.json (snapshot)


```
{
  "mp_dps": 70,
  "prime_cutoffs": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000
  ],
  "t": 10.0
}
```


5.91.5 References

- See the zeta / Dirichlet-series references in `refs.bib`.

5.91.6 Related experiments

- *E114: via : stability map on the critical line* (E114: via : stability map on the critical line)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)

5.92 E092: 1/ζ(s) via the Möbius Dirichlet series

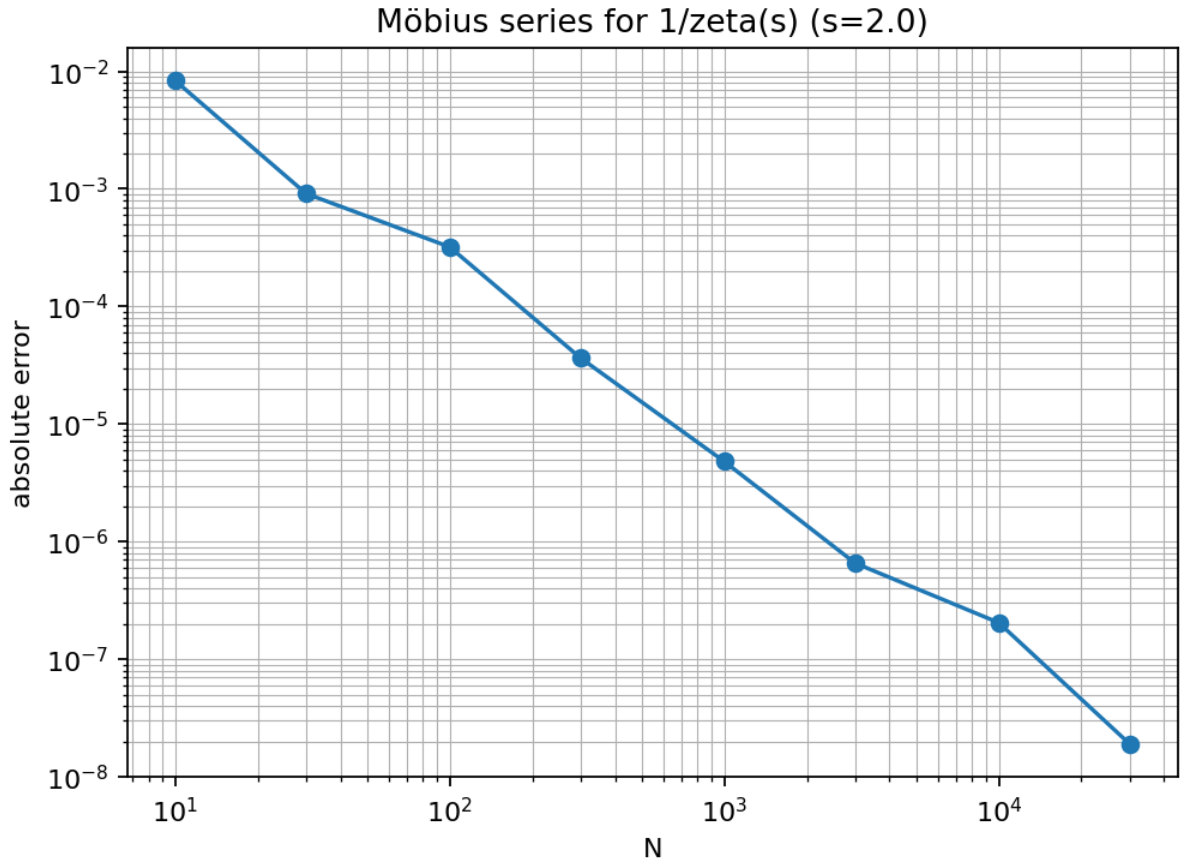


Fig. 11: E092: 1/ (s) via the Möbius Dirichlet series

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `mobius`, `dirichlet-series`, `numerics`

5.92.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.92.2 What is computed

- For (\Re(s)>1), compare (1/\zeta(s)) to the partial sums (\sum_{n\le N} \mu(n)/n^s).
- Plot convergence as `N` increases, and highlight dependence on `Re(s)`.

5.92.3 Notes

- This is the classic identity (\sum_{n\ge1} \mu(n)/n^s = 1/\zeta(s)) (absolute convergence for (\Re(s)>1)).

5.92.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We approximate 1/zeta(s) via the Dirichlet series sum mu(n)/n^s, which converges for Re(s) > 1.

params.json (snapshot)


```
{
  "mp_dps": 70,
  "n_values": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000,
    30000
  ],
  "s": 2.0
}
```


5.92.5 References

- See the zeta / Dirichlet-series references in `refs.bib`.

5.92.6 Related experiments

- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)

5.93 E093: −ζ′(s)/ζ(s) via the von Mangoldt series

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `dirichlet-series`, `numerics`

5.93.1 Highlights

- Focused numeric experiment with a single main figure.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.93.2 What is computed

- For (\Re(s)>1), compare (-\zeta'(s)/\zeta(s)) to the partial sums (\sum_{n\le N} \Lambda(n)/n^s).
- Plot the truncation error as a function of `N`.

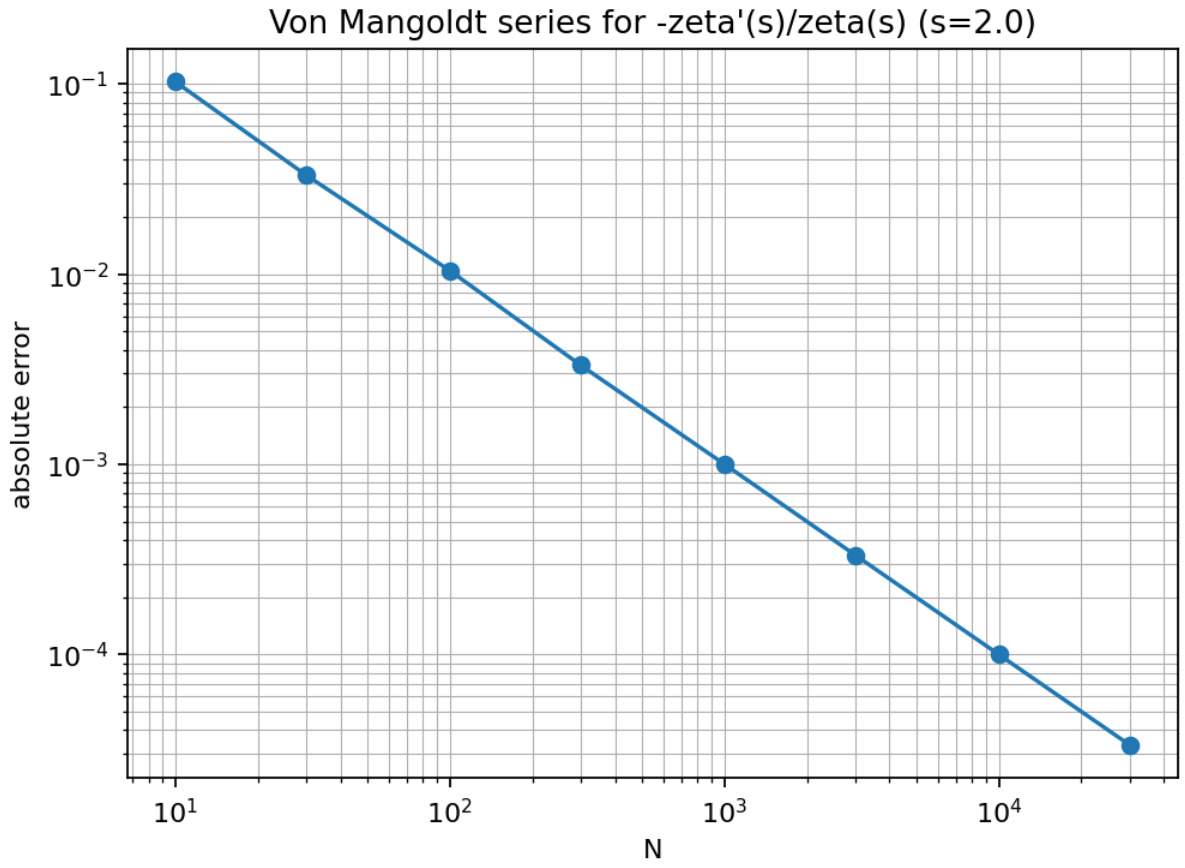


Fig. 12: E093: − (s)/ (s) via the von Mangoldt series

5.93.3 Notes

- Uses the von Mangoldt function (\Lambda(n)); the series converges absolutely for (\Re(s)>1).

5.93.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

We approximate -zeta'(s)/zeta(s) by partial sums of the von Mangoldt Dirichlet series.

params.json (snapshot)


```
{
  "mp_dps": 80,
  "n_values": [
    10,
    30,
    100,
    300,
    1000,
    3000,
    10000,
    30000
  ],
  "s": 2.0
}
```


5.93.5 References

- See the zeta / Dirichlet-series references in `refs.bib`.

5.93.6 Related experiments

- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)

5.94 E094: ω(n) vs. Ω(n): Erdős–Kac normalization

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `omega`, `arithmetic-functions`, `numerics`

5.94.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.94.2 What is computed

- Compare (n) and Ω(n) distributions and their Erdős–Kac normalizations.

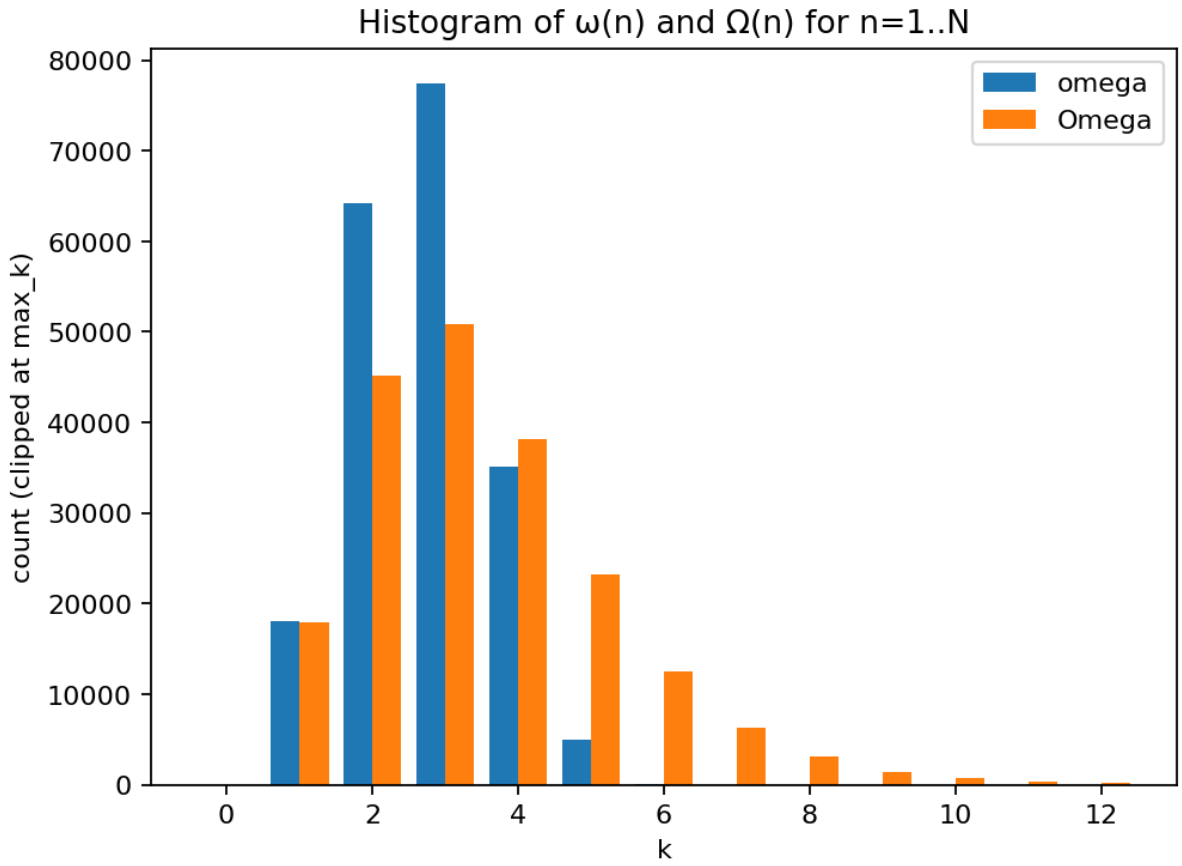


Fig. 13: E094: (n) vs. Ω(n): Erdős–Kac normalization

5.94.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.94.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.94.5 Artifacts

- figures/fig_01_omega_vs_bigomega_hist.png
- params.json
- report.md

Notes

- Mean (n) ≈ 2.725, Var (n) ≈ 0.888
- Mean Ω(n) ≈ 3.497, Var Ω(n) ≈ 3.080
- log log N ≈ 2.502 (Erdős–Kac heuristic scale)

params.json (snapshot)

```
{
  "max_k": 12,
  "n_max": 200000
}
```

5.94.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.94.7 Related experiments

- *E057: Erdős–Kac in practice* (Erdős–Kac in practice)
- *E095: Squarefree filter: (n)=Ω(n) when (n)≠0* (E095: Squarefree filter: (n)=Ω(n) when (n)≠0)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)

5.95 E095: Squarefree filter: ω(n)=Ω(n) when μ(n)≠0

Tags: `number-theory`, `model-checking`, `visualization`, `mobius`, `omega`, `arithmetic-functions`

5.95.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

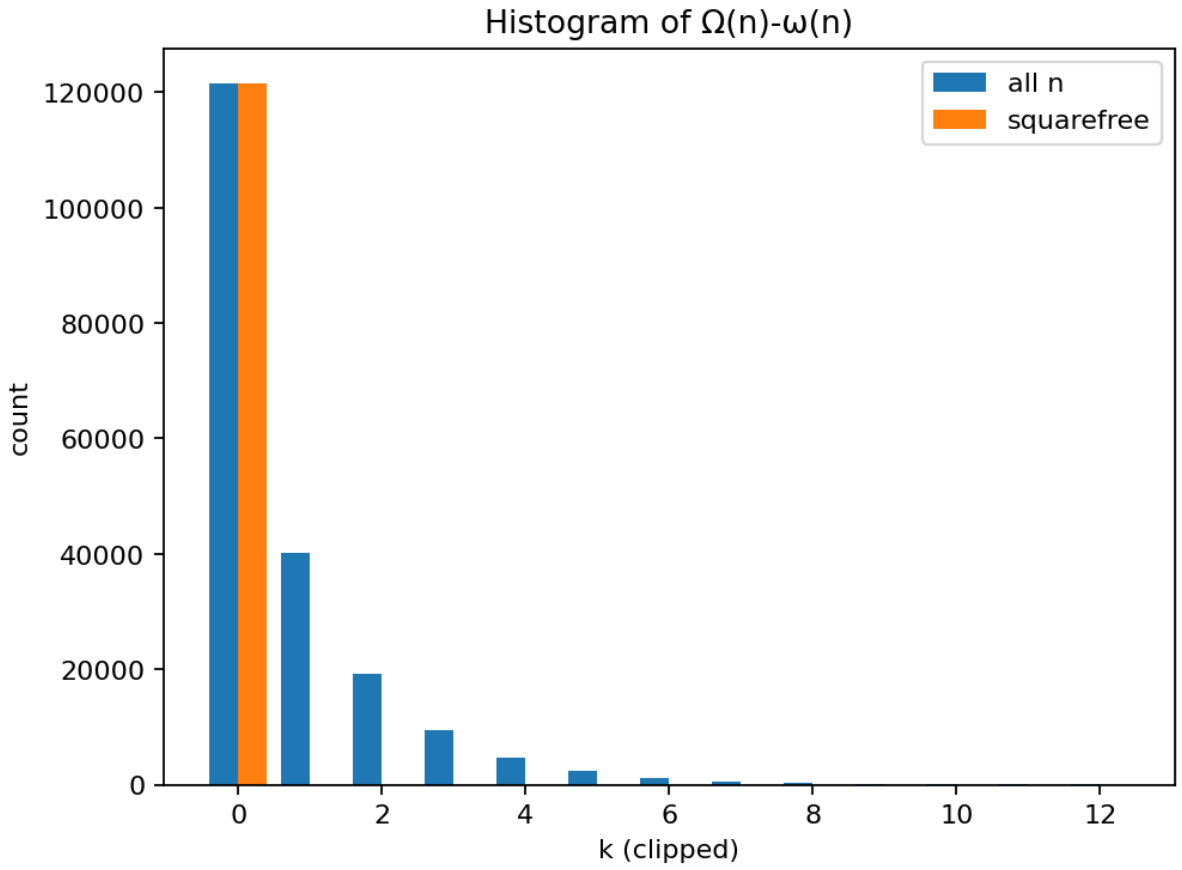


Fig. 14: E095: Squarefree filter: (n)=Ω(n) when (n)≠0

5.95.2 What is computed

- Use (n) as a squarefree indicator and verify (n)=Ω(n) on that subset.

5.95.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.95.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.95.5 Artifacts

- figures/fig_01_bigomega_minus_omega.png
- params.json
- report.md

Notes

- Fraction with Ω(n)= (n) on all integers: 0.6079
- Fraction with Ω(n)= (n) on squarefree integers: 1.0000 (should be 1.0)
- Ω(n)- (n) counts repeated prime powers (p^k with k≥2).

params.json (snapshot)

```
{
  "max_k": 12,
  "n_max": 200000
}
```

5.95.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.95.7 Related experiments

- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E055: Mertens function walk* (Mertens function walk)
- *E094: (n) vs. Ω(n): Erdős–Kac normalization* (E094: (n) vs. Ω(n): Erdős–Kac normalization)
- *E057: Erdős–Kac in practice* (Erdős–Kac in practice)
- *E121: Möbius inversion as convolution undo* (E121: Möbius inversion as convolution undo)

5.96 E096: Record-holders for τ(n)

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `divisor-function`, `arithmetic-functions`

5.96.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

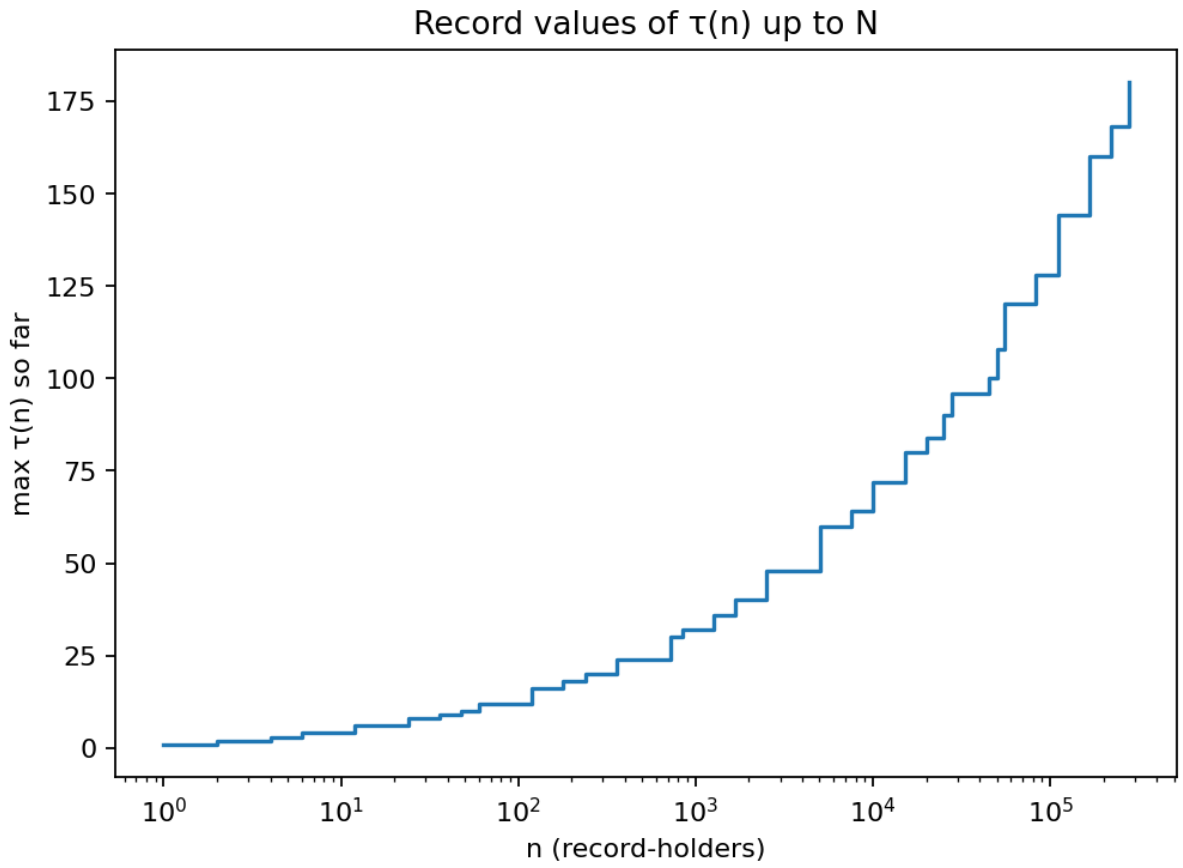


Fig. 15: E096: Record-holders for (n)

5.96.2 What is computed

- Track the running maxima of (n) up to N and visualize the growth of record values.

5.96.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.96.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.96.5 Artifacts

- figures/fig_01_tau_records.png
- params.json
- report.md

Notes

- Number of record-holders found: 33
- Last 10 record-holders (n, (n)): [(25200, 90), (27720, 96), (45360, 100), (50400, 108), (55440, 120), (83160, 128), (110880, 144), (166320, 160), (221760, 168), (277200, 180)]
- Record-holders are built from small primes with nonincreasing exponents (highly composite flavor).

params.json (snapshot)

```
{
  "max_points": 150,
  "n_max": 300000
}
```

5.96.6 References

- See the arithmetic-functions background pages in `docs/background/`.
- See the perfect-numbers and divisor-functions background pages in `docs/background/`.

5.96.7 Related experiments

- *E058: Divisor-count record highs* (Divisor-count record highs)
- *E059: Abundancy index landscape* (Abundancy index landscape)
- *E027: Record prime gaps vs. log^2 heuristic* (Record prime gaps vs. log^2 heuristic)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)

5.97 E097: σ(n)/n landscape: deficient, perfect, abundant

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `sigma`, `perfect`, `classification`

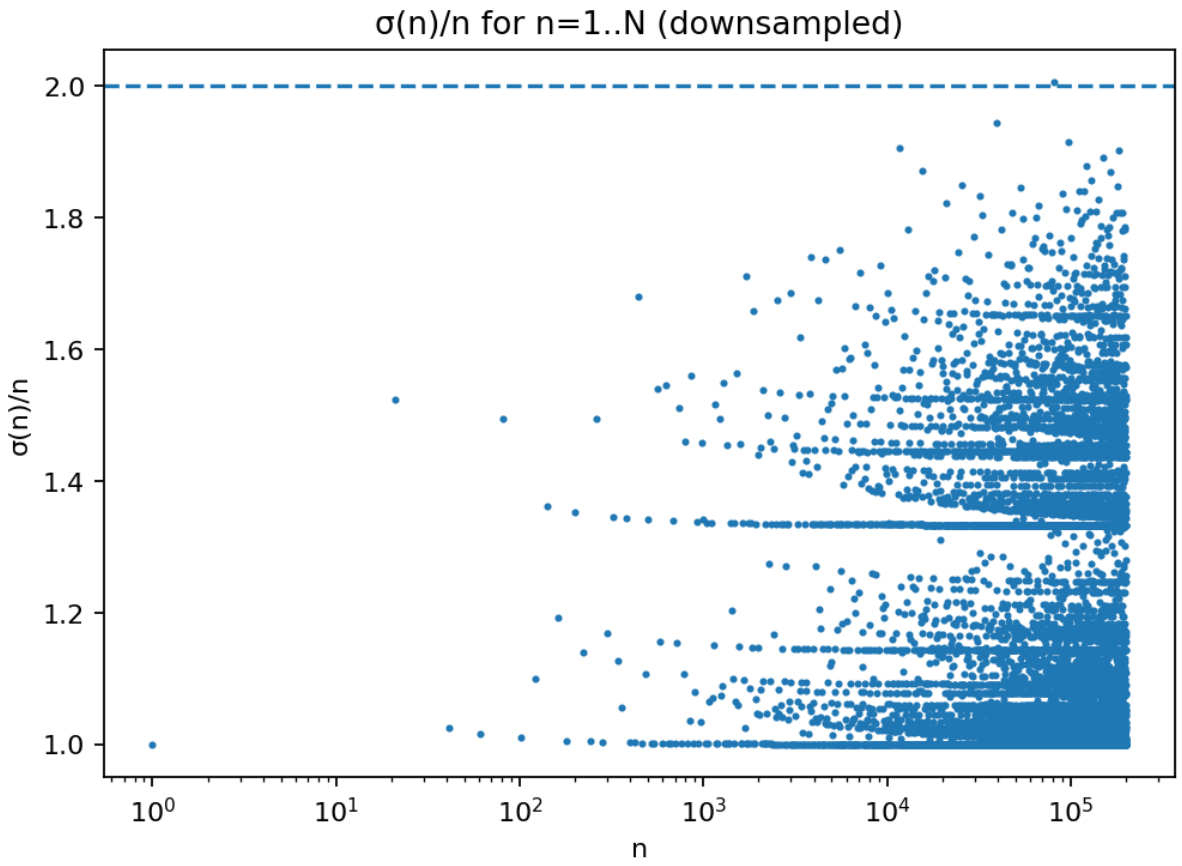


Fig. 16: E097: (n)/n landscape: deficient, perfect, abundant

5.97.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.97.2 What is computed

- Visualize (n)/n and classify integers using (n) compared to 2n.

5.97.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.97.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.97.5 Artifacts

- figures/fig_01_sigma_over_n_scatter.png
- params.json
- report.md

Notes

- Fraction abundant ((n)>2n) up to N: 0.2474
- Number of perfect numbers detected up to N: 7
- First perfect numbers in range (if any): [6, 28, 496, 8128, 65536, 130304, 131072]

params.json (snapshot)

```
{
  "n_max": 200000,
  "stride": 20
}
```

5.97.6 References

- See the arithmetic-functions background pages in `docs/background/`.
- See the perfect-numbers and divisor-functions background pages in `docs/background/`.

5.97.7 Related experiments

- *E059: Abundancy index landscape* (Abundancy index landscape)
- *E098: Maximizers of (n)/n^ across* (E098: Maximizers of (n)/n^ across)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)
- *E003: Abundancy Index Landscape* (Abundancy Index Landscape)
- *E010: Even perfect numbers from Mersenne primes* (Even perfect numbers from Mersenne primes)

5.98 E098: Maximizers of σ(n)/n^α across α

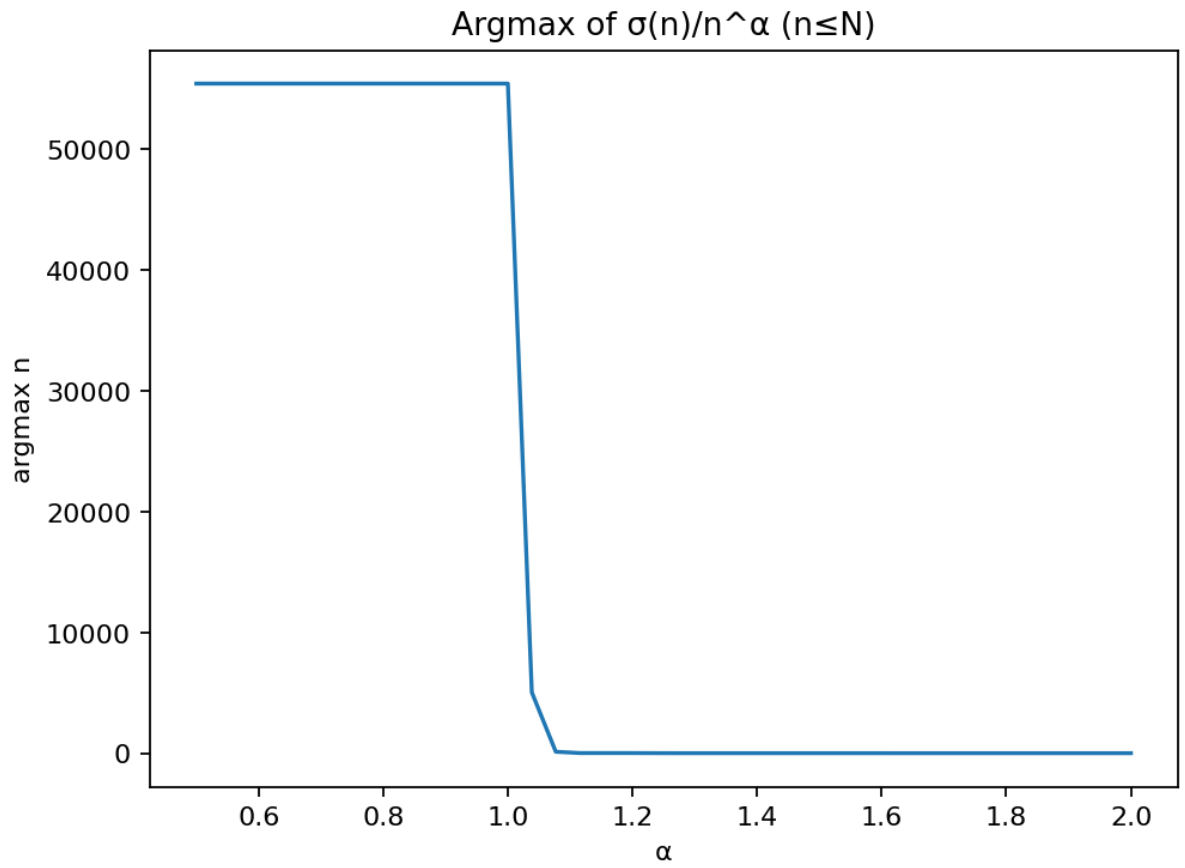


Fig. 17: E098: Maximizers of (n)/n^ across

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `sigma`, `optimization`

5.98.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.98.2 What is computed

- For on a grid, find n≤N maximizing (n)/n^ and show regime changes.

5.98.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.98.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.98.5 Artifacts

- figures/fig_01_argmax_vs_alpha.png
- params.json
- report.md

Notes

- Unique maximizers encountered: 7
- First transitions (alpha, argmax n): [(0.5, 55440), (1.0384615384615385, 5040), (1.076923076923077, 120), (1.1153846153846154, 12), (1.2307692307692308, 6), (1.2692307692307692, 2), (1.6153846153846154, 1)]
- This is a small-N proxy for highly/colossally abundant phenomena.

params.json (snapshot)

```
{
  "alpha_max": 2.0,
  "alpha_min": 0.5,
  "alpha_steps": 40,
  "n_max": 60000
}
```

5.98.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.98.7 Related experiments

- *E004: Computing \sigma(n) at Scale — Sieve vs. Factorization* (Computing $\sigma(n)$ at Scale — Sieve vs. Factorization)
- *E059: Abundancy index landscape* (Abundancy index landscape)
- *E097: (n)/n landscape: deficient, perfect, abundant* (E097: (n)/n landscape: deficient, perfect, abundant)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)
- *E002: Even Perfect Numbers — Generator and Growth* (Even Perfect Numbers — Generator and Growth)

5.99 E099: Jordan totients J_k: identities and ratios

Tags: `number-theory`, `model-checking`, `visualization`, `totient`, `arithmetic-functions`

5.99.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

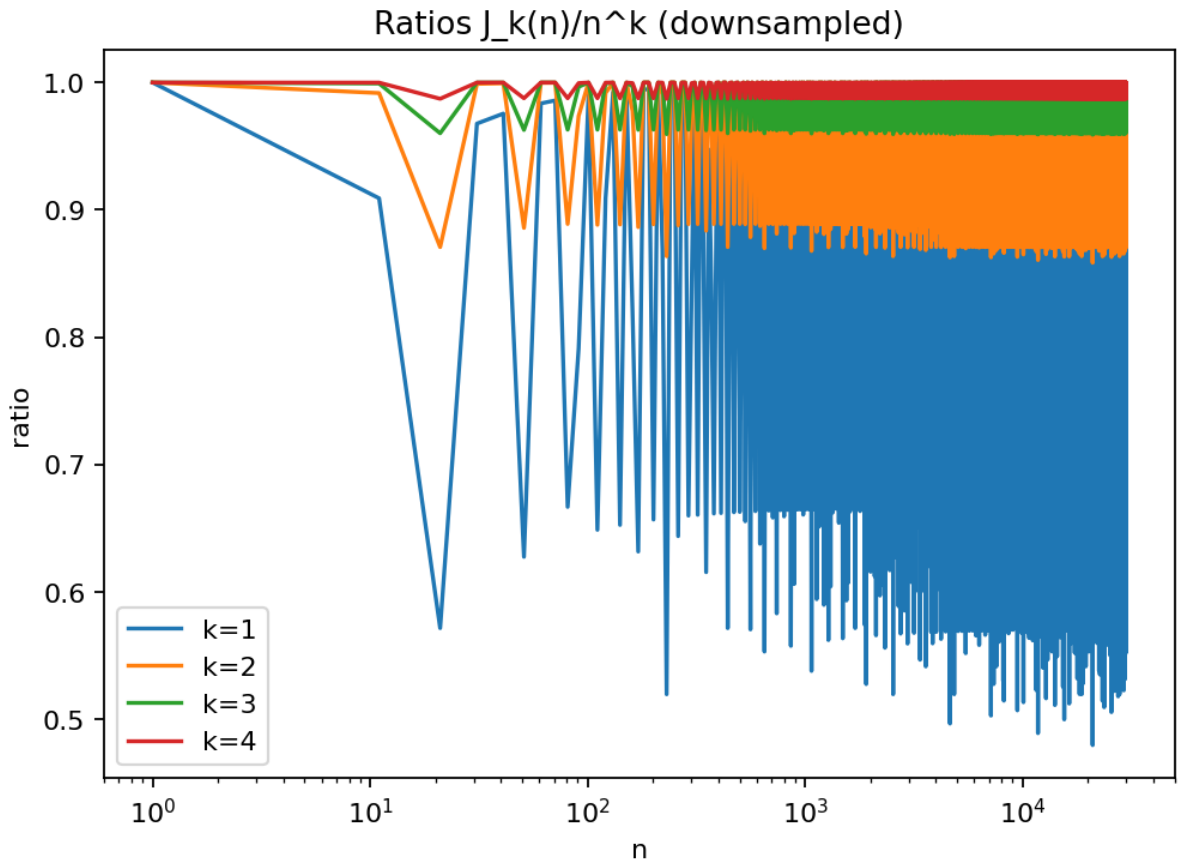


Fig. 18: E099: Jordan totients J_k: identities and ratios

5.99.2 What is computed

- Compute J_k(n), verify J_1= , and visualize ratios J_k(n)/n^k.

5.99.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.99.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.99.5 Artifacts

- figures/fig_01_jordan_ratios.png
- params.json
- report.md

Notes

- Mean ratios (downsampled) for k=1..k_max: [0.844324, 0.958415, 0.987113, 0.995815]
- Max |J_1(n)-phi(n)| on a 100-point sample: 0
- J_k(n) = n^k _{p|n}(1-1/p^k) (multiplicative).

params.json (snapshot)

```
{
  "k_max": 4,
  "n_max": 30000,
  "stride": 10
}
```

5.99.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.99.7 Related experiments

- *E060: Jordan totients* (Jordan totients)
- *E101: Reduced residues modulo q: concrete structure* (E101: Reduced residues modulo q: concrete structure)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E062: Carmichael (n) vs. (n)* (Carmichael (n) vs. (n))

5.100 E100: Carmichael λ(n) vs. Euler φ(n)

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `carmichael-lambda`, `totient`, `arithmetic-functions`

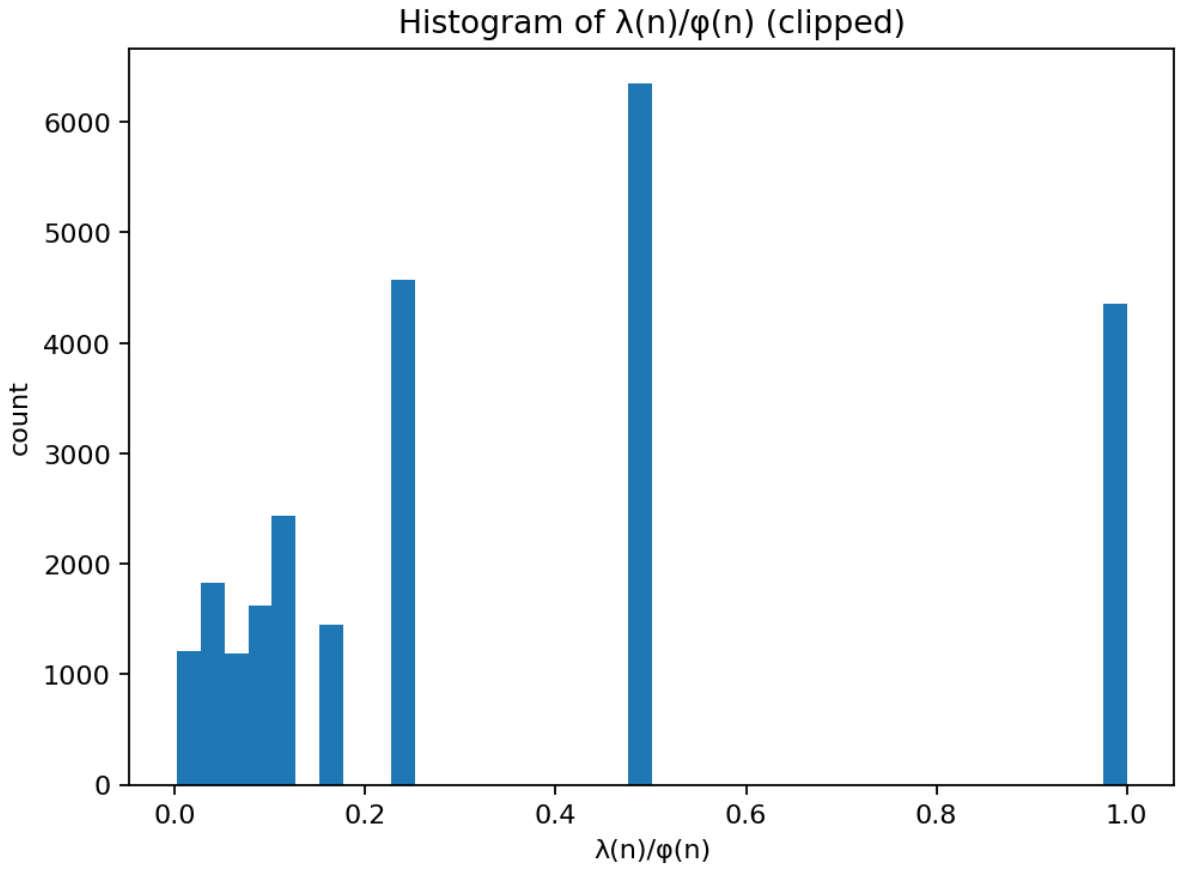


Fig. 19: E100: Carmichael (n) vs. Euler (n)

5.100.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.100.2 What is computed

- Compare (n) and (n) and visualize (n)/ (n) over a finite range.

5.100.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.100.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.100.5 Artifacts

- figures/fig_01_lambda_over_phi_hist.png
- params.json
- report.md

Notes

- min (n)/ (n): 0.002315
- max (n)/ (n): 1.000000
- (n) is the exponent of (Z/nZ)^× and often much smaller than (n).

params.json (snapshot)

```
{
  "bins": 40,
  "max_ratio": 1.0,
  "n_max": 25000
}
```

5.100.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.100.7 Related experiments

- *E062: Carmichael (n) vs. (n)* (Carmichael (n) vs. (n))
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E060: Jordan totients* (Jordan totients)
- *E099: Jordan totients J_k: identities and ratios* (E099: Jordan totients J_k: identities and ratios)

5.101 E101: Reduced residues modulo q: concrete structure

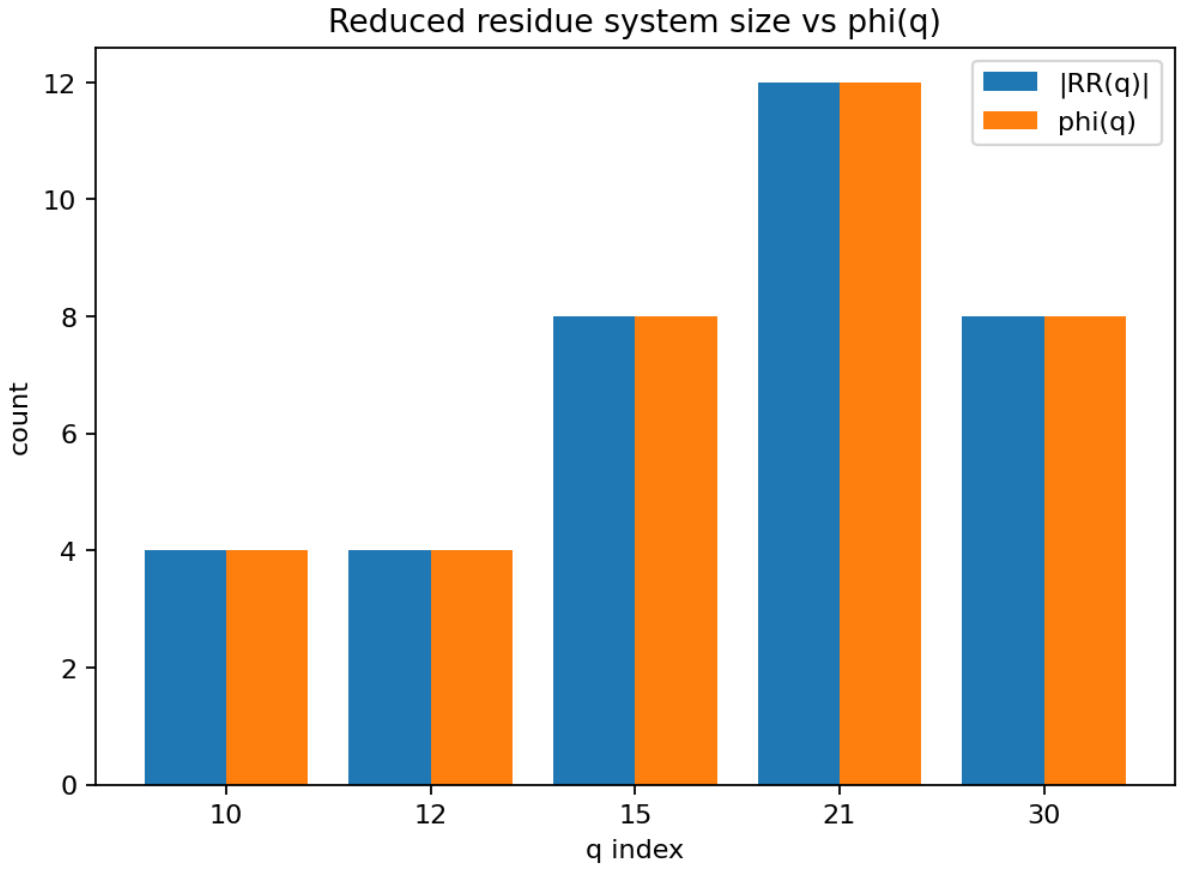


Fig. 20: E101: Reduced residues modulo q: concrete structure

Tags: `number-theory`, `dirichlet-characters`, `model-checking`, `visualization`, `totient`

5.101.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.101.2 What is computed

- List reduced residues mod q, verify count= (q), and summarize simple structure checks.

5.101.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.101.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.101.5 Artifacts

- figures/fig_01_reduced_residues_sizes.png
- params.json
- report.md

Notes

- Checked q values: [10, 12, 15, 21, 30]
- sizes match phi(q): True
- Example prefixes of RR(q): {10: [1, 3, 7, 9], 12: [1, 5, 7, 11], 15: [1, 2, 4, 7, 8, 11, 13, 14], 21: [1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20], 30: [1, 7, 11, 13, 17, 19, 23, 29]}

params.json (snapshot)


```
{
  "q_values": [
    10,
    12,
    15,
    21,
    30
  ]
}
```


5.101.6 References

- See the arithmetic-functions background pages in `docs/background/`.
- See `docs/background/dirichlet-characters.md`.

5.101.7 Related experiments

- *E099: Jordan totients J_k: identities and ratios* (E099: Jordan totients J_k: identities and ratios)
- *E108: Orthogonality heatmap for characters* (E108: Orthogonality heatmap for characters)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E060: Jordan totients* (Jordan totients)

5.102 E102: Dirichlet convolution identity zoo

Tags: `number-theory`, `model-checking`, `visualization`, `dirichlet-convolution`, `multiplicative`, `arithmetic-functions`

5.102.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

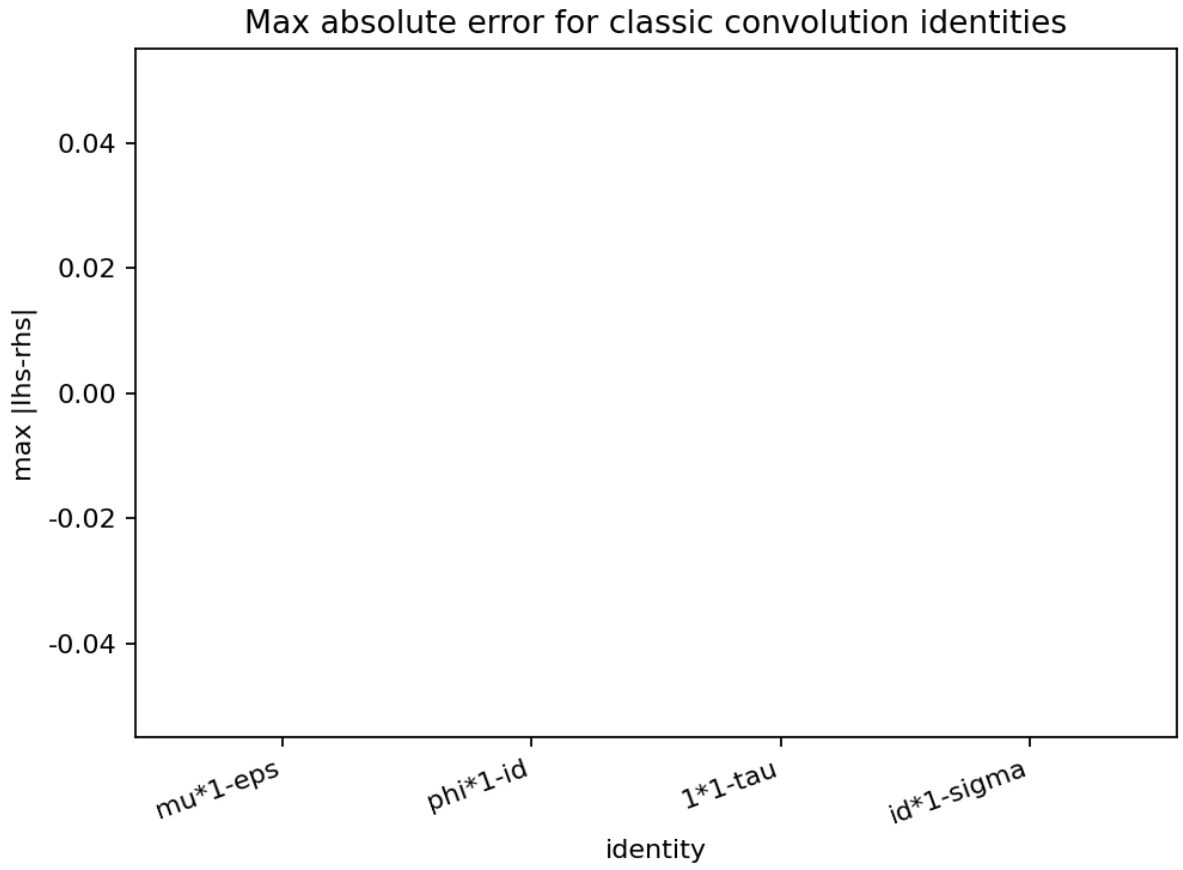


Fig. 21: E102: Dirichlet convolution identity zoo

5.102.2 What is computed

- Verify classic convolution identities numerically on 1..N (*1=* , 1=id, 1*1=).

5.102.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.102.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.102.5 Artifacts

- figures/fig_01_convolution_identity_errors.png
- params.json
- report.md

Notes

- Max errors: {'mu*1-eps': 0, 'phi*1-id': 0, '1*1-tau': 0, 'id*1-sigma': 0}
- All should be exactly 0 for these sieve-based implementations.

params.json (snapshot)

```
{
  "n_max": 20000
}
```

5.102.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.102.7 Related experiments

- *E063: Dirichlet convolution playground* (Dirichlet convolution playground)
- *E121: Möbius inversion as convolution undo* (E121: Möbius inversion as convolution undo)
- *E108: Orthogonality heatmap for characters* (E108: Orthogonality heatmap for characters)
- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E052: Totient ratio landscape* (Totient ratio landscape)

5.103 E103: Chebyshev ψ(x): prime powers drive jumps

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `mangoldt`, `explicit-formula`

5.103.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

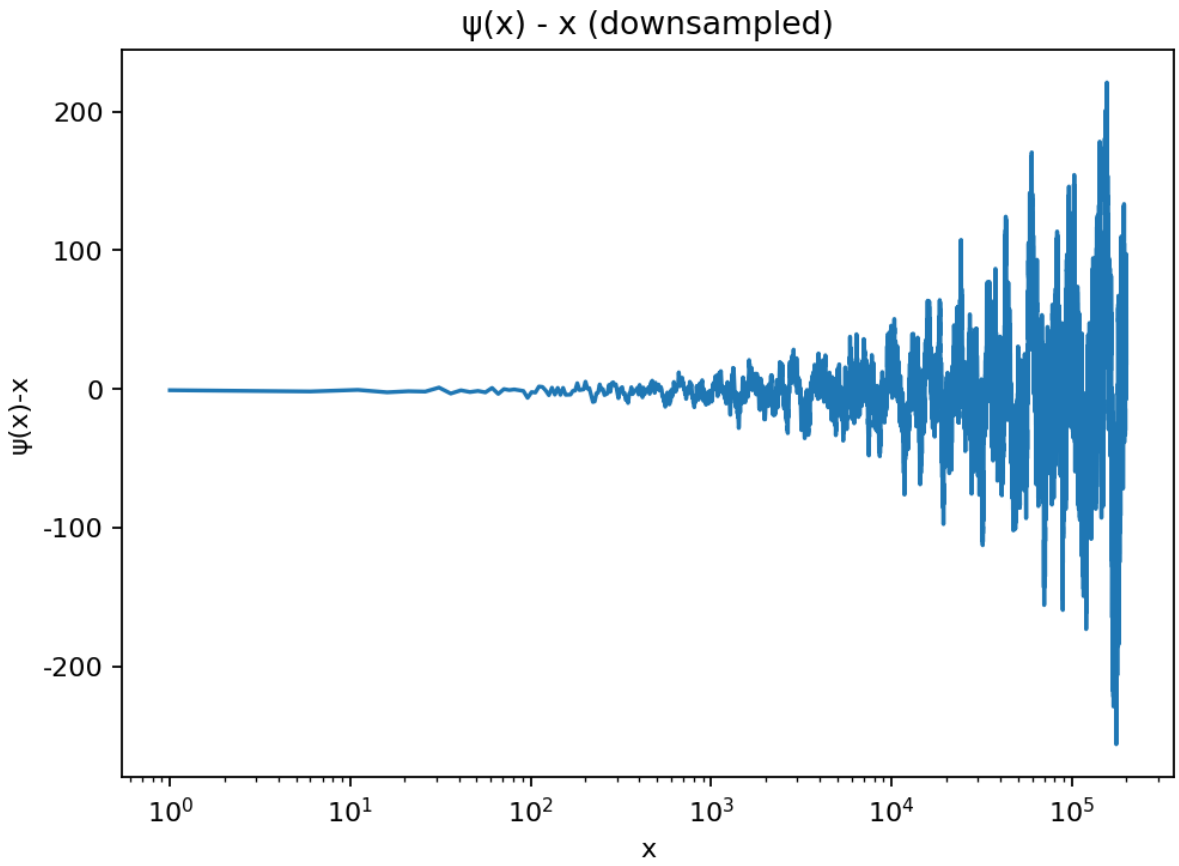


Fig. 22: E103: Chebyshev (x): prime powers drive jumps

5.103.2 What is computed

- Plot (x) and highlight jump contributions coming from prime powers.

5.103.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.103.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.103.5 Artifacts

- figures/fig_01_psi_minus_x.png
- params.json
- report.md

Notes

- Nonzero Λ(n) terms up to N (prime powers): 18120
- (x) is the summatory von Mangoldt function and jumps at prime powers.

params.json (snapshot)

```
{
  "n_max": 200000,
  "stride": 5
}
```

5.103.6 References

- See the relevant references in `refs.bib`.

5.103.7 Related experiments

- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)
- *E104: von Mangoldt Λ(n): support and statistics* (E104: von Mangoldt Λ(n): support and statistics)
- *E076: Chebyshev (x;q,a): weighted prime counts in progressions.* (Chebyshev (x;q,a): weighted prime counts in progressions.)
- *E080: Chebyshev bias: leader fraction vs. x.* (Chebyshev bias: leader fraction vs. x.)
- *E118: Chebyshev bias: lead-time statistics* (E118: Chebyshev bias: lead-time statistics)

5.104 E104: von Mangoldt Λ(n): support and statistics

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `mangoldt`, `arithmetic-functions`

5.104.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

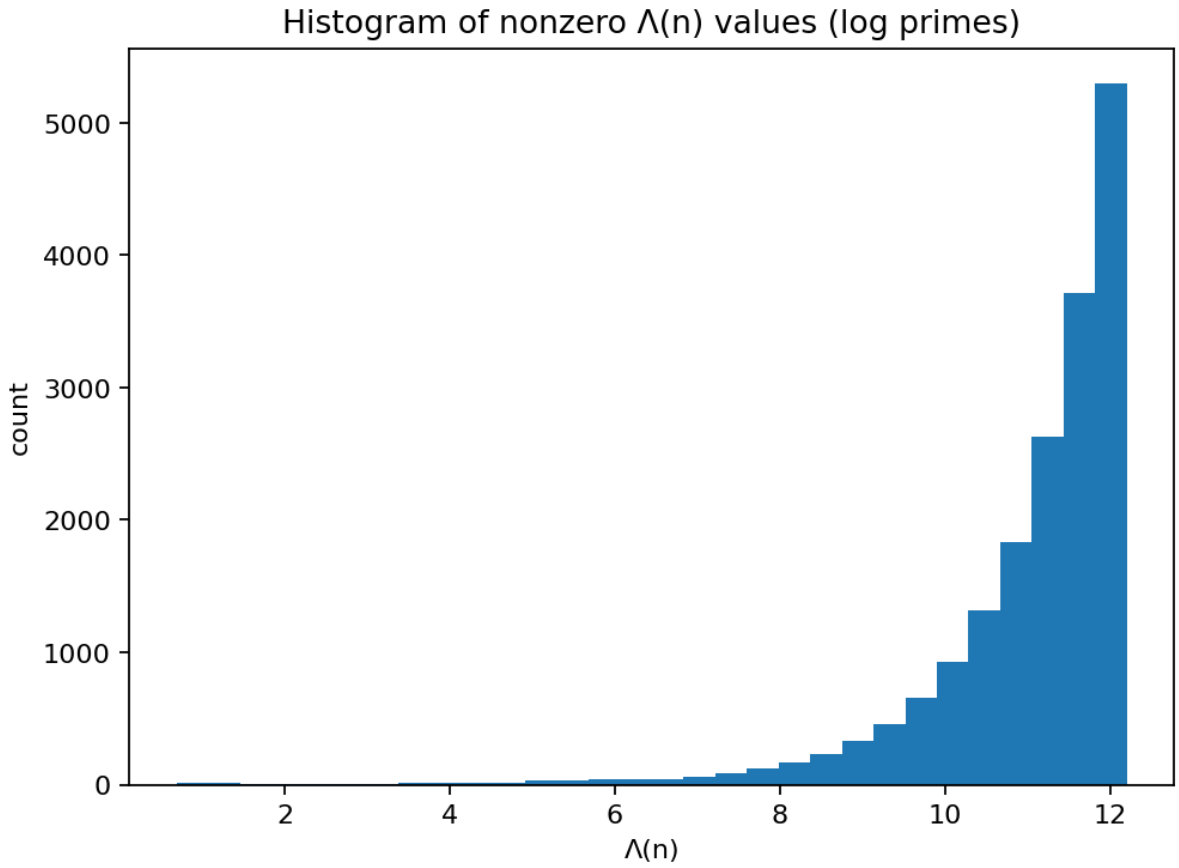


Fig. 23: E104: von Mangoldt Λ(n): support and statistics

5.104.2 What is computed

- Visualize where Λ(n) is nonzero (prime powers) and summarize basic statistics.

5.104.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.104.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.104.5 Artifacts

- figures/fig_01_lambda_hist.png
- params.json
- report.md

Notes

- Nonzero density: 0.090600
- Unique Λ(n) values are logs of primes; sample: [0.69314718, 1.09861229, 1.60943791, 1.94591015, 2.39789527, 2.56494936, 2.83321334, 2.94443898, 3.13549422, 3.36729583]

params.json (snapshot)

```
{
  "bins": 30,
  "n_max": 200000
}
```

5.104.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.104.7 Related experiments

- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)
- *E103: Chebyshev (x): prime powers drive jumps* (E103: Chebyshev (x): prime powers drive jumps)
- *E118: Chebyshev bias: lead-time statistics* (E118: Chebyshev bias: lead-time statistics)
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)

5.105 E105: Mertens M(x): scaling views

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `mobius`, `summatory`, `open-problems`

5.105.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

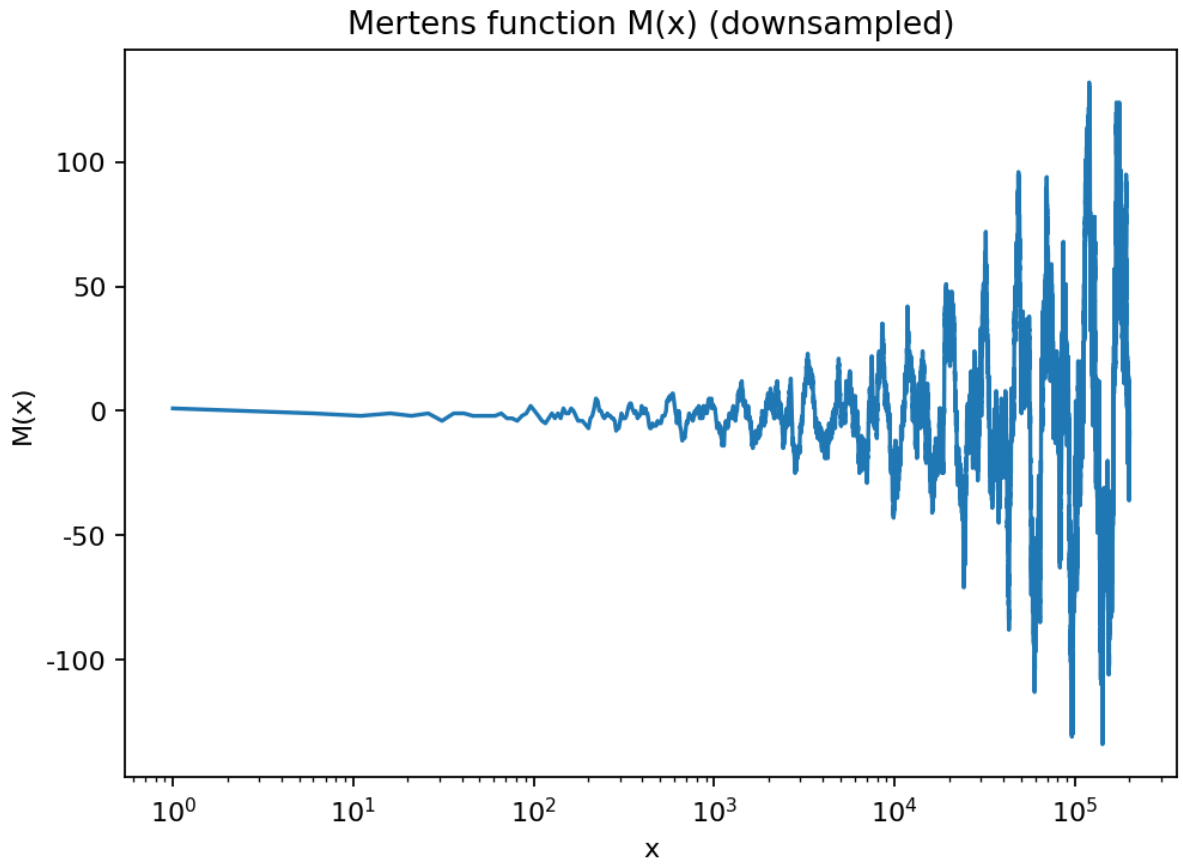


Fig. 24: E105: Mertens M(x): scaling views

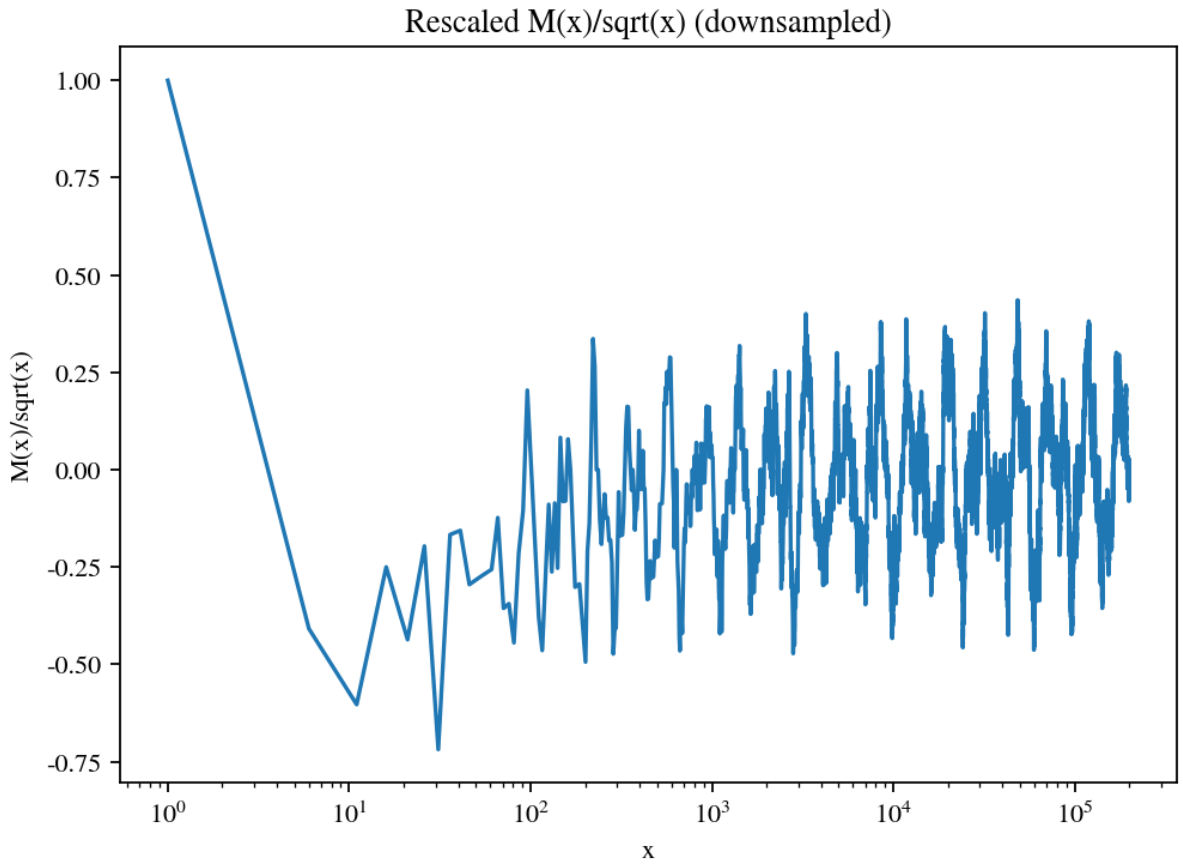


Fig. 25: E105: Mertens M(x): scaling views

5.105.2 What is computed

- Plot M(x)= _{n≤x} (n) and rescalings such as M(x)/√x.

5.105.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.105.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.105.5 Artifacts

- figures/fig_01_mertens_M.png
- figures/fig_02_mertens_rescaled.png
- params.json
- report.md

Notes

- max |M(x)| on 1..N: 134.0
- Random-walk-like oscillations appear across scales (visual heuristic).

params.json (snapshot)

```
{
  "n_max": 200000,
  "stride": 5
}
```

5.105.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.105.7 Related experiments

- *E055: Mertens function walk* (Mertens function walk)
- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E056: Liouville vs. Möbius walks* (Liouville vs. Möbius walks)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)
- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)

5.106 E106: Character gallery: real vs. complex

Tags: `number-theory`, `dirichlet-characters`, `visualization`, `multiplicative`

5.106.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

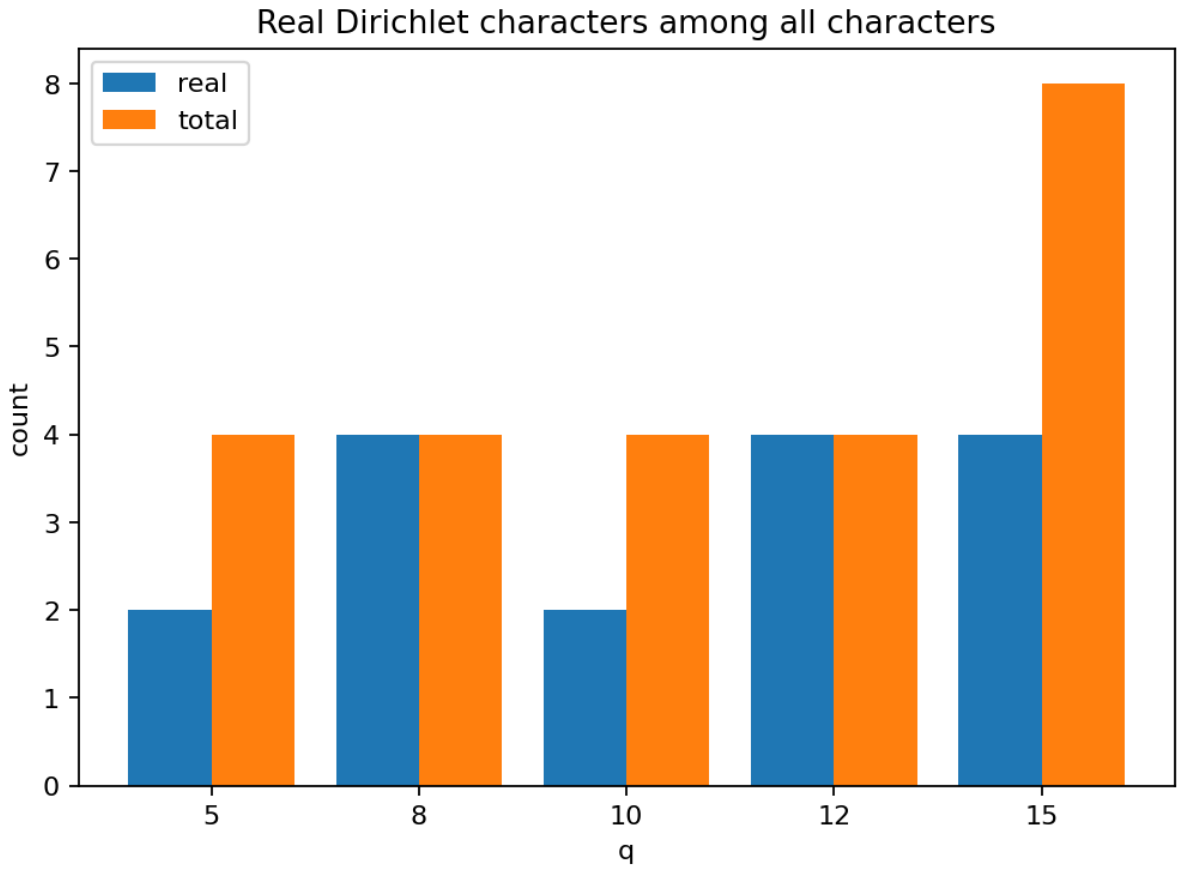


Fig. 26: E106: Character gallery: real vs. complex

5.106.2 What is computed

- Compare value-sets of Dirichlet characters and count real-valued characters.

5.106.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.106.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.106.5 Artifacts

- figures/fig_01_real_character_counts.png
- params.json
- report.md

Notes

- q values: [5, 8, 10, 12, 15]
- real/total: [(5, 2, 4), (8, 4, 4), (10, 2, 4), (12, 4, 4), (15, 4, 8)]
- Real characters correspond to quadratic characters and products thereof.

params.json (snapshot)


```
{
  "q_values": [
    5,
    8,
    10,
    12,
    15
  ]
}
```


5.106.6 References

- See `docs/background/dirichlet-characters.md`.

5.106.7 Related experiments

- *E108: Orthogonality heatmap for characters* (E108: Orthogonality heatmap for characters)
- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E077: Indicator via character orthogonality (sanity check).* (Indicator via character orthogonality (sanity check).)
- *E122: Character averages over primes* (E122: Character averages over primes)

5.107 E107: Conductor: primitive vs. induced characters

Tags: `number-theory`, `dirichlet-characters`, `quantitative-exploration`, `visualization`, `l-functions`

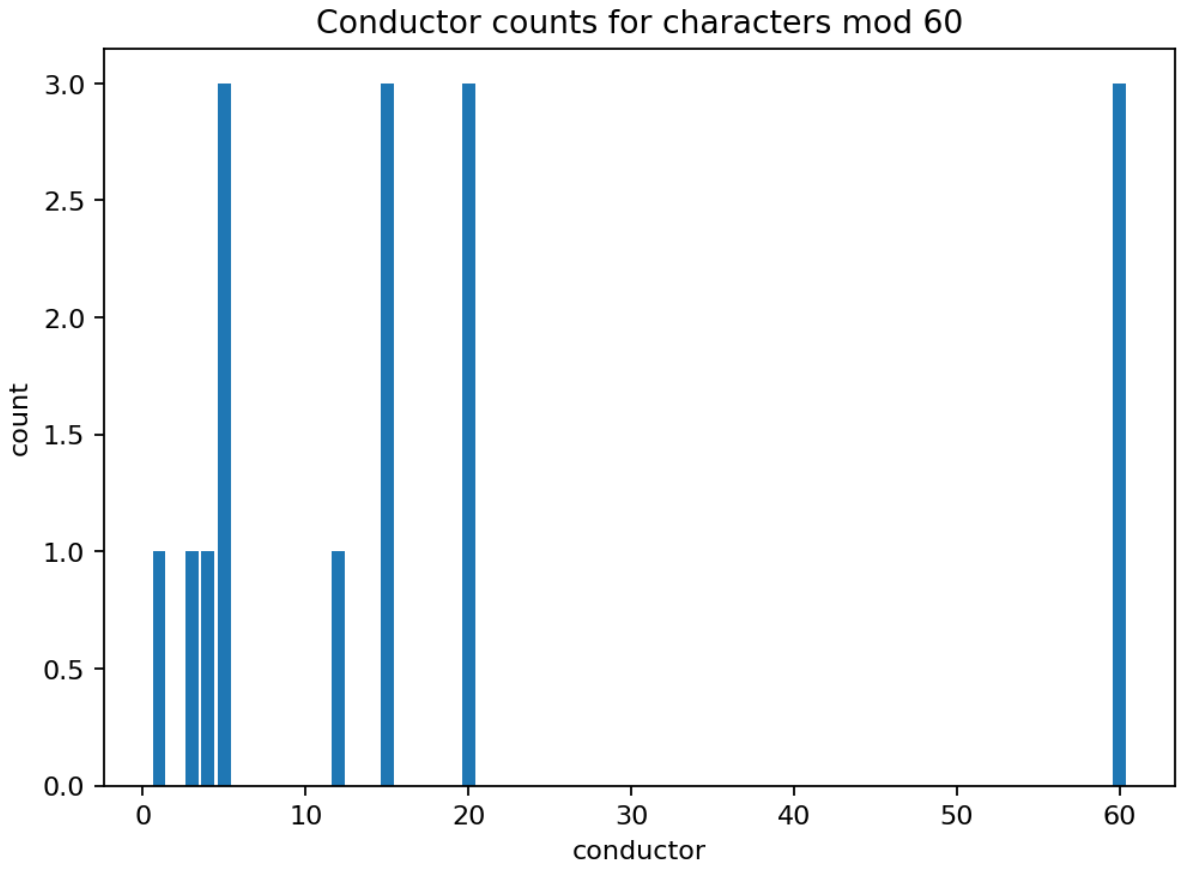


Fig. 27: E107: Conductor: primitive vs. induced characters

5.107.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.107.2 What is computed

- Compute conductors and separate primitive from imprimitive characters.

5.107.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.107.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.107.5 Artifacts

- figures/fig_01_conductor_hist.png
- params.json
- report.md

Notes

- Total characters: 16
- Non-principal characters: 15
- Conductors observed: [1, 3, 4, 5, 12, 15, 20, 60]

params.json (snapshot)

```
{
  "q": 60
}
```

5.107.6 References

- See `docs/background/dirichlet-characters.md`.
- See `docs/background/dirichlet-l-functions.md` and `docs/background/dirichlet-convolution.md`.

5.107.7 Related experiments

- *E079: Primitive vs. imprimitive characters: conductors.* (Primitive vs. imprimitive characters: conductors.)
- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E069: L(1,): slow convergence and smoothing.* (L(1,): slow convergence and smoothing.)
- *E122: Character averages over primes* (E122: Character averages over primes)
- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)

5.108 E108: Orthogonality heatmap for characters

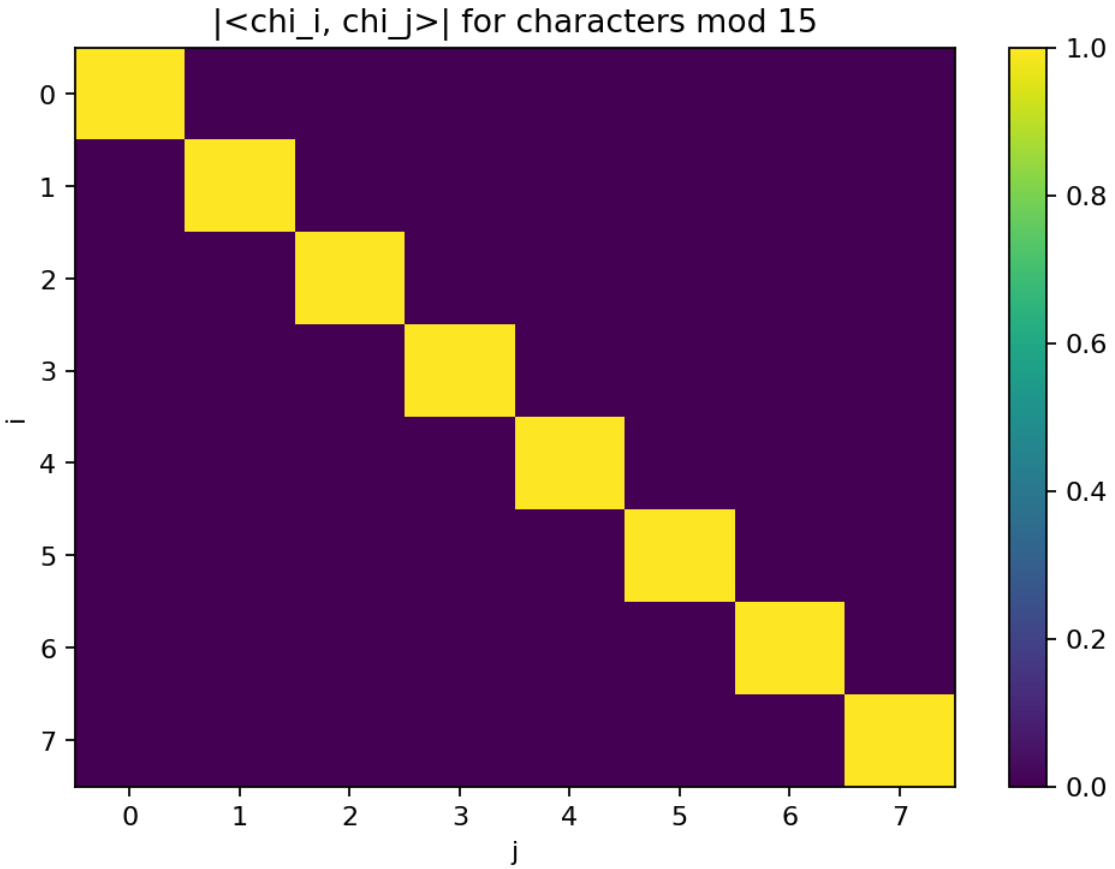


Fig. 28: E108: Orthogonality heatmap for characters

Tags: `number-theory`, `dirichlet-characters`, `model-checking`, `visualization`, `multiplicative`

5.108.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.108.2 What is computed

- Visualize orthogonality relations via inner-product matrices.

5.108.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.108.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.108.5 Artifacts

- figures/fig_01_orthogonality_abs.png
- params.json
- report.md

Notes

- Mean diagonal magnitude: 1.000000
- Mean off-diagonal magnitude: 0.000000
- For a correctly normalized inner product, diagonal entries dominate and off-diagonals are ~0.

params.json (snapshot)

```
{
  "q": 15
}
```

5.108.6 References

- See `docs/background/dirichlet-characters.md`.

5.108.7 Related experiments

- *E065: Orthogonality matrix for Dirichlet characters.* (Orthogonality matrix for Dirichlet characters.)
- *E102: Dirichlet convolution identity zoo* (E102: Dirichlet convolution identity zoo)
- *E121: Möbius inversion as convolution undo* (E121: Möbius inversion as convolution undo)
- *E106: Character gallery: real vs. complex* (E106: Character gallery: real vs. complex)
- *E077: Indicator via character orthogonality (sanity check).* (Indicator via character orthogonality (sanity check).)

5.109 E109: Gauss sums: magnitude patterns

Tags: `number-theory`, `dirichlet-characters`, `quantitative-exploration`, `visualization`, `numerics`

5.109.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.109.2 What is computed

- Compute Gauss sums ( ) and compare | ( )| to √q.

5.109.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

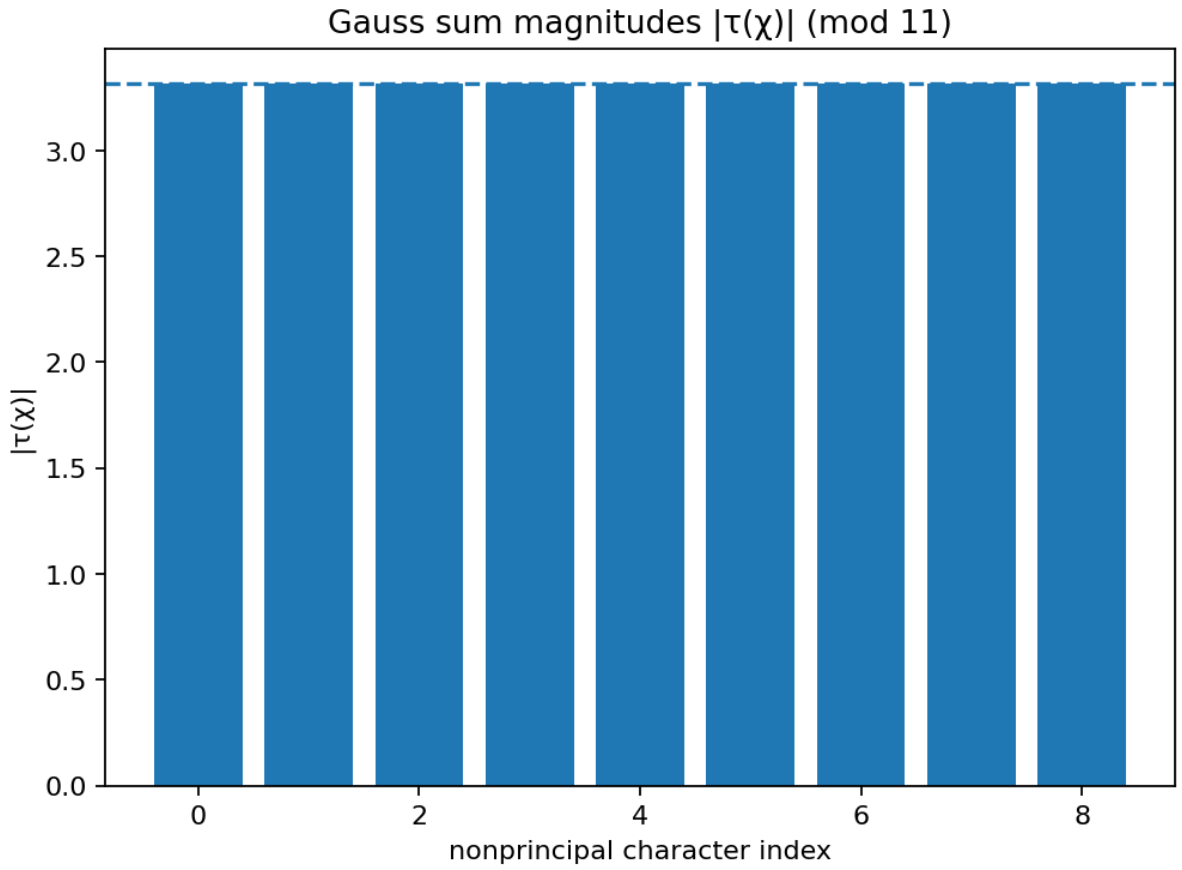


Fig. 29: E109: Gauss sums: magnitude patterns

5.109.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.109.5 Artifacts

- figures/fig_01_gauss_sum_magnitudes.png
- params.json
- report.md

Notes

- Nonprincipal characters: 9
- max | | ( )| - sqrt(q) |: 0.000000
- For prime q, nonprincipal characters are primitive and | ( )| = sqrt(q) (classic theorem).

params.json (snapshot)

```
{
  "q": 11
}
```

5.109.6 References

- See `docs/background/dirichlet-characters.md`.

5.109.7 Related experiments

- *E067: Gauss sums: magnitude vs. sqrt(q).* (Gauss sums: magnitude vs. sqrt(q).)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E120: Liouville (n): partial sums and parity* (E120: Liouville (n): partial sums and parity)

5.110 E110: Dirichlet L-series partial sums at s=1 and s=1/2

Tags: `number-theory`, `l-functions`, `quantitative-exploration`, `visualization`, `dirichlet-series`, `numerics`

5.110.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.110.2 What is computed

- Compare partial Dirichlet series behavior near s=1 and on the critical line.

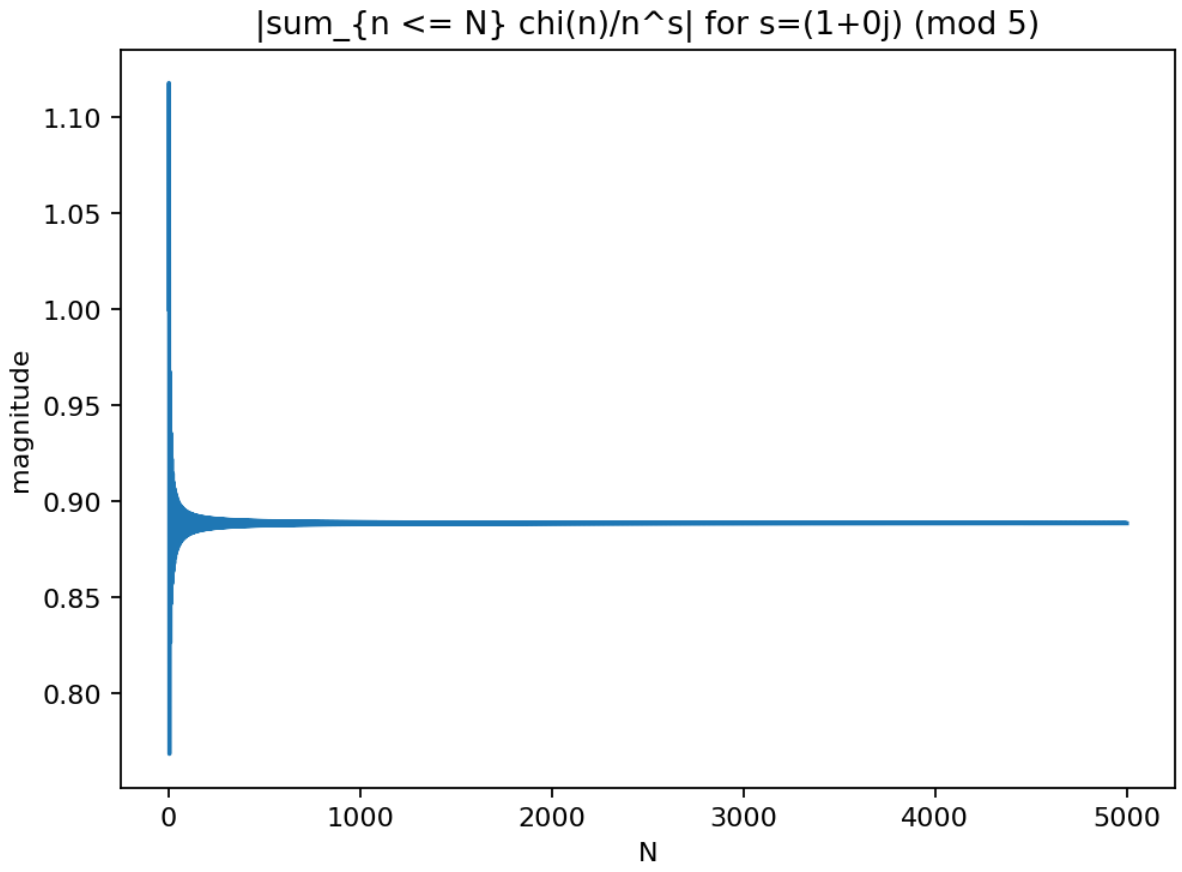


Fig. 30: E110: Dirichlet L-series partial sums at s=1 and s=1/2

5.110.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.110.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.110.5 Artifacts

- figures/fig_01_L_partial_magnitude.png
- params.json
- report.md

Notes

- Using first nonprincipal character mod q.
- Final partial sum at N=n_max: (0.8646862659836099+0.20411306614158545j).
- For nonprincipal and Re(s)=1, the series converges (slowly).

params.json (snapshot)

```
{
  "n_max": 5000,
  "q": 5,
  "s_re": 1.0
}
```

5.110.6 References

- See `docs/background/dirichlet-l-functions.md` and `docs/background/dirichlet-convolution.md`.

5.110.7 Related experiments

- *E111: Euler product vs. Dirichlet series for L(s,)* (E111: Euler product vs. Dirichlet series for L(s,))
- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E093: − (s)/ (s) via the von Mangoldt series* (E093: − (s)/ (s) via the von Mangoldt series)

5.111 E111: Euler product vs. Dirichlet series for L(s,χ)

Tags: `number-theory`, `l-functions`, `model-checking`, `visualization`, `dirichlet-series`, `numerics`

5.111.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

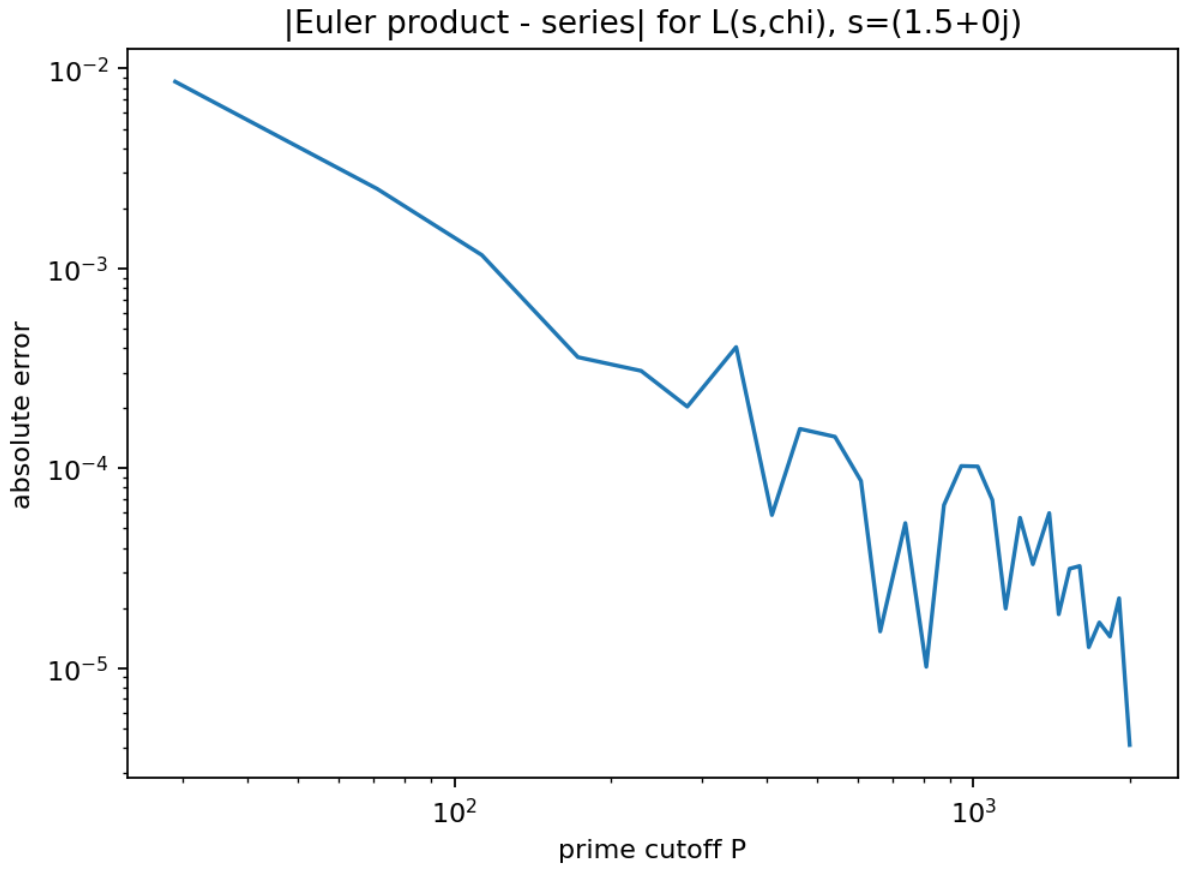


Fig. 31: E111: Euler product vs. Dirichlet series for L(s,)

5.111.2 What is computed

- Compare truncated Euler products and truncated series for L(s,).

5.111.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.111.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.111.5 Artifacts

- figures/fig_01_euler_product_error.png
- params.json
- report.md

Notes

- Reference computed by series up to N=8000
- Best (min) absolute error across P cutoffs: 4.123e-06
- For Re(s)>1, Euler product and series converge to the same L(s,).

params.json (snapshot)

```
{
  "n_series": 8000,
  "p_max": 2000,
  "p_steps": 30,
  "q": 5,
  "s_re": 1.5
}
```

5.111.6 References

- See `docs/background/dirichlet-l-functions.md` and `docs/background/dirichlet-convolution.md`.

5.111.7 Related experiments

- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))
- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)

5.112 E112: Prime race: π(x;q,a) − π(x;q,b)

Tags: `number-theory`, `prime-races`, `visualization`, `aps`, `primes`

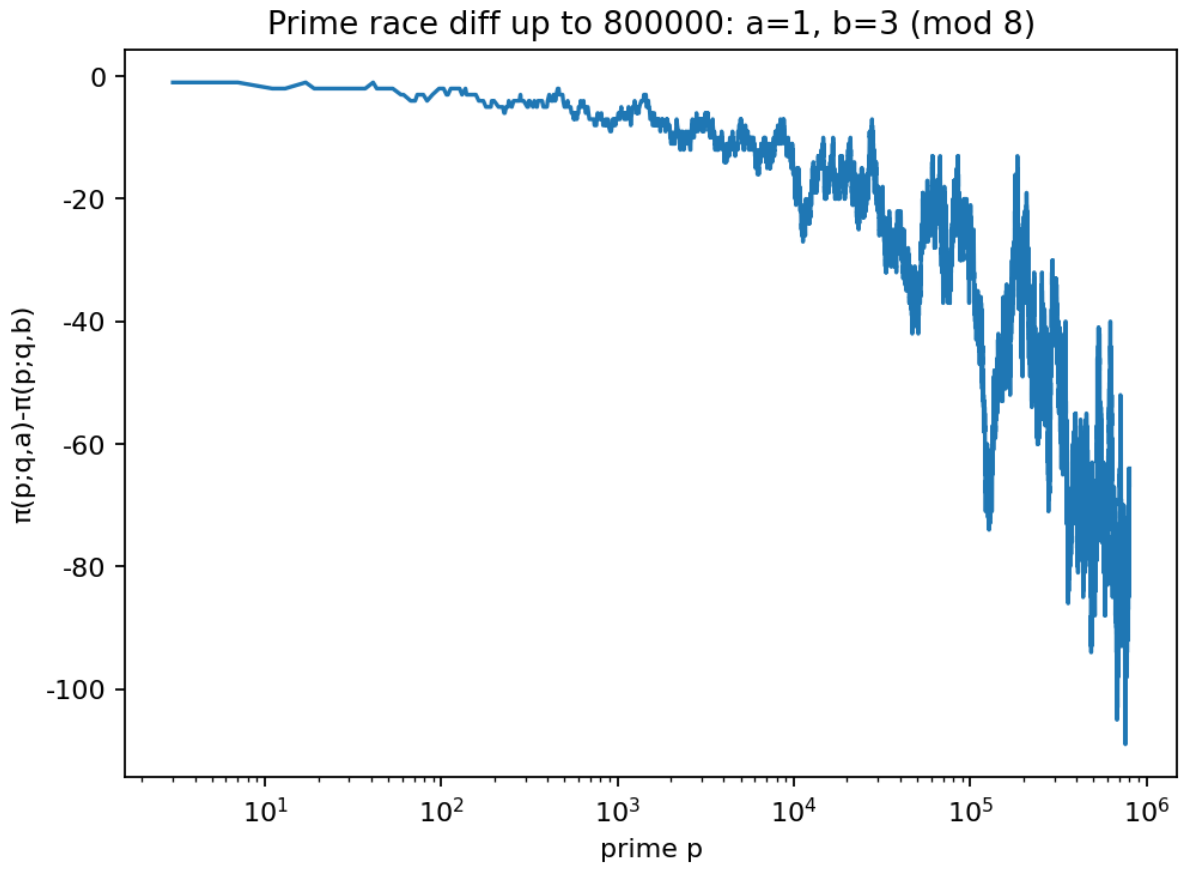


Fig. 32: E112: Prime race: (x;q,a) − (x;q,b)

5.112.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.112.2 What is computed

- Compute prime counts in residue classes and plot difference curves for representative pairs.

5.112.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.112.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.112.5 Artifacts

- figures/fig_01_prime_race_diff.png
- params.json
- report.md

Notes

- Final lead at p_max: -64
- The diff changes sign; persistent bias is subtle and depends on q,a,b.

params.json (snapshot)

```
{
  "a": 1,
  "b": 3,
  "p_max": 800000,
  "q": 8
}
```

5.112.6 References

- See `docs/background/primes-in-arithmetic-progressions.md` and `docs/background/prime-number-races.md`.

5.112.7 Related experiments

- *E113: First prime in each residue class* (E113: First prime in each residue class)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)

5.113 E113: First prime in each residue class

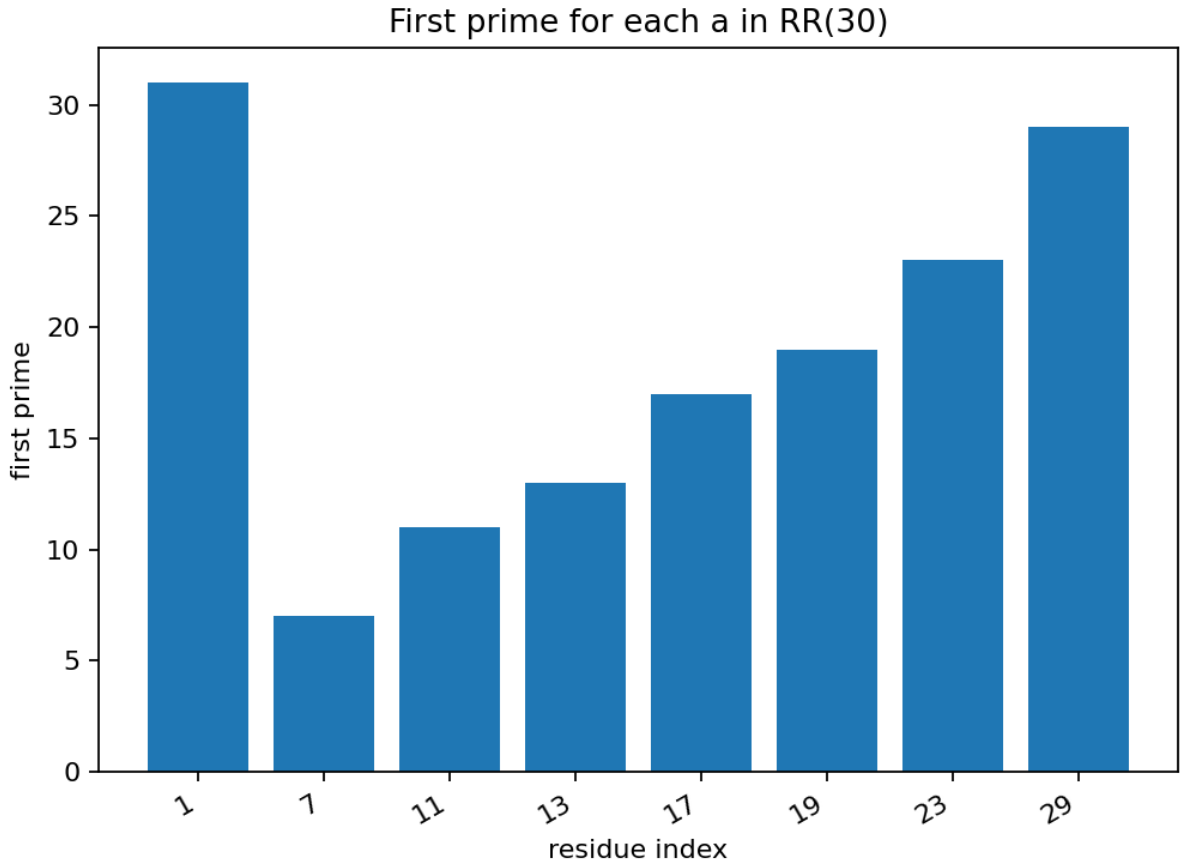


Fig. 33: E113: First prime in each residue class

Tags: `number-theory`, `prime-races`, `visualization`, `aps`, `primes`

5.113.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.113.2 What is computed

- For each reduced residue a (mod q), find the smallest prime p≡a (mod q) and visualize results.

5.113.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.113.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.113.5 Artifacts

- figures/fig_01_first_prime_per_residue.png
- params.json
- report.md

Notes

- RR(q) size: 8
- Missing residues within search bound: []
- This is a small-N version of the Linnik / least prime in AP story.

params.json (snapshot)

```
{
  "p_max": 200000,
  "q": 30
}
```

5.113.6 References

- See `docs/background/primes-in-arithmetic-progressions.md` and `docs/background/prime-number-races.md`.

5.113.7 Related experiments

- *E112: Prime race: (x;q,a) − (x;q,b)* (E112: Prime race: (x;q,a) − (x;q,b))
- *E023: Residue class distribution mod q* (Residue class distribution mod q)
- *E070: Primes in residue classes: pi(x; q, a).* (Primes in residue classes: pi(x; q, a).)
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)

5.114 E114: ζ via η: stability map on the critical line

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `critical-line`, `numerics`

5.114.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.114.2 What is computed

- Study stability of (1/2+it) via -acceleration under truncation changes.

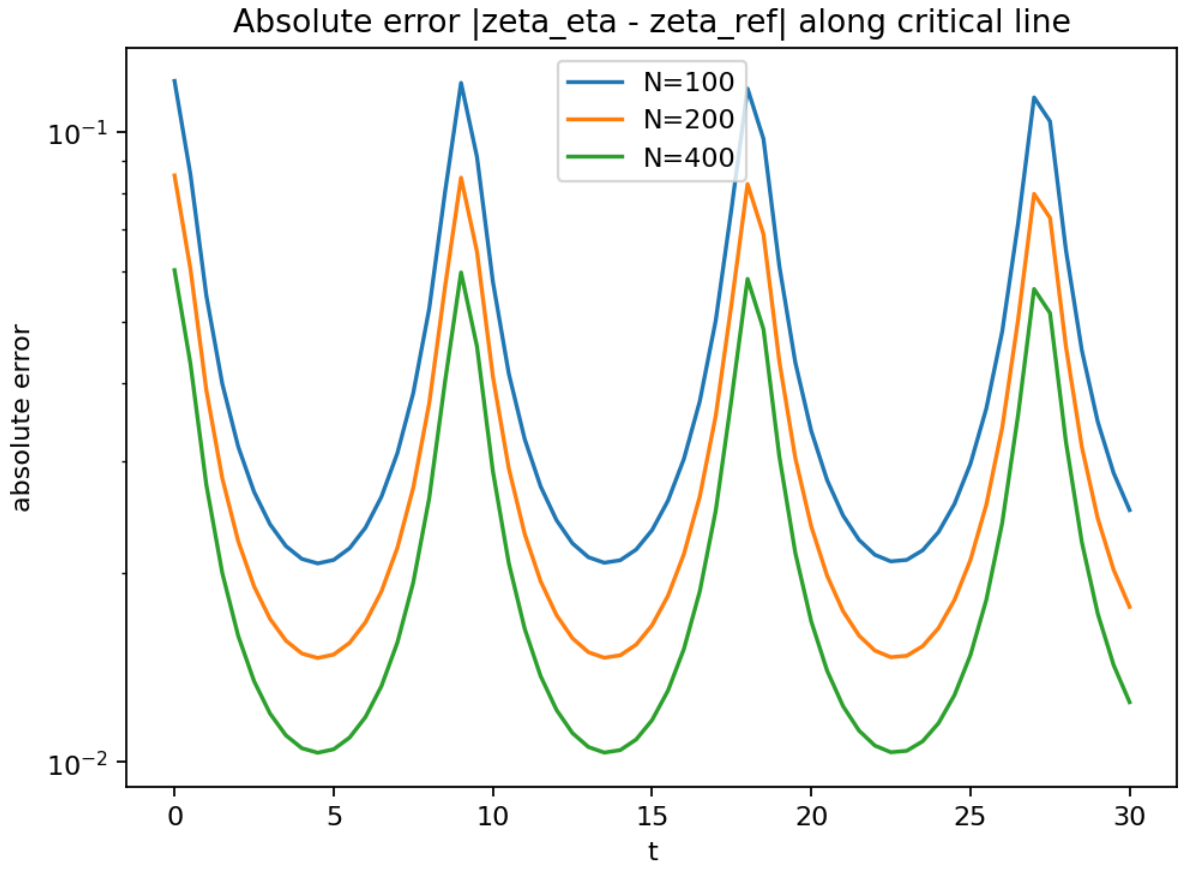


Fig. 34: E114: via : stability map on the critical line

5.114.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.114.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.114.5 Artifacts

- figures/fig_01_eta_truncation_errors.png
- params.json
- report.md

Notes

- n_terms -> max error: {100: '1.20e-01', 200: '8.52e-02', 400: '6.03e-02'}
- -series converges for Re(s)>0, but near Re(s)=1/2 the truncation error is still visible.

params.json (snapshot)

```
{
  "eval_dps": 50,
  "n_terms_values": [
    100,
    200,
    400
  ],
  "ref_dps": 70,
  "t_max": 30.0,
  "t_steps": 61
}
```

5.114.6 References

- See the zeta / zeros background pages in `docs/background/`.

5.114.7 Related experiments

- *E091: Partial Euler products on the critical line* (E091: Partial Euler products on the critical line)
- *E084: | (1/2+it)| growth snapshots* (E084: | (1/2+it)| growth snapshots)
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E082: Zeta(s) series convergence* (E082: Zeta(s) series convergence)
- *E083: Series vs. Euler product ( )* (E083: Series vs. Euler product ( ))

5.115 E115: Hardy Z: sign changes and zero bracketing

Tags: `analysis`, `quantitative-exploration`, `visualization`, `riemann-zeta`, `hardy-z`, `zeta-zeros`

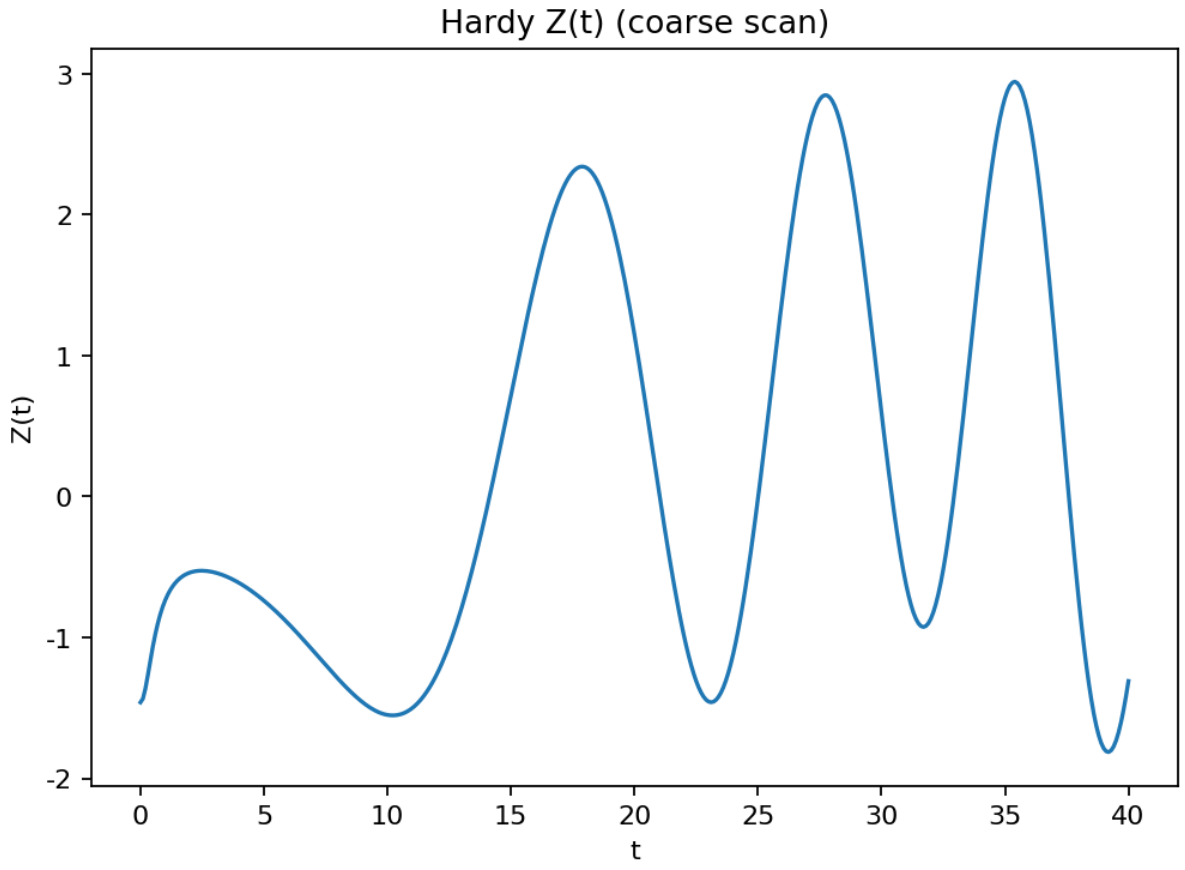


Fig. 35: E115: Hardy Z: sign changes and zero bracketing

5.115.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.115.2 What is computed

- Scan Hardy Z(t), bracket sign changes, and visualize candidate zeros.

5.115.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.115.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.115.5 Artifacts

- figures/fig_01_hardy_Z_scan.png
- params.json
- report.md

Notes

- Sign-change intervals (first 12): [(14.100000000000001, 14.200000000000001), (21.0, 21.1), (25.0, 25.1), (30.400000000000002, 30.5), (32.9, 33.0), (37.5, 37.6)]
- Each sign change suggests a zero of (1/2+it) in that interval.

params.json (snapshot)

```
{
  "dps": 40,
  "dt": 0.1,
  "t_max": 40.0
}
```

5.115.6 References

- See the zeta / zeros background pages in `docs/background/`.

5.115.7 Related experiments

- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)
- *E116: Gram points and zero-counting heuristics* (E116: Gram points and zero-counting heuristics)
- *E087: Gram points and spacing* (E087: Gram points and spacing)
- *E081: Prime race sign changes: first crossings table.* (Prime race sign changes: first crossings table.)

5.116 E116: Gram points and zero-counting heuristics

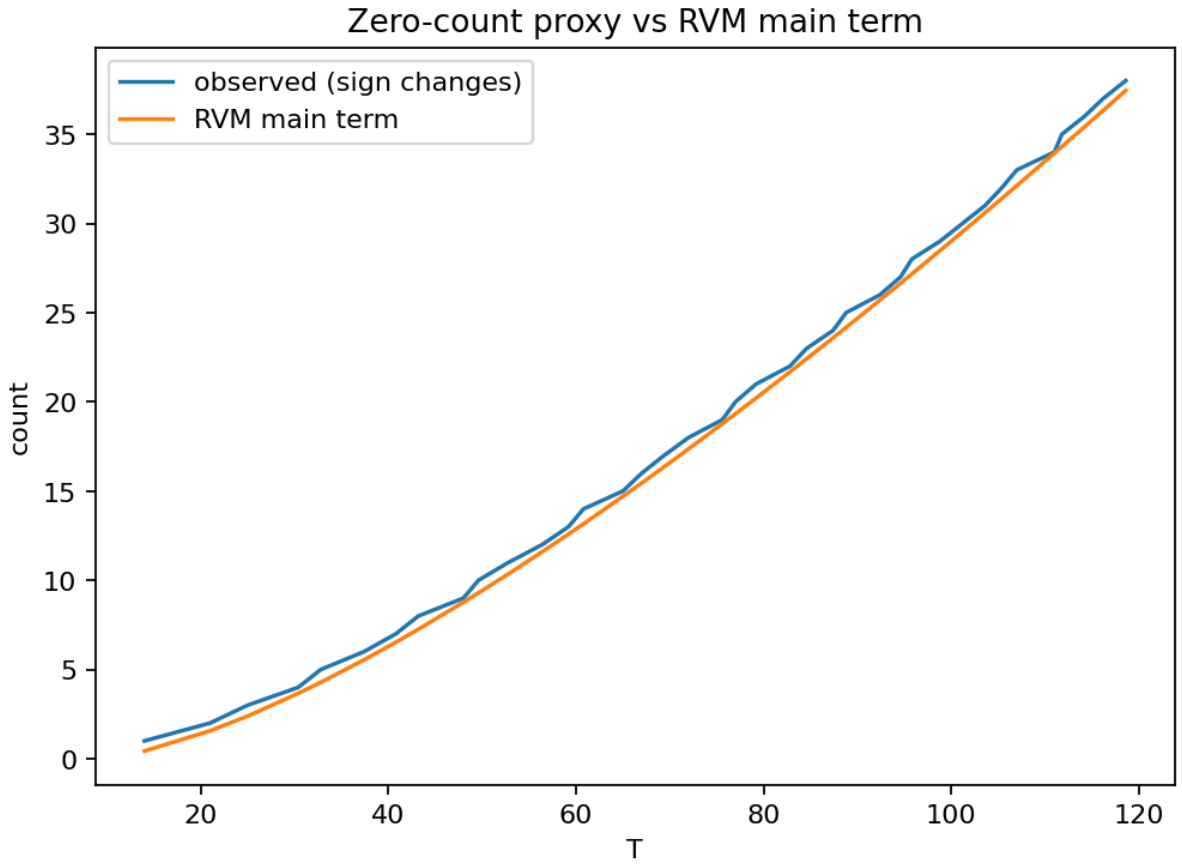


Fig. 36: E116: Gram points and zero-counting heuristics

Tags: `analysis`, `model-checking`, `visualization`, `riemann-zeta`, `gram-points`, `zeta-zeros`

5.116.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.116.2 What is computed

- Compare Gram-point heuristics and simple zero-counting diagnostics over a finite range.

5.116.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.116.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.116.5 Artifacts

- figures/fig_01_zero_count_vs_rvm.png
- params.json
- report.md

Notes

- Observed sign changes up to T: 38
- RVM main term at last observed T: 37.454
- Coarse scan undercounts (missed multiple zeros within a single dt bin) but tracks the growth trend.

params.json (snapshot)

```
{
  "dps": 40,
  "dt": 0.2,
  "t_max": 120.0
}
```

5.116.6 References

- See the zeta / zeros background pages in `docs/background/`.

5.116.7 Related experiments

- *E087: Gram points and spacing* (E087: Gram points and spacing)
- *E088: Zero counting via Riemann–von Mangoldt* (E088: Zero counting via Riemann–von Mangoldt)
- *E115: Hardy Z: sign changes and zero bracketing* (E115: Hardy Z: sign changes and zero bracketing)
- *E086: Hardy Z(t) near zeros* (E086: Hardy Z(t) near zeros)
- *E117: Prime-counting approximations: li(x) and friends* (E117: Prime-counting approximations: li(x) and friends)

5.117 E117: Prime-counting approximations: li(x) and friends

Tags: `number-theory`, `model-checking`, `visualization`, `li-x`, `pnt`, `bounds`

5.117.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.117.2 What is computed

- Compare (x) to li(x) and related approximations and visualize the signed error.

5.117.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

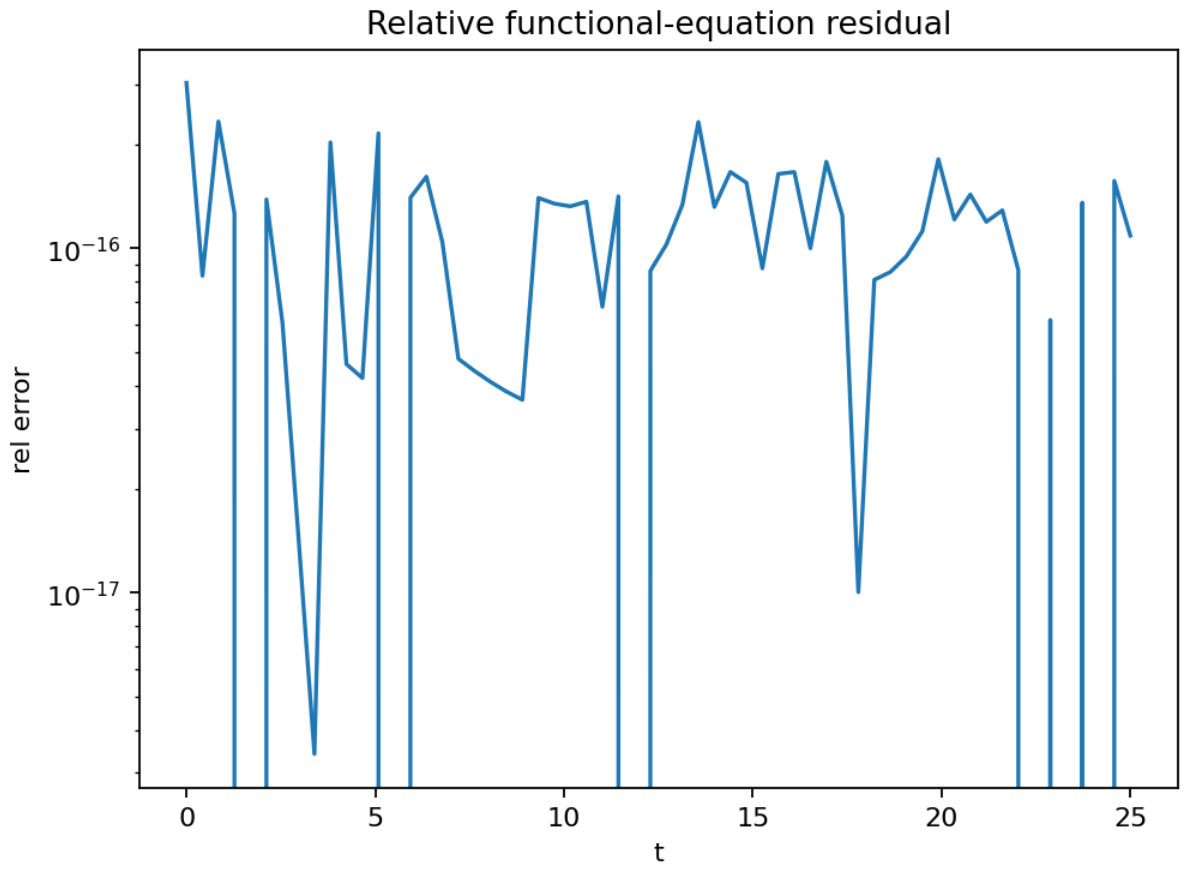


Fig. 37: E117: Prime-counting approximations: li(x) and friends

5.117.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.117.5 Artifacts

- figures/fig_01_functional_equation_residual.png
- params.json
- report.md

Notes

- max relative residual on grid: 3.03e-16
- This checks numerical consistency of chi_factor and mpmath zeta implementation on a small grid.

params.json (snapshot)

```
{
  "dps": 70,
  "sigma": 0.2,
  "t_max": 25.0,
  "t_steps": 60
}
```

5.117.6 References

- See the relevant references in `refs.bib`.

5.117.7 Related experiments

- *E123: (x;q,a) vs. a simple baseline* (E123: (x;q,a) vs. a simple baseline)
- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)
- *E116: Gram points and zero-counting heuristics* (E116: Gram points and zero-counting heuristics)
- *E058: Divisor-count record highs* (Divisor-count record highs)
- *E119: Summatory totient Φ(x) scaling check* (E119: Summatory totient Φ(x) scaling check)

5.118 E118: Chebyshev bias: lead-time statistics

Tags: `number-theory`, `prime-races`, `quantitative-exploration`, `visualization`, `heuristics`

5.118.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.118.2 What is computed

- Measure how often one residue class leads another and visualize lead-time statistics.

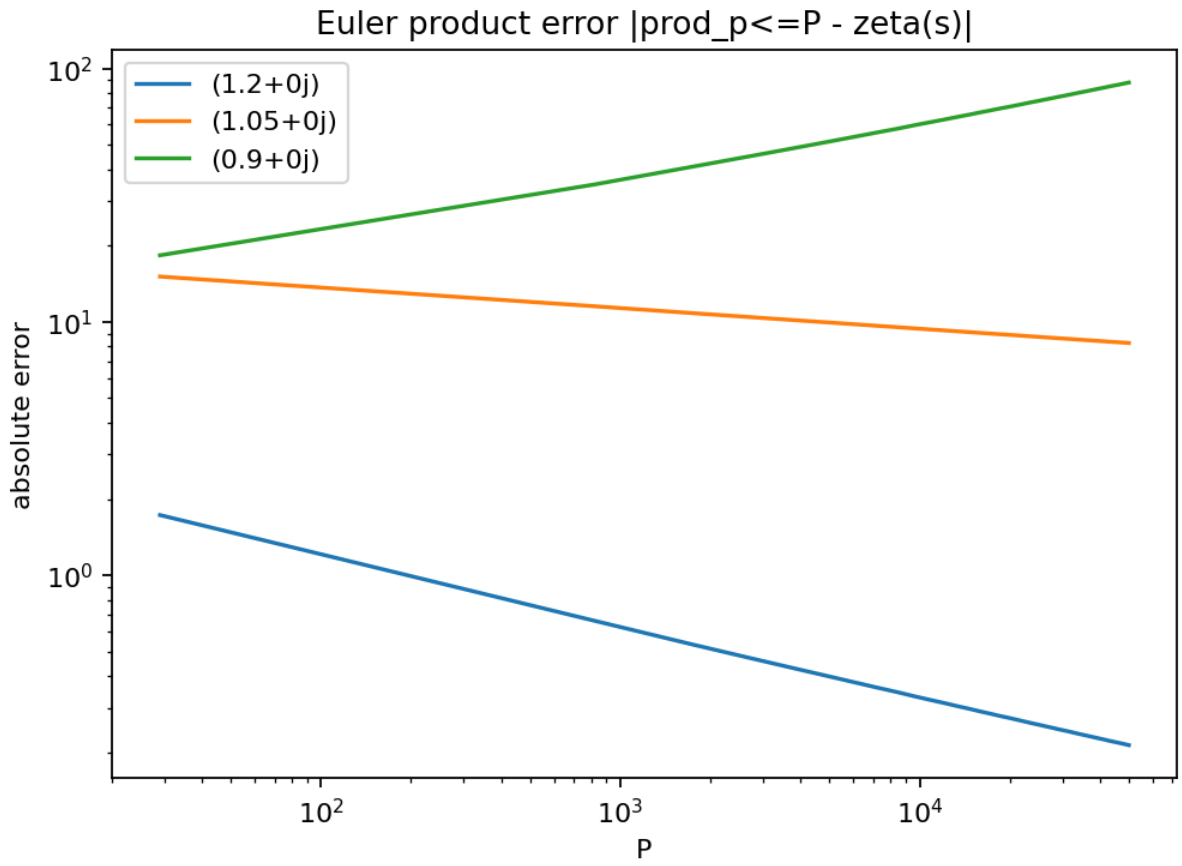


Fig. 38: E118: Chebyshev bias: lead-time statistics

5.118.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.118.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.118.5 Artifacts

- figures/fig_01_euler_product_breakdown.png
- params.json
- report.md

Notes

- Best (min) abs error per s: {'(1.2+0j)': '2.14e-01', '(1.05+0j)': '8.26e+00', '(0.9+0j)': '1.83e+01'}
- As Re(s) approaches 1 from the right, convergence slows; for Re(s)<=1 the Euler product no longer converges.

params.json (snapshot)


```
{
  "dps": 60,
  "p_max": 50000,
  "p_steps": 40,
  "s_values": [
    {
      "imag": 0.0,
      "real": 1.2
    },
    {
      "imag": 0.0,
      "real": 1.05
    },
    {
      "imag": 0.0,
      "real": 0.9
    }
  ]
}
```


5.118.6 References

- See `docs/background/primes-in-arithmetic-progressions.md` and `docs/background/prime-number-races.md`.

5.118.7 Related experiments

- *E080: Chebyshev bias: leader fraction vs. x.* (Chebyshev bias: leader fraction vs. x.)
- *E057: Erdős–Kac in practice* (Erdős–Kac in practice)
- *E076: Chebyshev (x;q,a): weighted prime counts in progressions.* (Chebyshev (x;q,a): weighted prime counts in progressions.)
- *E061: Chebyshev (x) and prime powers* (Chebyshev (x) and prime powers)

- *E103: Chebyshev (x): prime powers drive jumps* (E103: Chebyshev (x): prime powers drive jumps)

5.119 E119: Summatory totient Φ(x) scaling check

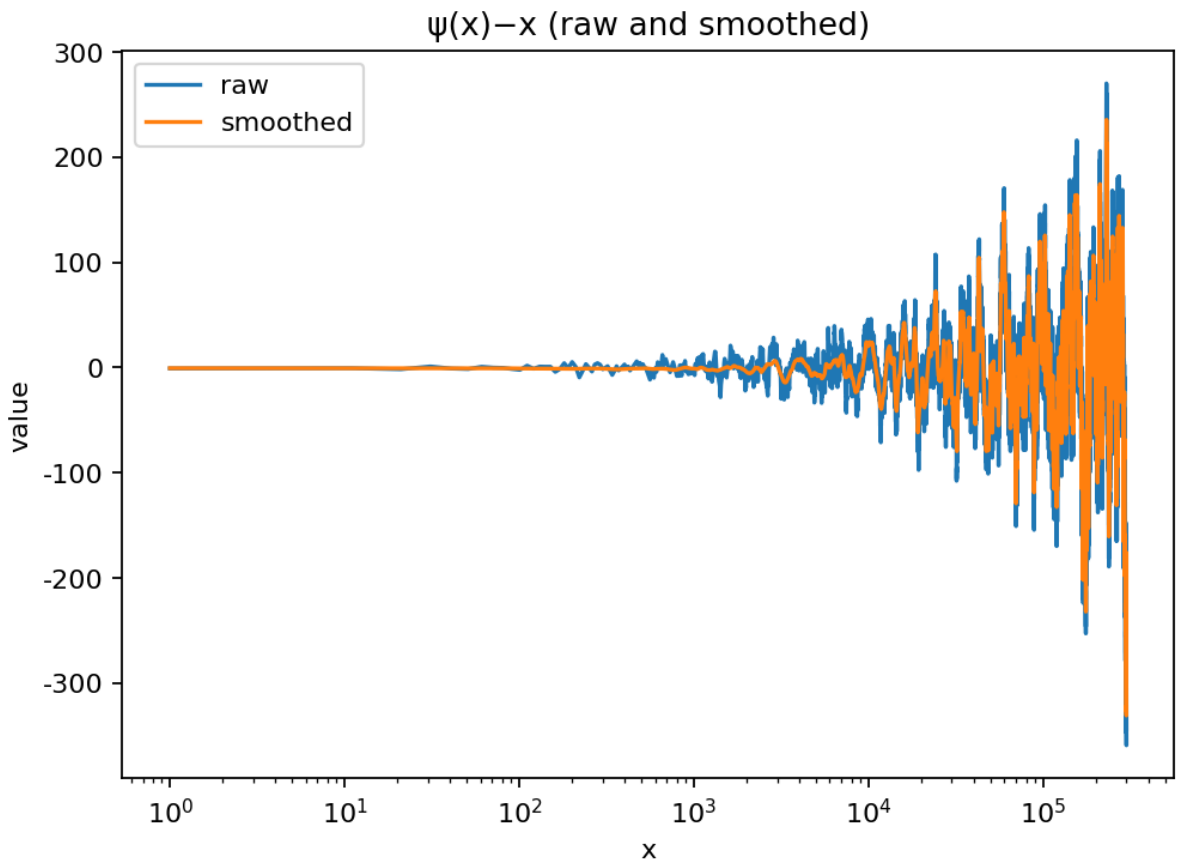


Fig. 39: E119: Summatory totient Φ(x) scaling check

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `totient`, `summatory`, `bounds`

5.119.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.119.2 What is computed

- Plot Φ(x)= _{n≤x} (n) and compare to the main-term scaling 3x²/ ².

5.119.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.119.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.119.5 Artifacts

- figures/fig_01_psi_minus_x_smoothed.png
- params.json
- report.md

Notes

- max | (x)−x| (raw): 360.3
- max | (x)−x| (smoothed): 331.0
- Smoothing suppresses prime-power spikes and reveals slower oscillations.

params.json (snapshot)

```
{
  "n_max": 300000,
  "stride": 10,
  "window": 500
}
```

5.119.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.119.7 Related experiments

- *E105: Mertens M(x): scaling views* (E105: Mertens M(x): scaling views)
- *E052: Totient ratio landscape* (Totient ratio landscape)
- *E053: Inverse totient multiplicities* (Inverse totient multiplicities)
- *E058: Divisor-count record highs* (Divisor-count record highs)
- *E117: Prime-counting approximations: li(x) and friends* (E117: Prime-counting approximations: li(x) and friends)

5.120 E120: Liouville λ(n): partial sums and parity

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `liouville`, `omega`

5.120.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI “slow” suite.

5.120.2 What is computed

- Explore partial sums of the Liouville function and simple parity correlations.

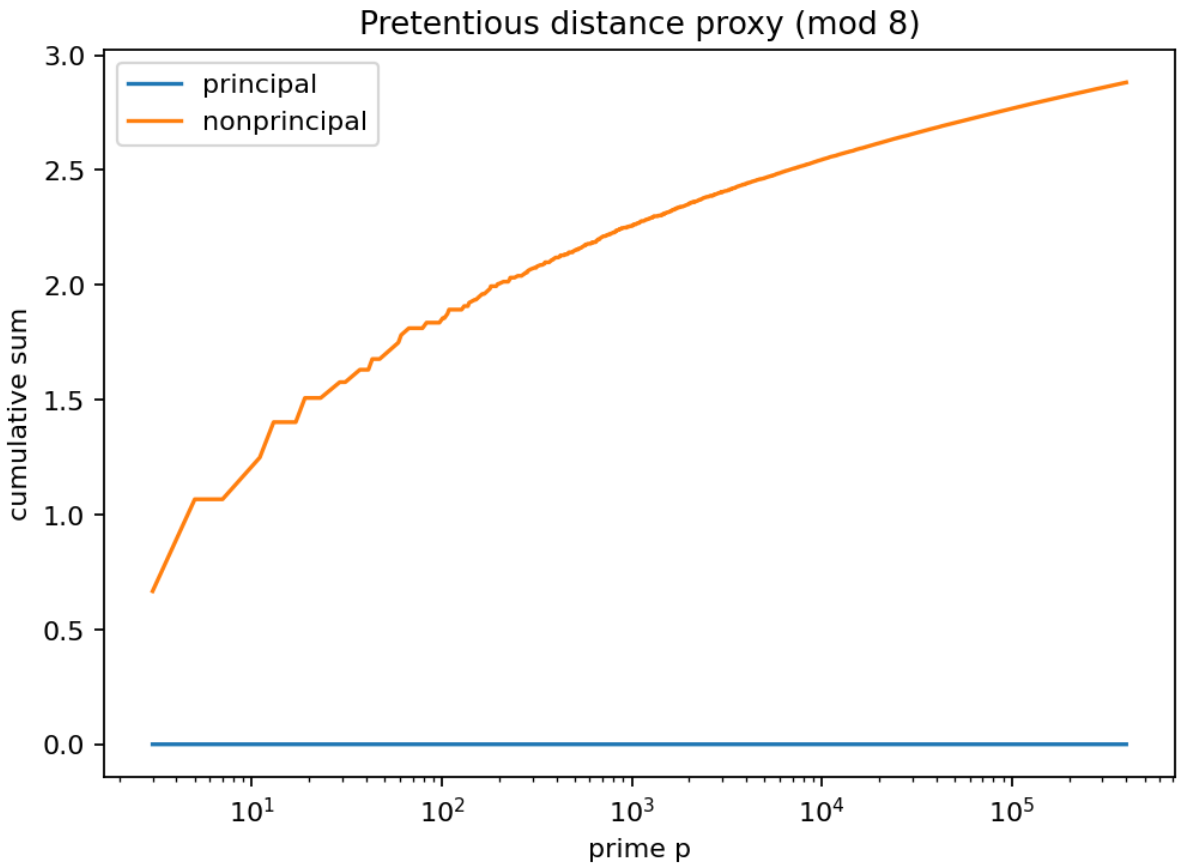


Fig. 40: E120: Liouville (n): partial sums and parity

5.120.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.120.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.120.5 Artifacts

- figures/fig_01_pretentious_distance_proxy.png
- params.json
- report.md

Notes

- Final value (principal): 0.000000
- Final value (nonprincipal): 2.881212
- This is a toy version of the pretentious distance framework (Granville–Soundararajan).

params.json (snapshot)

```
{
  "p_max": 400000,
  "q": 8
}
```

5.120.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.120.7 Related experiments

- *E056: Liouville vs. Möbius walks* (Liouville vs. Möbius walks)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)
- *E078: Max partial sums across characters.* (Max partial sums across characters.)
- *E110: Dirichlet L-series partial sums at s=1 and s=1/2* (E110: Dirichlet L-series partial sums at s=1 and s=1/2)
- *E057: Erdős–Kac in practice* (Erdős–Kac in practice)

5.121 E121: Möbius inversion as convolution undo

Tags: `number-theory`, `model-checking`, `visualization`, `dirichlet-convolution`, `mobius`, `multiplicative`

5.121.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

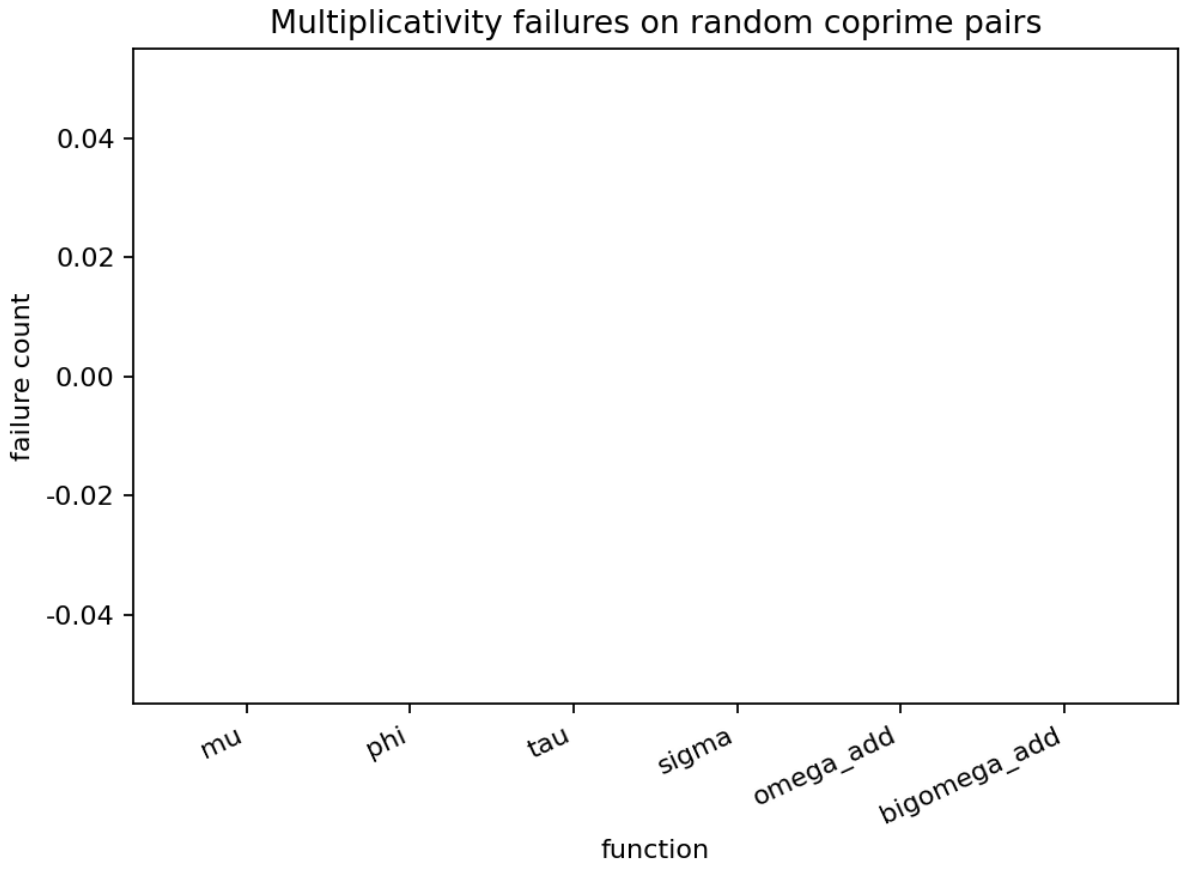


Fig. 41: E121: Möbius inversion as convolution undo

5.121.2 What is computed

- Recover a function from its Dirichlet convolution with 1 via Möbius inversion.

5.121.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.121.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.121.5 Artifacts

- figures/fig_01_multiplicativity_failures.png
- params.json
- report.md

Notes

- Tested coprime pairs (attempted): 3000
- Failure counts: {'mu': 0, 'phi': 0, 'tau': 0, 'sigma': 0, 'omega_add': 0, 'bigomega_add': 0}
- These should be 0 (within the sampled range and for coprime pairs).

params.json (snapshot)

```
{
  "n_max": 100000,
  "pairs": 3000
}
```

5.121.6 References

- See the arithmetic-functions background pages in `docs/background/`.

5.121.7 Related experiments

- *E102: Dirichlet convolution identity zoo* (E102: Dirichlet convolution identity zoo)
- *E063: Dirichlet convolution playground* (Dirichlet convolution playground)
- *E054: Squarefree density via Möbius* (Squarefree density via Möbius)
- *E056: Liouville vs. Möbius walks* (Liouville vs. Möbius walks)
- *E092: 1/ (s) via the Möbius Dirichlet series* (E092: 1/ (s) via the Möbius Dirichlet series)

5.122 E122: Character averages over primes

Tags: `number-theory`, `dirichlet-characters`, `l-functions`, `quantitative-exploration`, `visualization`, `aps`

5.122.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

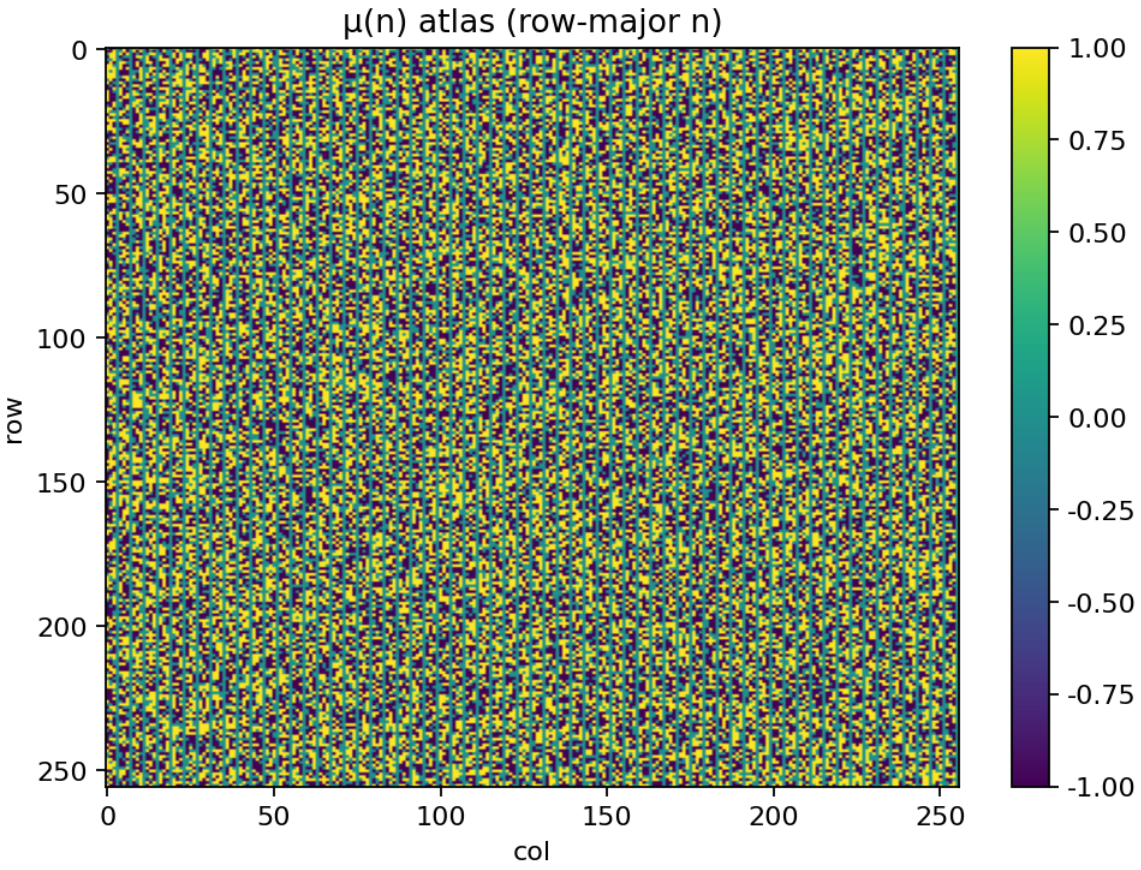


Fig. 42: E122: Character averages over primes

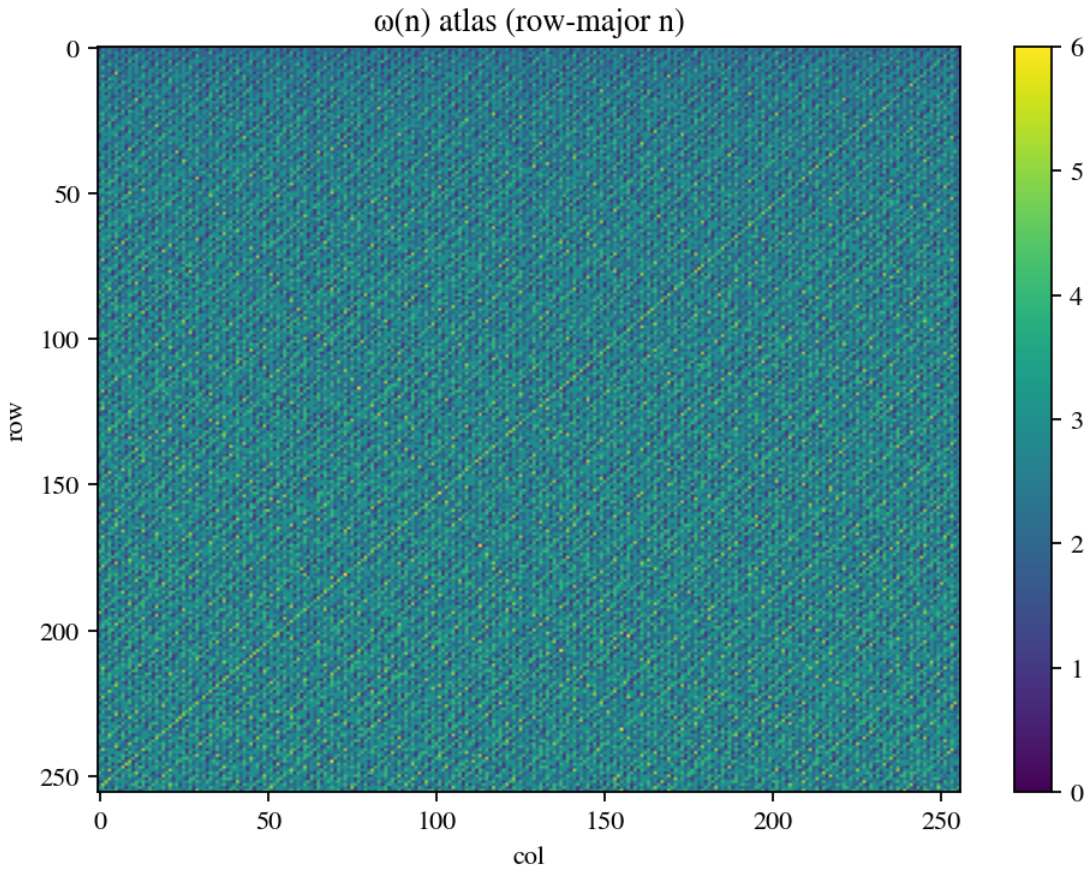


Fig. 43: E122: Character averages over primes

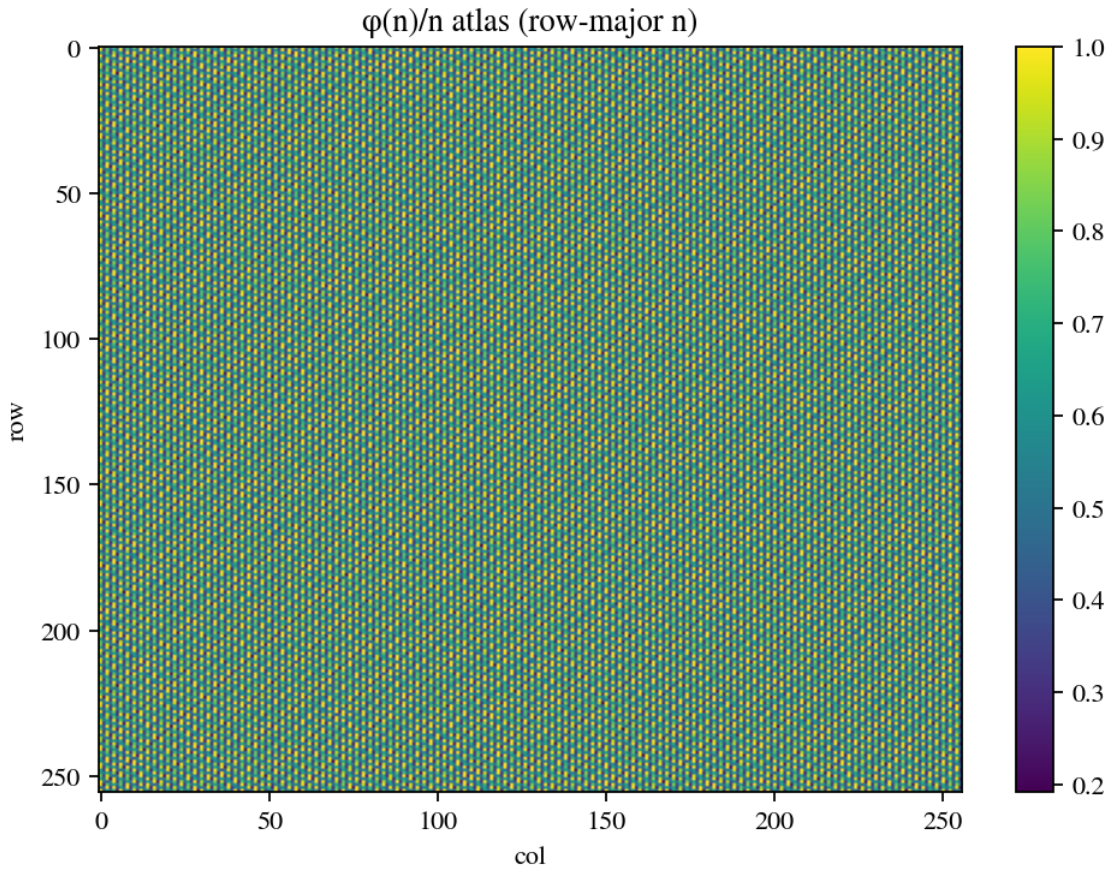


Fig. 44: E122: Character averages over primes

5.122.2 What is computed

- Average (p) over primes in arithmetic progressions and compared to equidistribution heuristics.

5.122.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.122.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.122.5 Artifacts

- figures/fig_01_mu_atlas.png
- figures/fig_02_omega_atlas.png
- figures/fig_03_phi_ratio_atlas.png
- params.json
- report.md

Notes

- Atlas size: side=256 => N=65536
- Visual textures encode multiplicativity and prime-power structure.

params.json (snapshot)

```
{
  "side": 256
}
```

5.122.6 References

- See `docs/background/dirichlet-characters.md`.
- See `docs/background/dirichlet-l-functions.md` and `docs/background/dirichlet-convolution.md`.
- See `docs/background/primes-in-arithmetic-progressions.md` and `docs/background/prime-number-races.md`.

5.122.7 Related experiments

- *E068: Dirichlet L(s,): series vs. Euler product (partial approximations).* (Dirichlet L(s,): series vs. Euler product (partial approximations).)
- *E069: L(1,): slow convergence and smoothing.* (L(1,): slow convergence and smoothing.)
- *E107: Conductor: primitive vs. induced characters* (E107: Conductor: primitive vs. induced characters)
- *E064: Dirichlet character tables (phase view).* (Dirichlet character tables (phase view).)
- *E066: Character partial sums: cancellation profiles.* (Character partial sums: cancellation profiles.)

5.123 E123: π(x;q,a) vs. a simple baseline

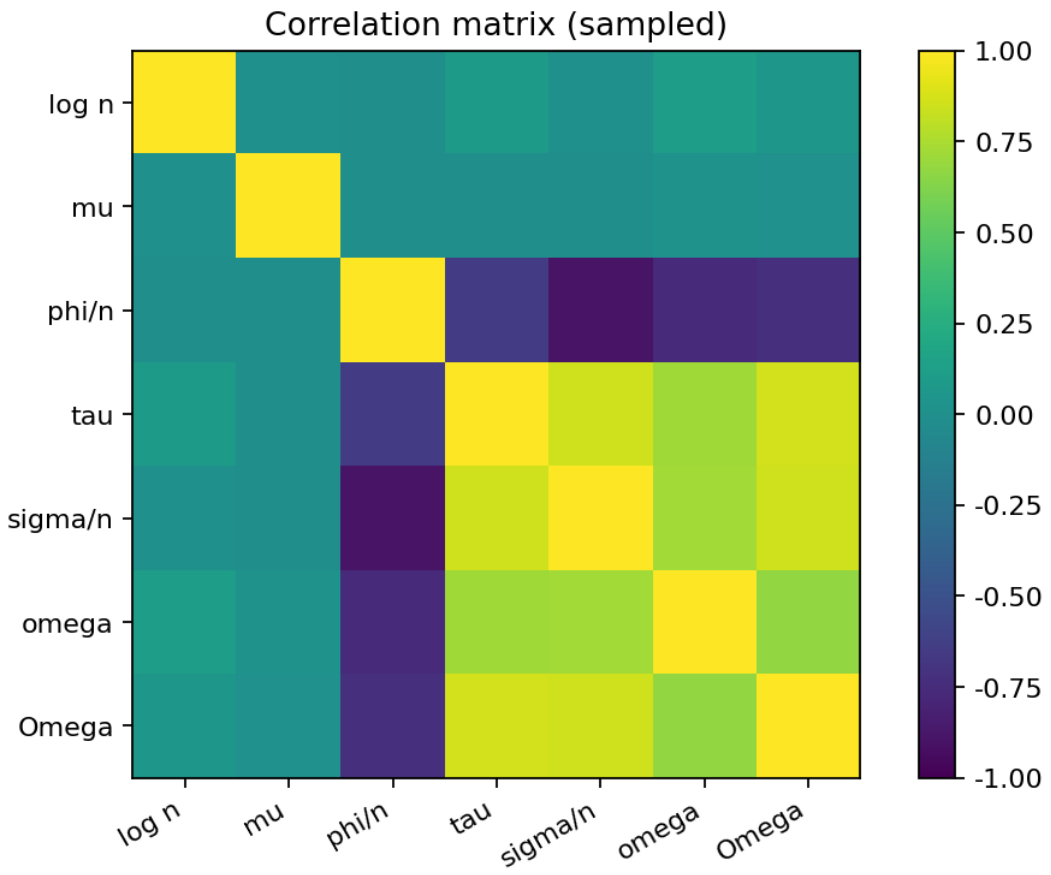


Fig. 45: E123: (x;q,a) vs. a simple baseline

Tags: `number-theory`, `prime-races`, `model-checking`, `visualization`, `aps`, `bounds`

5.123.1 Highlights

- Focused numeric experiment with a small set of figures.
- Parameters saved to `params.json` for reproducibility.
- Defaults are chosen, so the experiment remains feasible for the CI "slow" suite.

5.123.2 What is computed

- Compare (x;q,a) to a baseline x/((q) log x) and visualize deviations.

5.123.3 Notes

- This page summarizes the intent; see the generated `report.md` in `out/` for concrete outputs.

5.123.4 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

5.123.5 Artifacts

- figures/fig_01_correlation_matrix.png
- params.json
- report.md

Notes

- Variables: ['log n', 'mu', 'phi/n', 'tau', 'sigma/n', 'omega', 'Omega']
- Strongest |corr| pairs (name1, name2, |corr|): [('phi/n', 'sigma/n', 0.8944), ('tau', 'Omega', 0.8619), ('sigma/n', 'Omega', 0.856), ('tau', 'sigma/n', 0.8556), ('phi/n', 'omega', 0.7562), ('phi/n', 'Omega', 0.7209), ('sigma/n', 'omega', 0.7199), ('tau', 'omega', 0.7186)]
- Interpret cautiously: many variables are highly non-Gaussian and arithmetic in nature.

params.json (snapshot)

```
{
  "n_max": 80000,
  "stride": 1
}
```

5.123.6 References

- See `docs/background/primes-in-arithmetic-progressions.md` and `docs/background/prime-number-races.md`.

5.123.7 Related experiments

- *E117: Prime-counting approximations: li(x) and friends* (E117: Prime-counting approximations: li(x) and friends)
- *E072: Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).* (Prime race mod 4: pi(x;4,3) vs. pi(x;4,1).)
- *E073: Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).* (Prime race mod 3: pi(x;3,2) vs. pi(x;3,1).)
- *E074: Prime race mod 8: leaderboard among 1,3,5,7.* (Prime race mod 8: leaderboard among 1,3,5,7.)
- *E075: Prime race statistic: distribution on a log-grid.* (Prime race statistic: distribution on a log-grid.)

5.124 E124: Klauber triangle structure

size 101: 10,201 numbers and 1,252 primes

size 201: 40,401 numbers and 4,236 primes

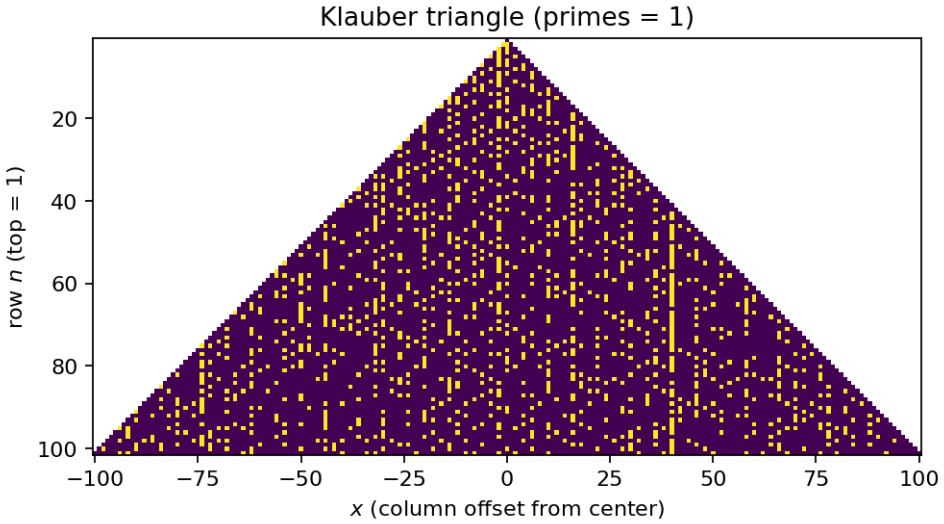
size 301: 90,601 numbers and 8,769 primes

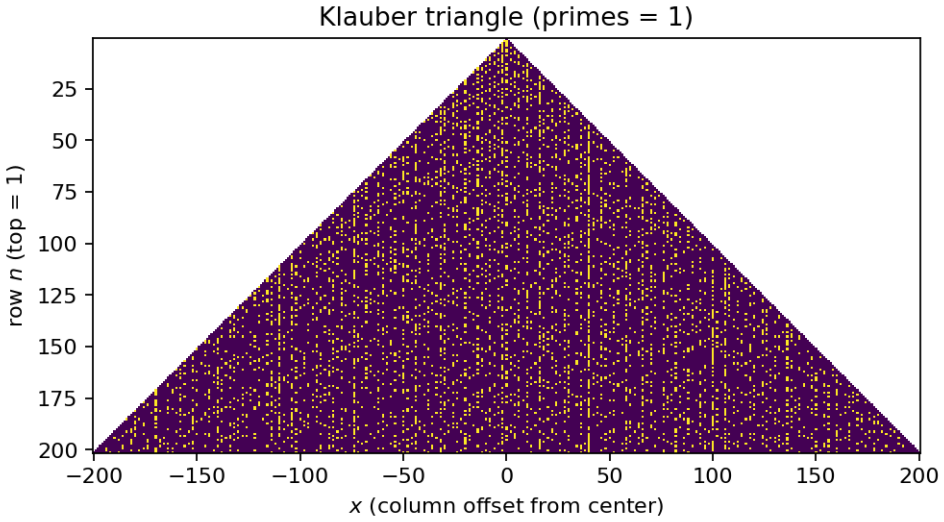
size 401: 160,801 numbers and 14,752 primes

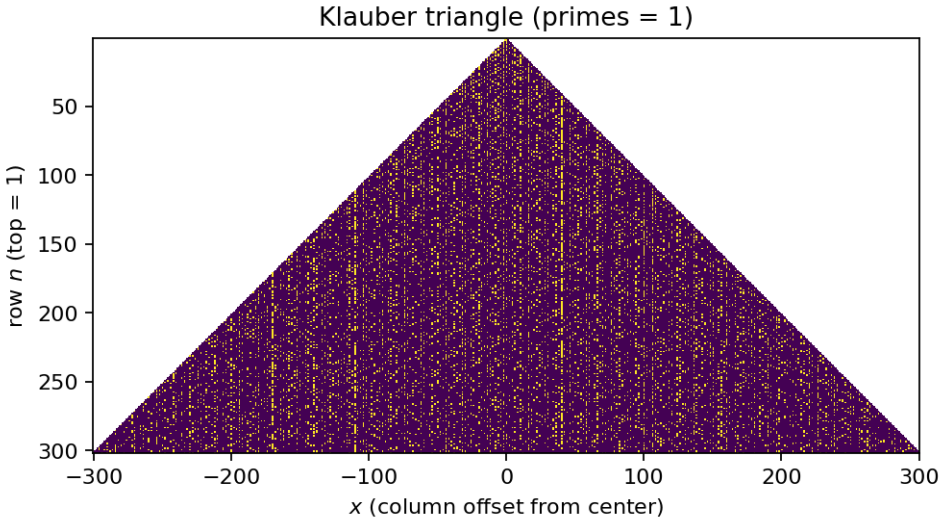
size 999: 998,001 numbers and 78,359 primes

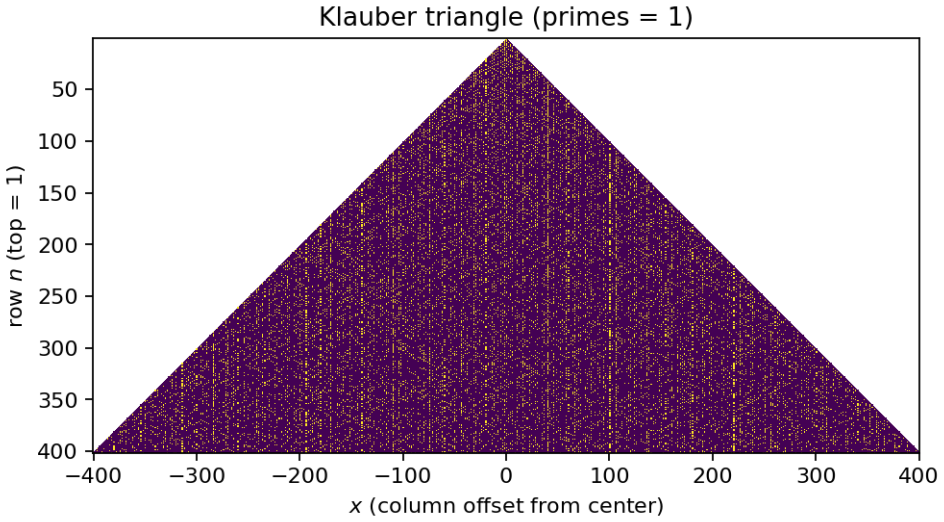
Prime counts shown are $\pi(size^2)$, i.e., the number of primes $\leq size^2$.

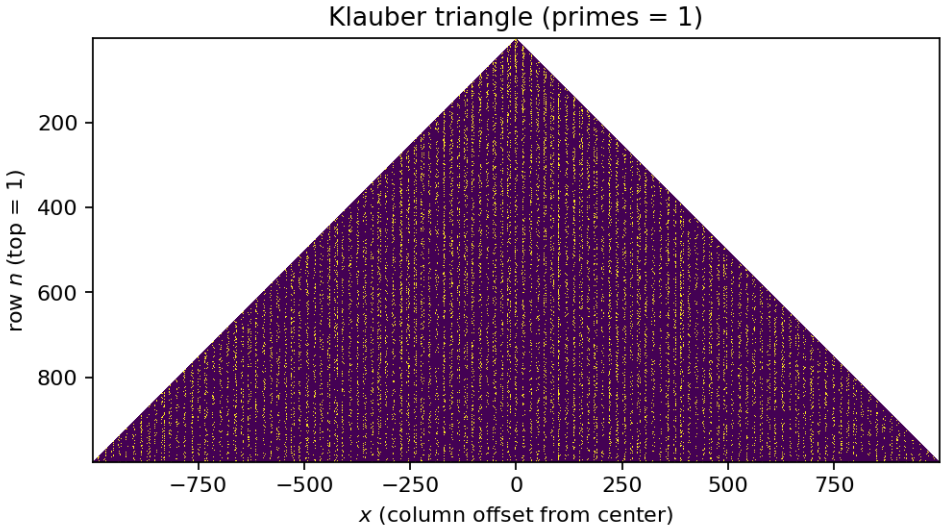
Tags: `number-theory`, `quantitative-exploration`, `visualization`
See: *Valid Tags*.


Klauber triangle (primes = 1)
row n (top = 1)
20
40
60
80
100
−100
−75
−50
−25
0
25
50
75
100
x (column offset from center)


Klauber triangle (primes = 1)
row n (top = 1)
25
50
75
100
125
150
175
200
−200
−150
−100
−50
0
50
100
150
200
x (column offset from center)


Klauber triangle (primes = 1)
row n (top = 1)
50
100
150
200
250
300
−300
−200
−100
0
100
200
300
x (column offset from center)


Klauber triangle (primes = 1)
row n (top = 1)
50
100
150
200
250
300
350
400
−400
−300
−200
−100
0
100
200
300
400
x (column offset from center)


Klauber triangle (primes = 1)
row n (top = 1)
200
400
600
800
−750
−500
−250
0
250
500
750
x (column offset from center)

5.124.1 Highlights

- A triangular "prime map" where row m contains the integers from $(m-1)^2+1$ to m^2.
- Parameterizable run: vary `--size` from the command line (default: `301`).
- Writes reproducible artifacts (`params.json`, `report.md`, and figures) into `out/e124/`.

5.124.2 Goal

Render the **Klauber triangle** (a square-number-based triangular arrangement) and highlight primes to reveal linear "streaks" and other geometric structures, then compare how that structure changes as `size` increases.

5.124.3 Background (quick refresher)

What is the Klauber triangle?

The **Klauber triangle** arranges the natural numbers into rows indexed by $m = 1, 2, 3, \ldots$ such that row m contains the consecutive integers

$$(m-1)^2+1,\ (m-1)^2+2, \ldots,\ m^2.$$

Each row has $2m-1$ entries, and the triangle up to row `size` contains the integers $1, \ldots, \text{size}^2$. When primes are highlighted, surprisingly strong visual streaks can appear. See [Hill, 2016].

Optional background pages

- *Prime numbers refresher*
- *Quadratic polynomials (algebraic) refresher*
- *Euler's prime-generating polynomial refresher*

5.124.4 Research question

Which prime-rich streaks are visible in the Klauber triangle for a given `size`, and how do those structures change as `size` increases?

5.124.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** build a finite triangle window containing integers $1, \ldots, \text{size}^2$ and run a deterministic primality test.
- **Observable(s):** streaks/lines, local clustering, and how visibility changes with `size`.
- **Parameter space:** `size` (and, if you extend it, rendering/annotation choices).
- **Outcome:** figures and a short report capturing the key observation and caveats.
- **Reproducibility:** outputs saved to `out/e124/` with a parameter snapshot.

5.124.6 Experiment design

- **Computation:** place integers $1, \ldots, \text{size}^2$ into the Klauber triangle row rule; mark primes.
- **Coordinates:** x is the integer column offset from the center column (0 at the center); the vertical axis is the row index n with $n = 1$ at the top.
- **Prime classification:** primes are computed using a sieve up to size^2.
- **Outputs:** one or more figures showing prime positions in the triangle.
- **Artifacts written:**

- `figures/fig_01_klauber_triangle*.png` (main figure) and `figures/e124_hero_<size>.png` (published hero)
- `params.json`
- `report.md`

Plot axes and coordinate conventions

The triangle is indexed by a row number and a horizontal offset within that row:

- **y-axis:** the row index $m \in \{1, \ldots, \text{size}\}$ (with $m = 1$ at the top)
- **x-axis:** the integer column offset from the center of the row:

$$x \in \{-(m-1), \ldots, (m-1)\}$$

For a given row m and offset x, the corresponding integer in the Klauber triangle is:

$$n(m, x) = (m-1)^2 + \big(x + (m-1)\big) + 1.$$

This formula matches the construction "odd squares on the left edge": the leftmost entry in row m is $n(m, -(m-1)) = (m-1)^2 + 1$, and the row increases by 1 as x moves to the right.

If the implementation renders the triangle as a 2D array with Matplotlib `imshow`, the `extent` should be chosen so that tick labels correspond to the displayed (x, m) values. When `origin="upper"` is used, m increases downward visually; in that case, interpret the y-axis tick labels accordingly (or flip the axis via the `extent`).

5.124.7 How to run

```
make run EXP=e124
```

Override the triangle size:

```
make run EXP=e124 ARGS="--size 501"
```

Direct invocation (always works):

```
uv run --extra dev python -m mathxlab.experiments.e124 --out out/e124 --size 501
```

5.124.8 Notes / pitfalls

- Larger `size` means more numbers (size^2) and a longer run time.
- "Looks-true" trap: visual regularity suggests hypotheses but does not prove theorems about primes.
- Rendering choices (marker size, interpolation) can hide or exaggerate streaks.

5.124.9 Extensions

- Quantify streaks: compute prime density along selected triangle-aligned lines and plot the densities.
- Highlight the square boundaries (m^2) or row indices to relate features to the row rule.
- Compare multiple sizes side-by-side (same styling and axis conventions).

5.124.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e124
```

Parameters

- size: `301` (odd); visualizes integers `1..90601`.

Notes

- The triangle arranges row n as `(n-1)^2+1 .. n^2` (row length `2n-1`).
- Prime-rich lines are linked to quadratic polynomials, similar to the Ulam spiral.
- This experiment is deterministic; `seed` does not change the output.

params.json (snapshot)

```
{
  "size": 301
}
```

5.124.11 References

- Klauber triangle definition and examples: [Hill, 2016].
- Exposition connecting multiple prime visualizations (includes Klauber and Sacks): [Brockmann, 2019].
- Recreational-math context and "pattern traps": [Gardner, 1983], [Hoffman, 1989].

See also ../references.

5.124.12 Related experiments

- *E024: Ulam spiral structure* (Ulam spiral structure)
- *E125: Sacks spiral structure* (Sacks spiral structure)
- *E126: Hexagonal number spiral structure* (Hexagonal number spiral structure)
- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (Quadratic prime-run atlas ($n^2 + an + b$))
- *E128: Quadratic modular obstructions (Euler-type)* (Quadratic modular obstructions (Euler-type))
- *E129: Euler lucky constants for n^2 + n + b* (Euler lucky constants for $n^2 + n + b$)

5.124.13 Parameters (example)

- size: `301` (triangle rows: 301, integers `1..90,601`)

Recommended `size` range

The Klauber triangle up to row `size` contains size^2 integers, so runtime and memory scale roughly with size^2.

A practical range that works well for most runs:

- **Minimum (still meaningful):** `size = 101`
- **Default / recommended:** `size = 301`

- **Comfortable upper range on typical laptops (varies):** `size = 999`
- **Above ~`size = 1501`:** expect noticeably higher runtime (and potentially memory pressure), depending on the machine and the primality implementation.

Rule of thumb: start with `301`, then try `501`, `701`, `901`. Increase further only if runtime remains acceptable.

5.124.14 Summary

We render the Klauber triangle containing the first 90,601 positive integers and highlight primes. Even at this moderate scale, visually prominent streaks appear: some triangle-aligned lines contain noticeably more primes than nearby lines.

5.124.15 Key observations

- Several streaks stand out strongly against a background that still looks relatively "noise-like".
- The most visible structures shift as `size` increases; some streaks weaken while others become clearer.

5.124.16 Interpretation

The Klauber triangle is a structured re-indexing of the integers where many visually straight features correspond to simple polynomial or arithmetic progressions in the underlying index. This makes it a useful "pattern detector" for prime-rich sequences, but it is not evidence of a global rule.

5.124.17 Caveats

- Finite window: patterns can look stronger/weaker depending on the chosen `size` and zoom level.
- Visualization choices matter: marker size and interpolation can hide or exaggerate streaks.
- Different triangle conventions (if you change the row rule) change which streaks appear.

5.125 E125: Sacks spiral structure

size 101: 10,201 numbers and 1,252 primes

size 201: 40,401 numbers and 4,236 primes

size 301: 90,601 numbers and 8,769 primes

size 401: 160,801 numbers and 14,752 primes

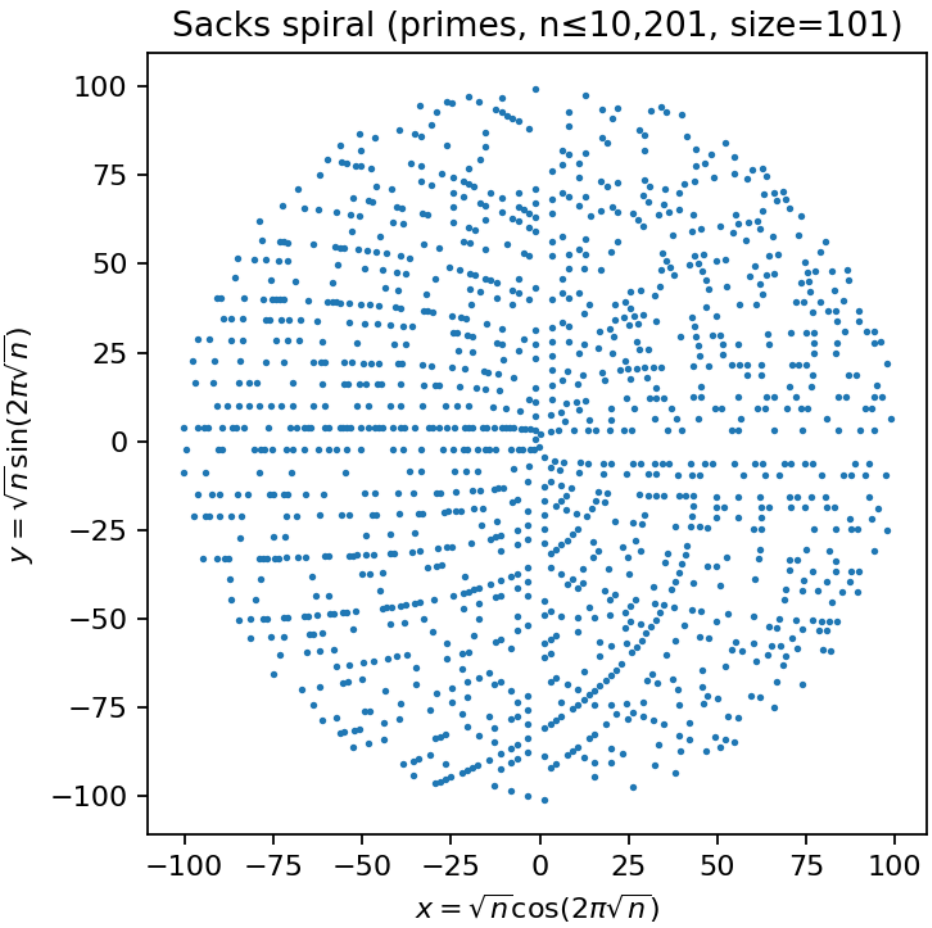
size 999: 998,001 numbers and 78,359 primes

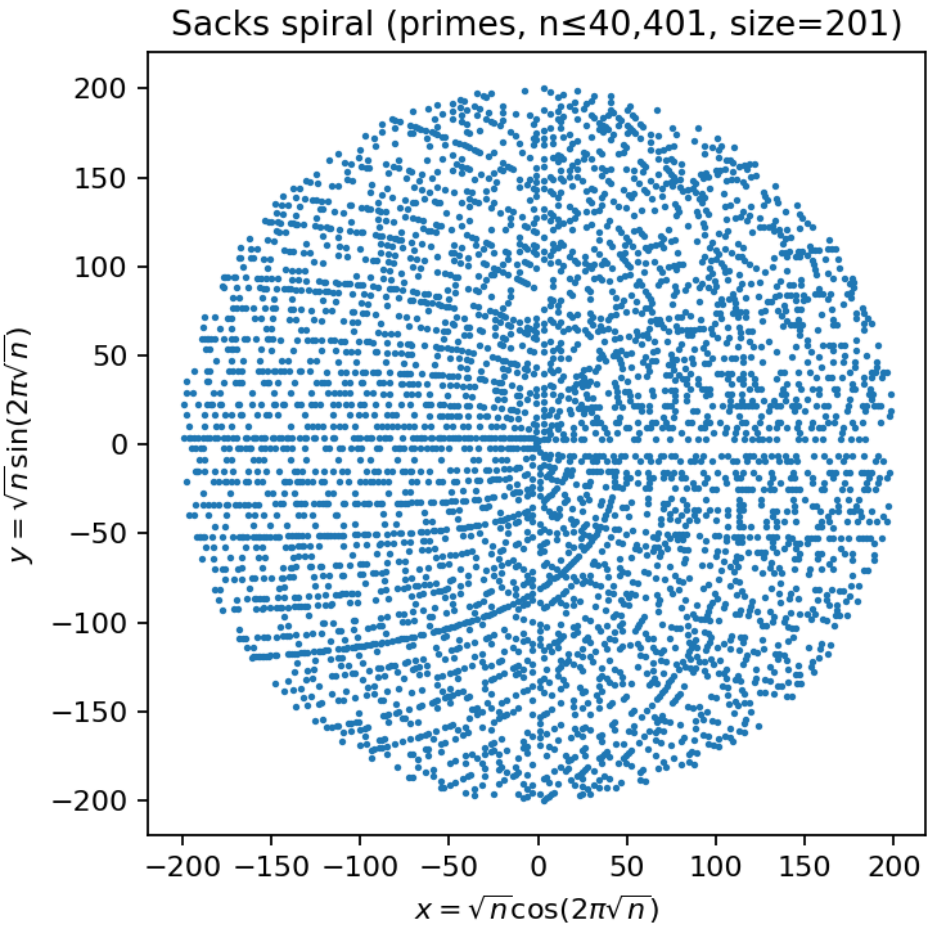
Prime counts shown are $\pi(size^2)$, i.e., the number of primes $\leq size^2$.

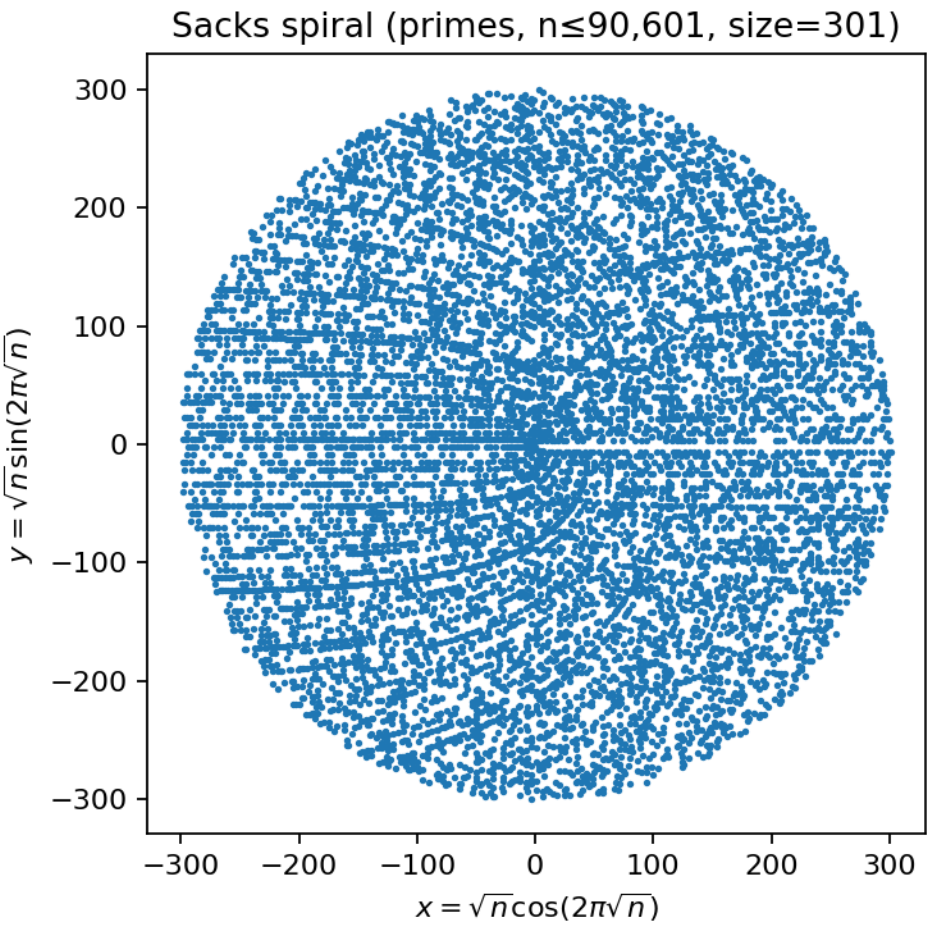
Tags: `number-theory`, `quantitative-exploration`, `visualization`
See: *Valid Tags*.

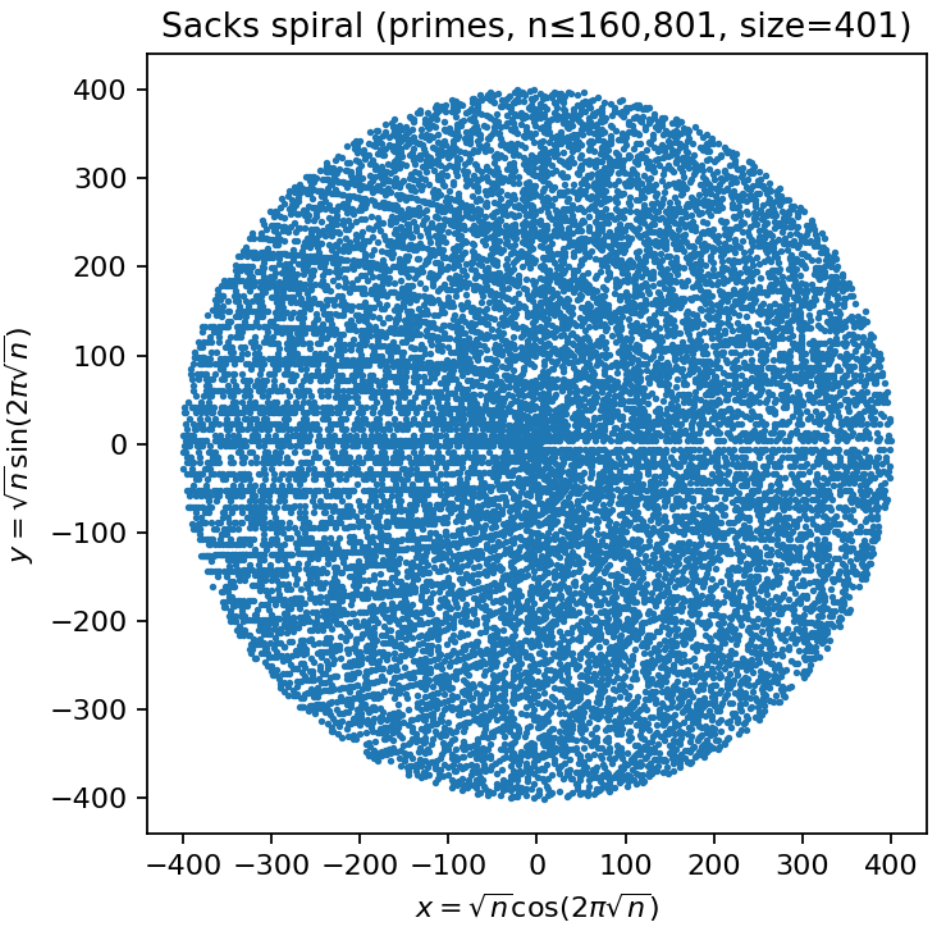
5.125.1 Highlights

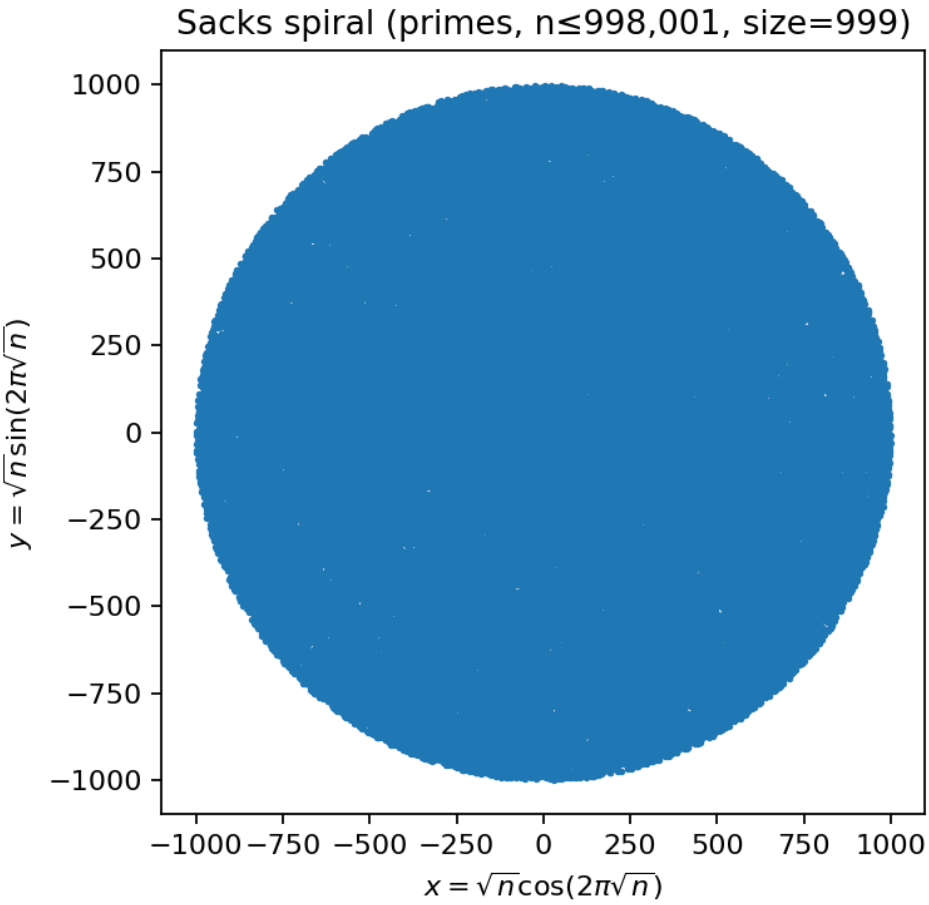
- Places integers on a spiral using polar coordinates $r = \sqrt{n}$ and $\theta = 2\pi\sqrt{n}$.
- Parameterizable run: vary `--size` from the command line (default: `301`).
- Writes reproducible artifacts (`params.json`, `report.md`, and figures) into `out/e125/`.


Sacks spiral (primes, n≤10,201, size=101)
100
75
50
25
0
−25
−50
−75
−100
−100 −75 −50 −25 0 25 50 75 100
$y = \sqrt{n}\sin(2\pi\sqrt{n})$
$x = \sqrt{n}\cos(2\pi\sqrt{n})$


Sacks spiral (primes, n≤40,401, size=201)
200
150
100
50
0
−50
−100
−150
−200
−200 −150 −100 −50 0 50 100 150 200
$y = \sqrt{n}\sin(2\pi\sqrt{n})$
$x = \sqrt{n}\cos(2\pi\sqrt{n})$


Sacks spiral (primes, n≤90,601, size=301)
300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
$y = \sqrt{n}\sin(2\pi\sqrt{n})$
$x = \sqrt{n}\cos(2\pi\sqrt{n})$


Sacks spiral (primes, n≤160,801, size=401)
400
300
200
100
0
−100
−200
−300
−400
−400 −300 −200 −100 0 100 200 300 400
$y = \sqrt{n}\sin(2\pi\sqrt{n})$
$x = \sqrt{n}\cos(2\pi\sqrt{n})$


Sacks spiral (primes, n≤998,001, size=999)
1000
750
500
250
0
−250
−500
−750
−1000
−1000 −750 −500 −250 0 250 500 750 1000
$y = \sqrt{n}\sin(2\pi\sqrt{n})$
$x = \sqrt{n}\cos(2\pi\sqrt{n})$

5.125.2 Goal

Render the **Sacks spiral** and highlight primes as a point cloud to reveal spiral-aligned structure, then compare how the apparent density and "spiral fragments" change as `size` increases.

5.125.3 Background (quick refresher)

What is the Sacks spiral?

In the **Sacks spiral**, each integer n is mapped to polar coordinates

$$r_n = \sqrt{n}, \qquad \theta_n = 2\pi\sqrt{n},$$

and plotted as $(x, y) = (r\cos\theta,\ r\sin\theta)$. Highlighting primes yields striking spiral fragments and density variations. See [Brockmann, 2019].

Optional background pages

- *Prime numbers refresher*

5.125.4 Research question

How does the visual "prime spiral density" change as `size` increases (i.e., as we include integers up to size^2), and which spiral-aligned structures persist across scales?

5.125.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** compute primes up to size^2 and apply a deterministic mapping $n \mapsto (x, y)$.
- **Observable(s):** spiral fragments, density contrasts, and how visibility changes with `size`.
- **Parameter space:** `size` (and, if you extend it, marker size / rendering choices).
- **Outcome:** figures and a short report capturing the key observation and caveats.
- **Reproducibility:** outputs saved to `out/e125/` with a parameter snapshot.

5.125.6 Experiment design

- **Computation:** for integers $1, \ldots, \text{size}^2$, compute a prime mask; map prime indices to (x, y) via the Sacks rule.
- **Coordinates:** Cartesian coordinates (x, y) are computed from $r = \sqrt{n}$ and $\theta = 2\pi\sqrt{n}$ via $(x, y) = (r\cos\theta,\ r\sin\theta)$.
- **Prime classification:** primes are computed deterministically up to size^2 (sieve).
- **Outputs:** one or more scatter plots of prime points (composites omitted).
- **Artifacts written:**
  - `figures/fig_01_sacks_spiral*.png` (main figure) and `figures/e125_hero_<size>.png` (published hero)
  - `params.json`
  - `report.md`

Axis scaling note

The run includes integers $n \leq \text{size}^2$. With the Sacks choice $r = \sqrt{n}$, the maximal radius shown is

$$r_{\max} = \sqrt{\text{size}^2} = \text{size},$$

, so the plot naturally fits inside a disk of radius about `size`.

5.125.7 How to run

```
make run EXP=e125
```

Override the experiment size:

```
make run EXP=e125 ARGS="--size 501"
```

Direct invocation (always works):

```
uv run --extra dev python -m mathxlab.experiments.e125 --out out/e125 --size 501
```

5.125.8 Notes / pitfalls

- Scatter plots are sensitive to marker size and resolution: the same data can look different depending on rendering.
- Larger `size` means more numbers (size^2) and a longer run time.
- "Looks-true" trap: a visually strong spiral does not by itself imply a theorem about primes.

5.125.9 Extensions

- Add a faint background of composite points (with transparency) to compare prime vs. composite geometry.
- Plot multiple sizes on identical axes to compare density and visible spiral fragments.
- Add simple quantitative summaries (e.g., radial density bins for primes vs. all numbers).

5.125.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e125
```

Parameters

- size: `301` (odd); visualizes integers `1..90601`.

Notes

- Construction: `r = sqrt(n), θ = 2π sqrt(n)` so squares align on a ray.
- Prime-rich curves correspond to quadratic polynomials under this embedding.
- This experiment is deterministic; `seed` does not change the output.

params.json (snapshot)

```
{
  "size": 301
}
```

5.125.11 References

- Exposition connecting multiple prime visualizations (includes Sacks spiral): [Brockmann, 2019].
- Recreational-math context and "pattern traps": [Gardner, 1983], [Hoffman, 1989].

See also ../references.

5.125.12 Related experiments

- *E024: Ulam spiral structure* (Ulam spiral structure)
- *E124: Klauber triangle structure* (Klauber triangle structure)
- *E126: Hexagonal number spiral structure* (Hexagonal number spiral structure)
- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (Quadratic prime-run atlas ($n^2 + an + b$))
- *E128: Quadratic modular obstructions (Euler-type)* (Quadratic modular obstructions (Euler-type))
- *E129: Euler lucky constants for n^2 + n + b* (Euler lucky constants for $n^2 + n + b$)

5.125.13 Parameters (example)

- size: `301` (implied total integers: `1..90,601`)

Recommended `size` range

The Sacks spiral run includes integers $1, \ldots, \text{size}^2$, so runtime and memory scale roughly with size^2 (prime classification dominates).

A practical range that works well for most runs:

- **Minimum (still meaningful):** `size = 101`
- **Default / recommended:** `size = 301`
- **Comfortable upper range on typical laptops (varies):** `size = 999`
- **Above ~`size = 1501`:** expect noticeably higher runtime (and potentially memory pressure), depending on the machine and the primality implementation.

Rule of thumb: start with `301`, then try `501`, `701`, `901`. Increase further only if runtime remains acceptable.

5.125.14 Summary

We compute primes among the first 90,601 positive integers and plot prime indices using the Sacks polar mapping. Even at this moderate scale, visible spiral-aligned prime fragments emerge, with noticeable density variation across radii.

5.125.15 Key observations

- Prime points form dense spiral segments rather than appearing uniformly "random" in the plane.
- The visibility of segments depends strongly on marker size and figure resolution.

5.125.16 Interpretation

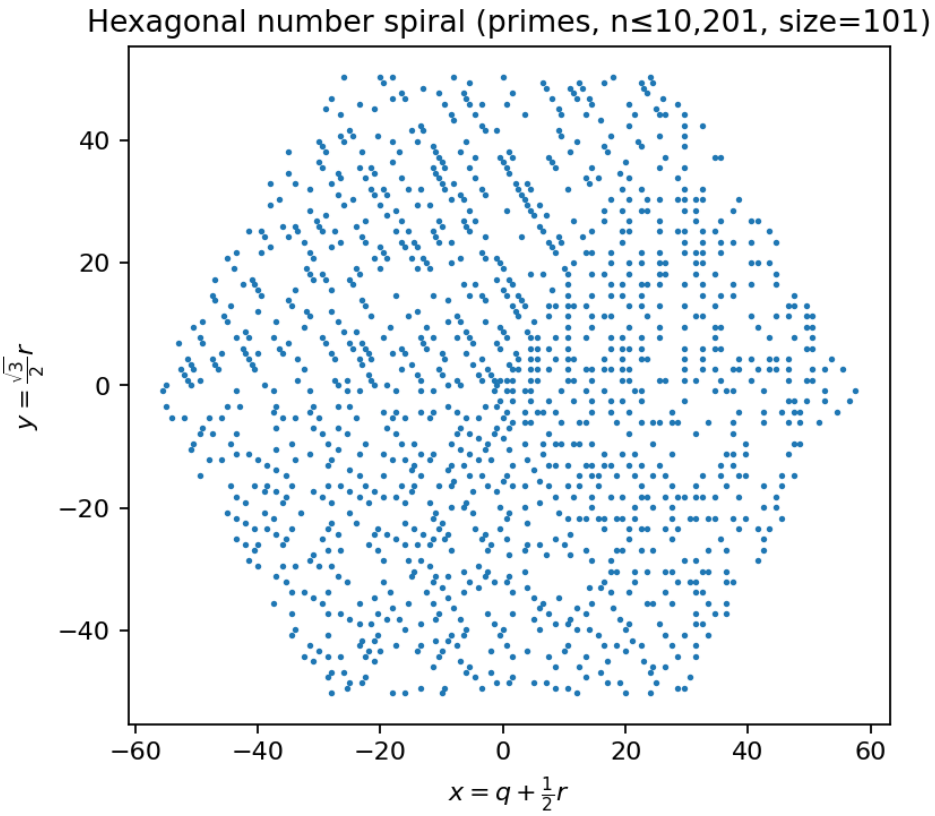
The Sacks mapping is a deterministic geometric transform of the integer line. It tends to visually emphasize arithmetic structure (including sequences with prime-rich behavior), making it a useful exploratory visualization tool — but not a proof technique.

5.125.17 Caveats

- Rendering matters: marker size, DPI, and axis limits can change perceived structure.
- Finite window: patterns can strengthen or weaken as `size` changes.
- Without quantitative overlays, conclusions remain qualitative.

5.126 E126: Hexagonal number spiral structure

size 101: 10,201 numbers and 1,252 primes



size 201: 40,401 numbers and 4,236 primes

size 301: 90,601 numbers and 8,769 primes

size 401: 160,801 numbers and 14,752 primes

size 999: 998,001 numbers and 78,359 primes

Prime counts shown are $\pi(size^2)$, i.e., the number of primes $\leq size^2$.

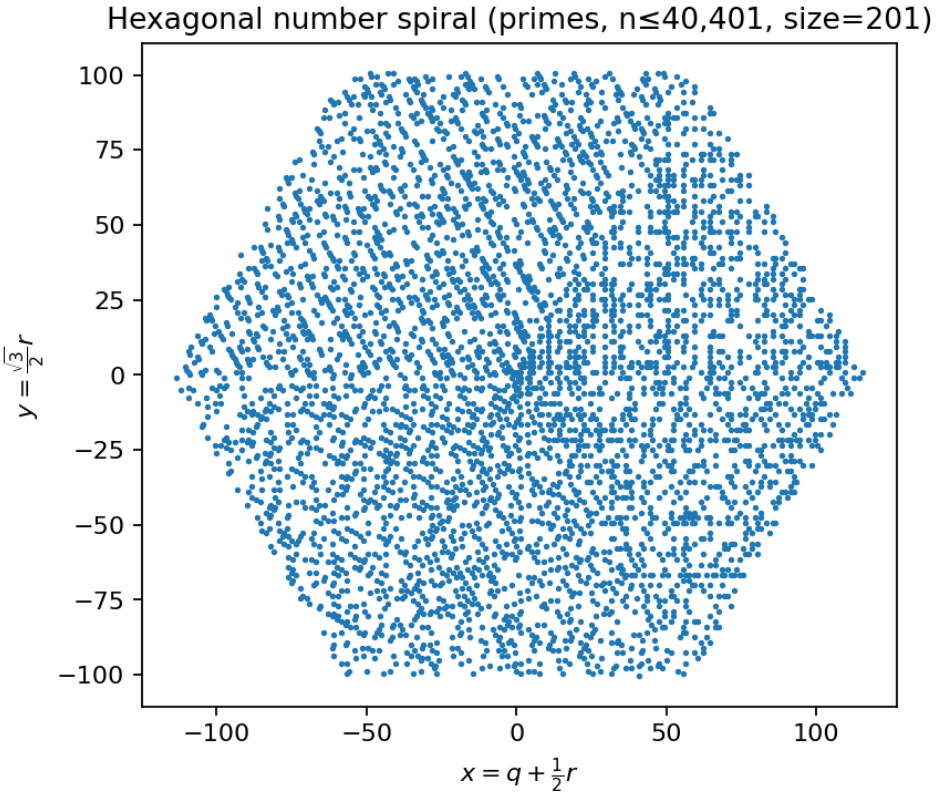
Tags: `number-theory`, `quantitative-exploration`, `visualization`
See: *Valid Tags*.

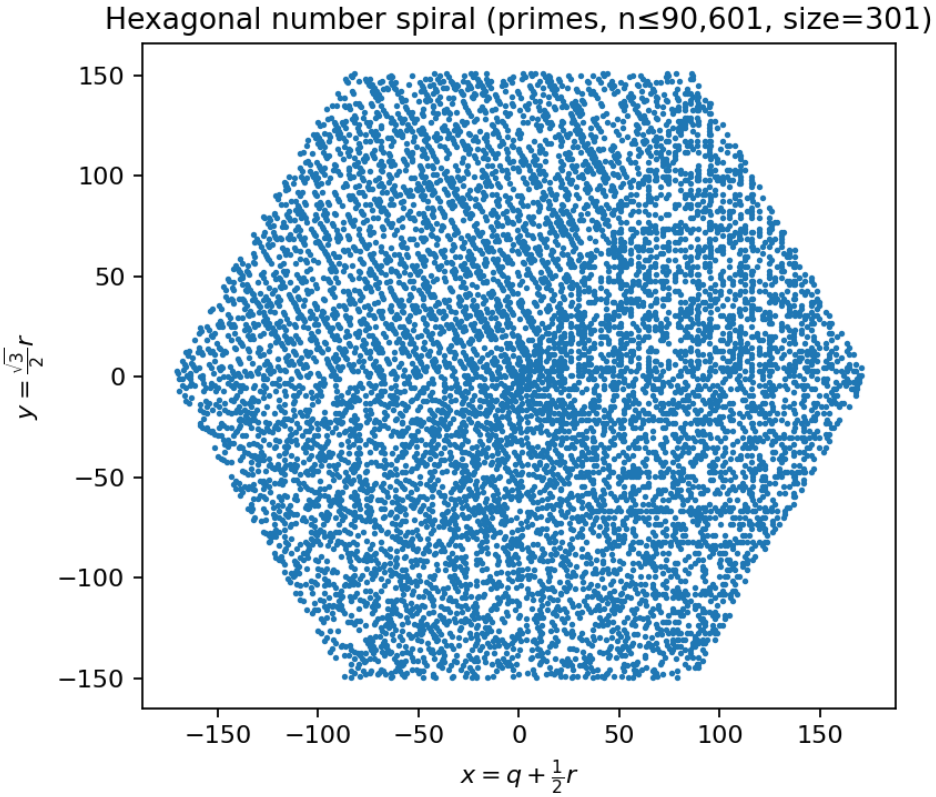
5.126.1 Highlights

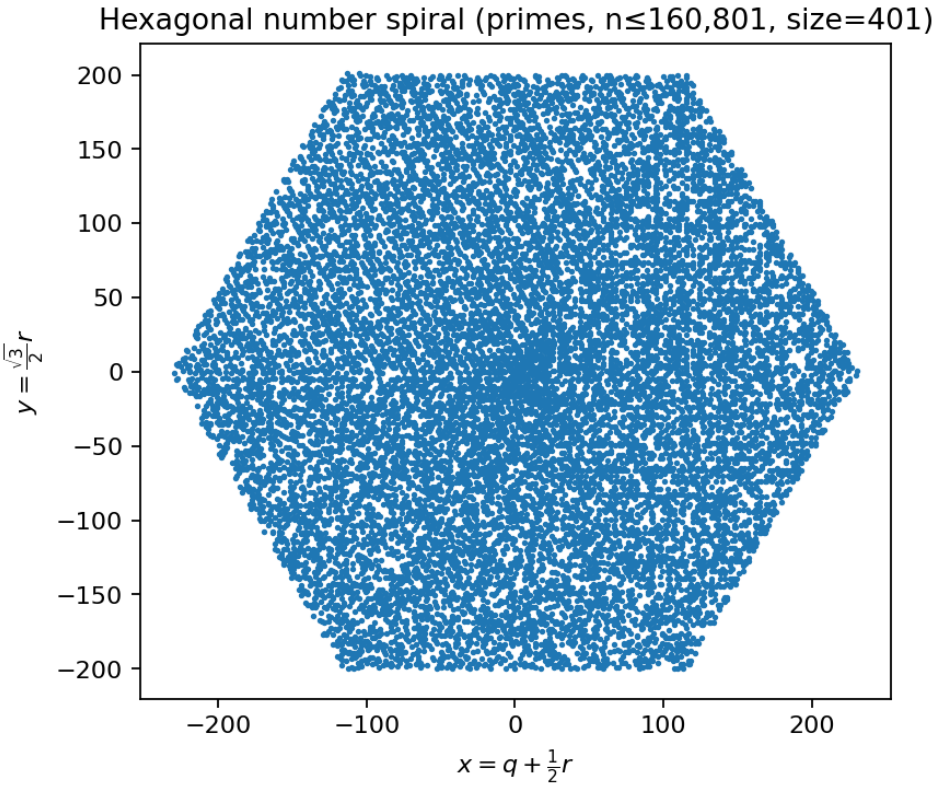
- Arranges integers on a **hexagonal lattice** using a centered spiral enumeration.
- Parameterizable run: vary `--size` from the command line (default: `301`).
- Writes reproducible artifacts (`params.json`, `report.md`, and figures) into `out/e126/`.

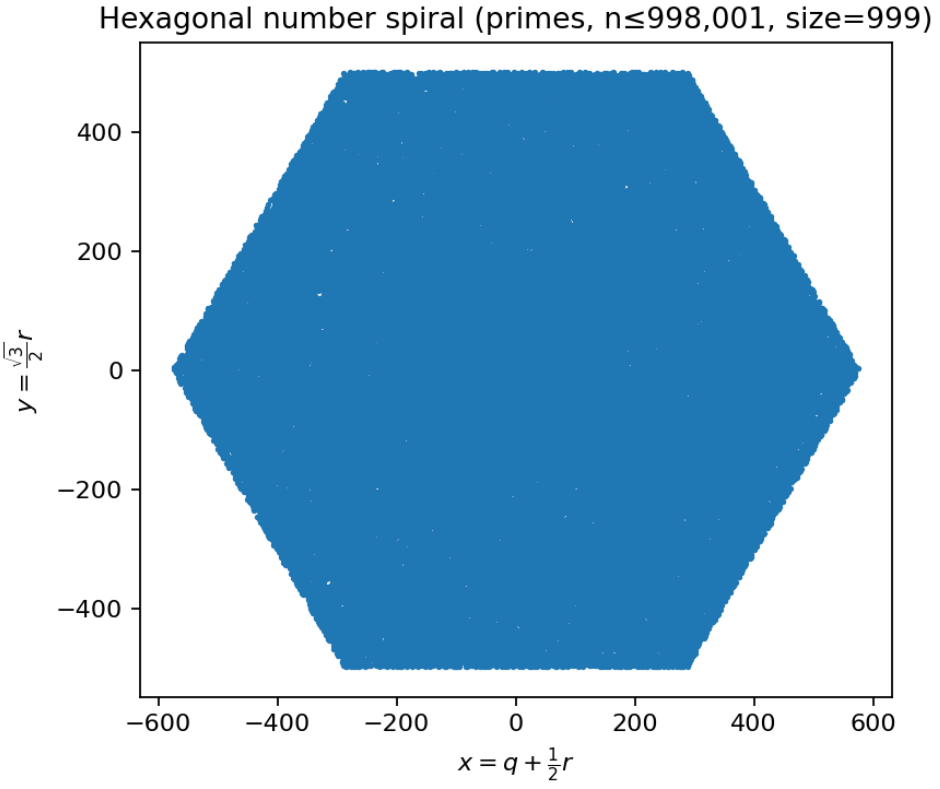
5.126.2 Goal

Render a **hexagonal number spiral** and highlight primes to reveal lattice-aligned structures, then compare how that structure changes as `size` increases.


Hexagonal number spiral (primes, n≤40,401, size=201)
100
75
50
25
0
−25
−50
−75
−100
−100
−50
0
50
100
$y = \frac{\sqrt{3}}{2} r$
$x = q + \frac{1}{2} r$


Hexagonal number spiral (primes, n≤90,601, size=301)
150
100
50
0
−50
−100
−150
−150
−100
−50
0
50
100
150
$y = \frac{\sqrt{3}}{2} r$
$x = q + \frac{1}{2} r$


Hexagonal number spiral (primes, n≤160,801, size=401)
200
150
100
50
0
−50
−100
−150
−200
−200
−100
0
100
200
$y = \frac{\sqrt{3}}{2} r$
$x = q + \frac{1}{2} r$


Hexagonal number spiral (primes, n≤998,001, size=999)
400
200
0
−200
−400
−600
−400
−200
0
200
400
600
$y = \frac{\sqrt{3}}{2} r$
$x = q + \frac{1}{2} r$

5.126.3 Background (quick refresher)

What is a hexagonal number spiral?

A **hexagonal number spiral** places integers on a hexagonal grid, starting at the center with 1 and then spiraling outward. Compared to the square-lattice Ulam spiral, the hex lattice emphasizes different geometric alignments and may reveal different-looking prime structures. See [Kurzweg, 2020].

Optional background pages

- *Prime numbers refresher*
- *Quadratic polynomials (algebraic) refresher*
- *Euler's prime-generating polynomial refresher*

5.126.4 Research question

Which geometric structures appear when primes are highlighted on a hexagonal spiral, and how stable are they as `size` increases (i.e., as we include integers up to size^2)?

5.126.5 Why this qualifies as a mathematical experiment

- **Finite procedure:** enumerate a finite spiral placement for integers $1, \ldots, \text{size}^2$ and run a deterministic primality test up to size^2.
- **Observable(s):** lattice-aligned streaks/clusters and how visibility changes with `size`.
- **Parameter space:** `size` (and, if you extend it, coordinate convention / rendering choices).
- **Outcome:** figures and a short report capturing the key observation and caveats.
- **Reproducibility:** outputs saved to `out/e126/` with a parameter snapshot.

5.126.6 Experiment design

- **Computation:** map integers $1, \ldots, \text{size}^2$ to hex-lattice coordinates using a centered spiral enumeration; mark primes.
- **Coordinates:** integers are mapped to axial hex coordinates (q, r) on a hex grid and embedded into the plane via $x = q + \frac{1}{2}r$ and $y = \frac{\sqrt{3}}{2}r$.
- **Prime classification:** primes are computed using a sieve up to size^2.
- **Outputs:** one or more scatter plots of prime points on the hex lattice (composites omitted).
- **Artifacts written:**
  - `figures/fig_01_hex_spiral*.png` (main figure) and `figures/e126_hero_<size>.png` (published hero)
  - `params.json`
  - `report.md`

5.126.7 How to run

```
make run EXP=e126
```

Override the experiment size:

```
make run EXP=e126 ARGS="--size 501"
```

Direct invocation (always works):

```
uv run --extra dev python -m mathxlab.experiments.e126 --out out/e126 --size 501
```

5.126.8 Notes / pitfalls

- Different conventions exist for hex-spiral enumeration; changing the convention can change visible structures.
- Larger `size` means more numbers (size^2) and a longer run time.
- "Looks-true" trap: lattice-aligned streaks do not prove a statement about primes.

5.126.9 Extensions

- Overlay ring boundaries (or distance-from-center contours) to relate features to the spiral geometry.
- Add simple quantitative overlays: density by hex-ring index or by lattice direction.
- Compare hex vs. square spirals at matched `N` (same number of integers).

5.126.10 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e126
```

Parameters

- size: `301` (odd); visualizes integers `1..90601`.

Notes

- Integers are placed by walking concentric hexagonal rings around the origin.
- Prime-rich lines/curves are expected analogues of the Ulam spiral phenomena.
- This experiment is deterministic; `seed` does not change the output.

params.json (snapshot)

```
{
  "size": 301
}
```

5.126.11 References

- Hex spiral background and construction ideas: [Kurzweg, 2020].
- Exposition connecting multiple prime visualizations: [Brockmann, 2019].
- Recreational-math context and "pattern traps": [Gardner, 1983], [Hoffman, 1989].

See also ../references.

5.126.12 Related experiments

- *E024: Ulam spiral structure* (Ulam spiral structure)
- *E124: Klauber triangle structure* (Klauber triangle structure)
- *E125: Sacks spiral structure* (Sacks spiral structure)

- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (Quadratic prime-run atlas ($n^2 + an + b$))
- *E128: Quadratic modular obstructions (Euler-type)* (Quadratic modular obstructions (Euler-type))
- *E129: Euler lucky constants for n^2 + n + b* (Euler lucky constants for $n^2 + n + b$)

5.126.13 Parameters (example)

- size: `301` (implied total integers: `1..90,601`)

Recommended `size` range

The hex spiral run includes integers $1, \ldots, \text{size}^2$, so runtime and memory scale roughly with size^2 (because about size^2 integers are classified as prime/not-prime).

A practical range that works well for most runs:

- **Minimum (still meaningful):** `size = 101`
- **Default / recommended:** `size = 301`
- **Comfortable upper range on typical laptops (varies):** `size = 999`
- **Above ~`size = 1501`:** expect noticeably higher runtime (and potentially memory pressure), depending on the machine and the primality implementation.

Rule of thumb: start with `301`, then try `501`, `701`, `901`. Increase further only if runtime remains acceptable.

5.126.14 Summary

We enumerate a hex-lattice spiral placement for the first 90,601 positive integers and highlight primes. At this scale, lattice-aligned structures can be visible: some directions show denser prime "streaks" than others.

5.126.15 Key observations

- Prime points show directional structure aligned to the hex lattice.
- Some structures are subtle and become easier to see with larger `size` and careful marker sizing.

5.126.16 Interpretation

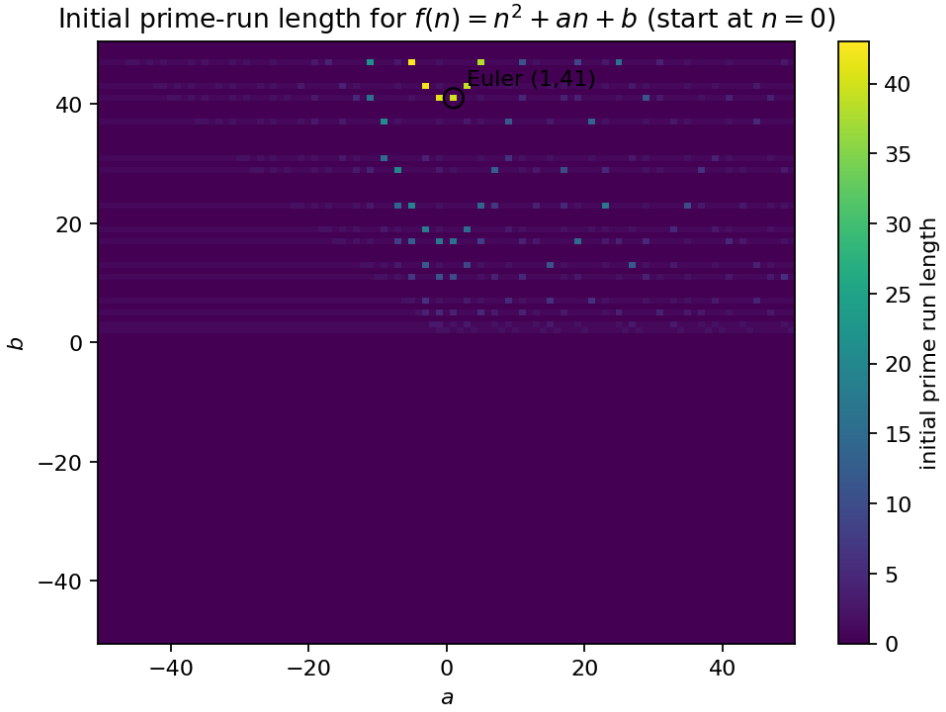
The hex spiral is an alternative geometry for indexing the same integers. As with the Ulam spiral, geometric placement can visually emphasize prime-rich progressions or polynomial-like sequences. This is a strong exploratory tool — but not evidence of a global law of primes.

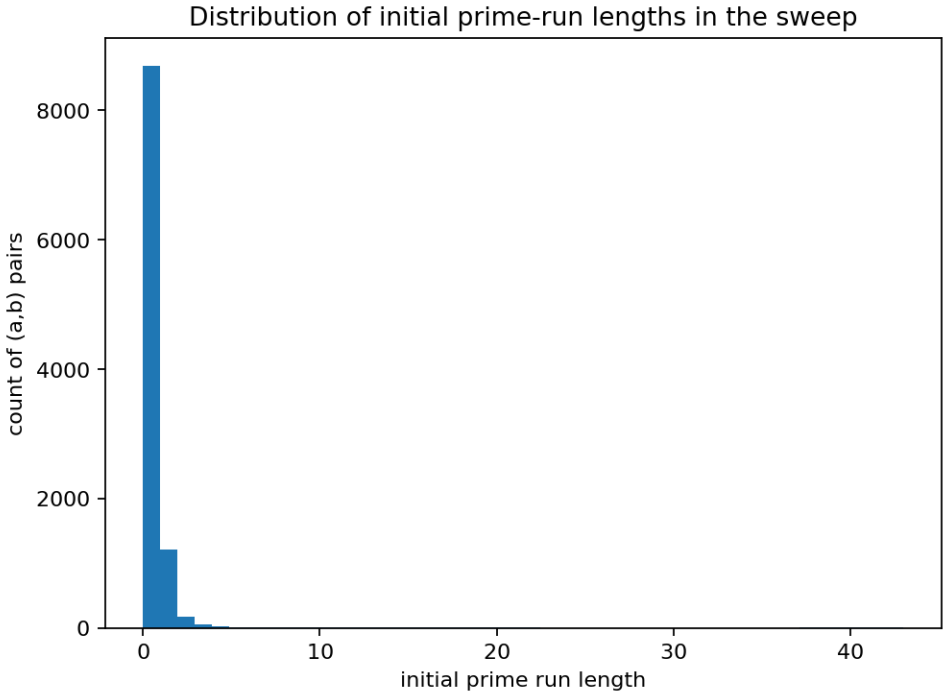
5.126.17 Caveats

- Different enumeration conventions change the picture.
- Finite window: patterns can look stronger/weaker depending on the chosen `size` and zoom level.
- Visualization choices matter: marker size, DPI, and axis limits can hide or exaggerate structure.

5.127 E127: Quadratic prime-run atlas ($n^2 + an + b$)

Tags: `number-theory`, `quantitative-exploration`, `visualization`, `primes`, `optimization` See: *Valid Tags*.


Initial prime-run length for $f(n) = n^2 + an + b$ (start at $n = 0$)
Euler (1,41)
40
20
0
−20
−40
b
−40
−20
0
20
40
a
40
35
30
25
20
15
10
5
0
initial prime run length


Distribution of initial prime-run lengths in the sweep
8000
6000
4000
2000
0
count of (a,b) pairs
0
10
20
30
40
initial prime run length

5.127.1 Highlights

- Heatmap of "how long primes last" for $f(n) = n^2 + an + b$ across a small (a,b) grid.
- Euler's $(a, b) = (1, 41)$ appears as a visible peak, but it is not unique.
- Turns a folklore fact into a reproducible, parameterized sweep with saved artifacts.

5.127.2 Goal

Explore how sensitive **initial prime streaks** are to the coefficients of a quadratic polynomial. We measure, for each (a,b), the largest L such that $f(0), f(1), \ldots, f(L-1)$ are all prime.

5.127.3 Background (quick refresher)

- *Quadratic polynomials (algebraic) refresher*
- *Euler's prime-generating polynomial refresher*
- *Prime numbers refresher*

5.127.4 Research question

Within a bounded coefficient grid, which quadratic polynomials produce the **longest initial prime runs**, and where does Euler's polynomial sit in that landscape?

5.127.5 Method

- Sweep integers $a \in [a_{\min}, a_{\max}]$ and $b \in [b_{\min}, b_{\max}]$.
- For each pair, evaluate $f(n)$ for $n = 0..N$ and record the initial prime-run length.
- Visualize run length as a heatmap and summarize the top candidates in the report.

Plot axes and coordinate conventions

The heatmap uses a 2D grid over the integer coefficients (a, b):

- **x-axis:** the coefficient a (horizontal direction)
- **y-axis:** the coefficient b (vertical direction)

If the sweep uses unit steps (the default), then each heatmap cell corresponds directly to an integer pair:

$$(a, b) \in \{a_{\min}, \ldots, a_{\max}\} \times \{b_{\min}, \ldots, b_{\max}\}.$$

More generally, if the heatmap array is indexed by (i, j) with $i = 0, \ldots, n_a - 1$ (columns) and $j = 0, \ldots, n_b - 1$ (rows), then the coefficients shown at that cell are:

- $$x = a = a_{\min} + i\,\Delta a$$
- $$y = b = b_{\min} + j\,\Delta b$$

For Matplotlib `imshow`, this is typically implemented via an `extent` so that tick labels match the coefficient values (e.g. `extent=(a_min-0.5, a_max+0.5, b_min-0.5, b_max+0.5)` for unit steps). If you flip the image origin (`origin="upper"`), the y-axis is inverted visually; in that case, interpret the y-axis tick labels accordingly.

5.127.6 How to run

```
make run EXP=e127
```

or:

```
uv run python -m mathxlab.experiments.e127
```

5.127.7 Outputs

This experiment follows the standard output contract:

- `out/e127/figures/` — generated figures (PNG)
- `out/e127/report.md` — short narrative report
- `out/e127/params.json` — run parameters (stable JSON)
- `out/e127/logs/` — run logs (created by the runner/Makefile)

5.127.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e127
```

Parameters

- a range: `[-50, 50]`
- b range: `[-50, 50]`
- n_run_max: `80` (test n=0..n_run_max)
- top_k: `12`

5.127.9 Key observation

Initial prime streaks vary sharply across (a,b). A few islands can look remarkably prime-rich on small ranges, which explains why Euler's famous polynomial stands out in short scans.

- Euler point (a=1,b=41) in this grid: run length **40** (n=0..39).

5.127.10 Best run lengths in this sweep

run length	a	b	polynomial
43	-5	47	$n^2 + -5n + 47$
42	-3	43	$n^2 + -3n + 43$
41	-1	41	$n^2 + -1n + 41$
40	1	41	$n^2 + 1n + 41$
39	3	43	$n^2 + 3n + 43$
38	5	47	$n^2 + 5n + 47$
22	-11	47	$n^2 + -11n + 47$
21	-9	37	$n^2 + -9n + 37$
20	-7	29	$n^2 + -7n + 29$
19	-5	23	$n^2 + -5n + 23$
18	-3	19	$n^2 + -3n + 19$
17	-1	17	$n^2 + -1n + 17$

5.127.11 Outputs

- `figures/fig_01_run_length_heatmap.png`
- `figures/fig_02_run_length_histogram.png`
- `params.json`
- `report.md`

params.json (snapshot)

```
{
  "a_max": 50,
  "a_min": -50,
  "b_max": 50,
  "b_min": -50,
  "n_run_max": 80,
  "seed": 1,
  "top_k": 12
}
```

5.127.12 References

- Euler's quadratic and "lucky numbers": contributors [2025], Weisstein [2025].
- Quadratic basics (discriminant, roots): Wikipedia contributors [2026].

5.127.13 Related experiments

- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Euler's polynomial: first counterexample witness)
- *E024: Ulam spiral structure* (Ulam spiral: prime-rich diagonals explained by quadratic forms)
- *E031: Admissibility and modular obstructions* (modular obstructions and admissibility)
- *E125: Sacks spiral structure* (Sacks spiral prime structure)
- *E126: Hexagonal number spiral structure* (hex-grid spiral prime structure)

5.128 E128: Quadratic modular obstructions (Euler-type)

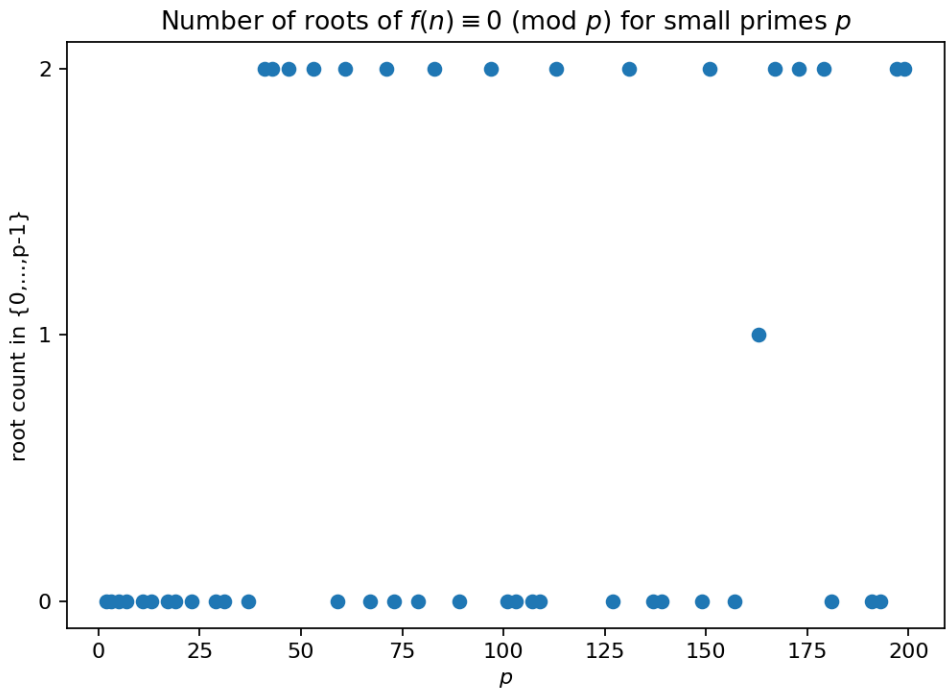
Tags: `number-theory`, `model-checking`, `counterexample-search`, `primes` See: *Valid Tags*.

5.128.1 Highlights

- Makes the "hidden reason" a prime streak must fail: modular obstructions.
- Lists explicit residue classes $n \bmod p$ that force divisibility of $f(n)$.
- Includes the classic witness $n = 41k$ for Euler's $n^2 + n + 41$.

5.128.2 Goal

Show how congruences explain why a "prime-generating polynomial" cannot stay prime forever. For each small prime modulus p, we compute the roots of $f(n) \equiv 0 \pmod{p}$.



5.128.3 Background (quick refresher)

- *Euler's prime-generating polynomial refresher*
- *Quadratic polynomials (algebraic) refresher*
- *Prime numbers refresher*

5.128.4 Research question

For Euler's $f(n) = n^2 + n + 41$, which small primes p divide some values $f(n)$, and which residue classes $n \bmod p$ generate those guaranteed composites?

5.128.5 Method

- Fix $f(n) = n^2 + an + b$ with $(a, b) = (1, 41)$.
- For each prime $p \le p_{\max}$, brute-force all residues $n \in \{0, \ldots, p-1\}$ and record solutions to $f(n) \equiv 0 \pmod{p}$.
- Plot the number of roots per prime and write a report table of residues.

5.128.6 How to run

```
make run EXP=e128
```

or:

```
uv run python -m mathxlab.experiments.e128
```

5.128.7 Outputs

This experiment follows the standard output contract:

- `out/e128/figures/` — generated figures (PNG)
- `out/e128/report.md` — short narrative report

- `out/e128/params.json` — run parameters (stable JSON)
- `out/e128/logs/` — run logs (created by the runner/Makefile)

5.128.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e128
```

Parameters

- a: `1`
- b: `41`
- p_max: `199`
- max_listed: `25`
- witness_k: `6`

5.128.9 Key observation

For any modulus p, the congruence f(n)≡0 (mod p) selects residue classes. Whenever n hits such a class, f(n) is divisible by p and therefore composite (unless f(n)=p).

5.128.10 A built-in infinite composite subsequence (Euler-style)

For f(n)=n^2+an+b and n=bk:

$$f(bk) = (bk)^2 + a(bk) + b = b\,(bk^2 + ak + 1).$$

So for k≥1, f(bk) is divisible by b (in absolute value).

k	n=bk	f(n)	divisible by
1	41	1763	41
2	82	6847	41
3	123	15293	41
4	164	27101	41
5	205	42271	41
6	246	60803	41

5.128.11 Roots modulo small primes

The table below lists primes p for which f(n)≡0 (mod p) has solutions.

p	root count	roots n (mod p)
41	2	0, 40
43	2	1, 41
47	2	2, 44
53	2	3, 49
61	2	4, 56
71	2	5, 65
83	2	6, 76
97	2	7, 89
113	2	8, 104
131	2	9, 121
151	2	10, 140
163	1	81
167	2	82, 84
173	2	11, 161
179	2	87, 91
197	2	12, 184
199	2	96, 102

5.128.12 Outputs

- `figures/fig_01_root_counts_mod_p.png`
- `params.json`
- `report.md`

params.json (snapshot)


```
{
  "a": 1,
  "b": 41,
  "max_listed": 25,
  "p_max": 199,
  "seed": 1,
  "witness_k": 6
}
```

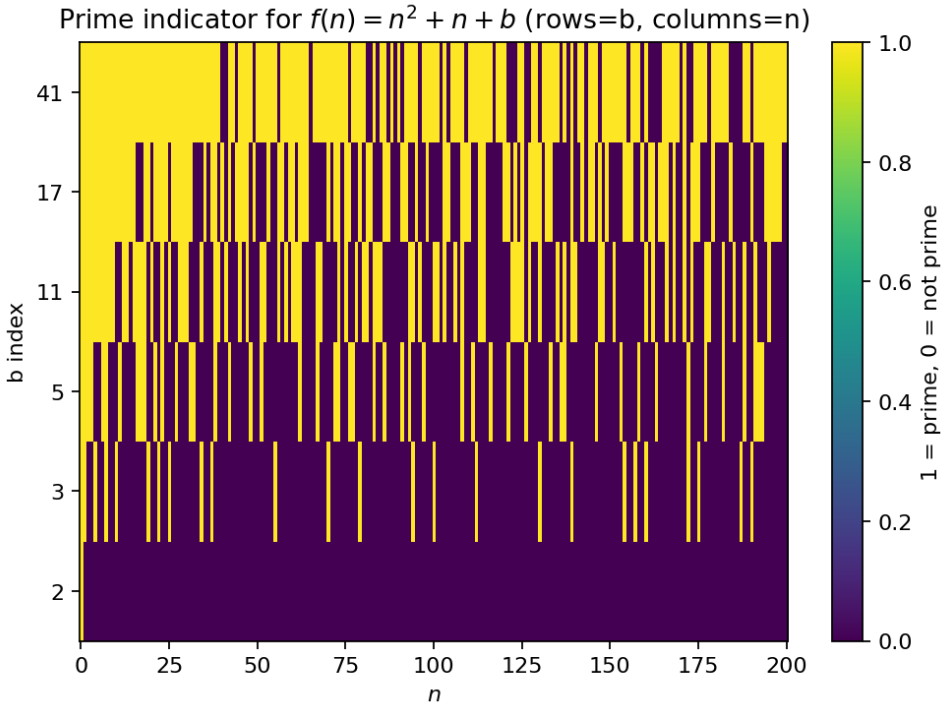

5.128.13 References

- Euler's quadratic and its folklore: contributors [2025], Weisstein [2025].
- Quadratic basics: Wikipedia contributors [2026].

5.128.14 Related experiments

- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (first explicit counterexample $f(40) = 41^2$)
- *E031: Admissibility and modular obstructions* (admissibility / modular obstructions in prime constellations)
- *E035: Primes in arithmetic progressions mod q* (primes in arithmetic progressions)
- *E038: Bertrand's postulate (computational verification)* (Bertrand's postulate: primes in intervals)

5.129 E129: Euler lucky constants for $n^2 + n + b$



Tags: `number-theory`, `quantitative-exploration`, `visualization`, `primes`, `heuristics` See: *Valid Tags*.

5.129.1 Highlights

- Compares several classical "lucky" constants b side-by-side.
- Heatmap shows prime/composite patterns across n for each b.
- Report includes the first failure n and a factorization witness.

5.129.2 Goal

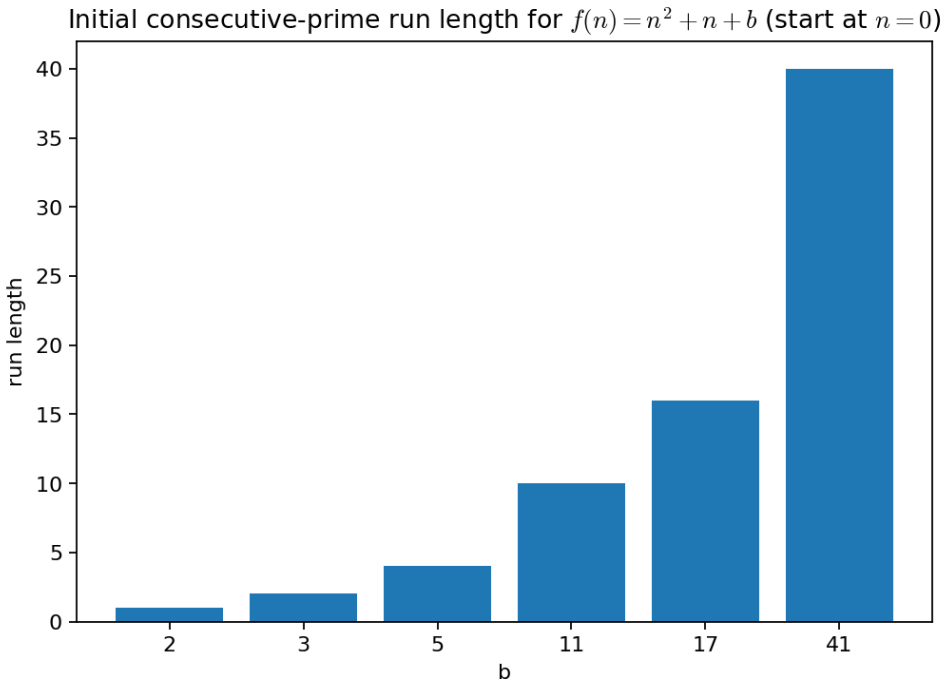
Compare how long $n^2 + n + b$ stays prime at the start for a small set of classical constants b. We quantify the initial prime streak length and visualize where it breaks.

5.129.3 Background (quick refresher)

- *Euler's prime-generating polynomial refresher*
- *Quadratic polynomials (algebraic) refresher*
- *Prime numbers refresher*

5.129.4 Research question

Among a short list of traditional "Euler lucky numbers" b, which yields the longest initial prime streak for $f(n) = n^2 + n + b$ and how do the failures look (first composite and its factors)?



5.129.5 Method

- Choose a list of constants b.
- For each b, evaluate $f(n)$ for $n = 0..N$.
- Use a sieve to classify values as prime/composite and measure the initial run length.
- Visualize a prime indicator map and a bar chart of run lengths.

5.129.6 How to run

```
make run EXP=e129
```

or:

```
uv run python -m mathxlab.experiments.e129
```

5.129.7 Outputs

This experiment follows the standard output contract:

- `out/e129/figures/` — generated figures (PNG)
- `out/e129/report.md` — short narrative report
- `out/e129/params.json` — run parameters (stable JSON)
- `out/e129/logs/` — run logs (created by the runner/Makefile)

5.129.8 Published run snapshot

If this experiment is included in the docs gallery, include the published snapshot (report + params).

Reproduce:

```
make run EXP=e129
```

Parameters

- n_max: `200`
- b_values: `[2, 3, 5, 11, 17, 41]`

5.129.9 Summary

b	initial prime run	first non-prime n	f(n)	factorization
2	1	1	4	2^2
3	2	2	9	3^2
5	4	4	25	5^2
11	10	10	121	11^2
17	16	16	289	17^2
41	40	40	1681	41^2

Notes

- Many quadratics look prime-rich on small ranges; a short streak does not imply a deep theorem.
- For any fixed b, there are always modular obstructions (e.g. the b-multiple subsequence for n=bk when b is prime).

5.129.10 Outputs

- `figures/fig_01_prime_indicator_heatmap.png`
- `figures/fig_02_initial_run_lengths.png`
- `params.json`
- `report.md`

params.json (snapshot)

```
{
  "b_values": [
    2,
    3,
    5,
    11,
    17,
    41
  ],
  "max_listed": 10,
  "n_max": 200,
  "seed": 1
}
```

5.129.11 References

- Lucky numbers and Euler's polynomial: contributors [2025], Weisstein [2025].

5.129.12 Related experiments

- *E013: Prime-polynomial counterexamples (Euler's n^2 + n + 41)* (Euler's $n^2 + n + 41$ counterexample witness)
- *E127: Quadratic prime-run atlas (n^2 + a n + b)* (full (a,b) atlas for $n^2 + an + b$)

- *E128: Quadratic modular obstructions (Euler-type)* (modular obstructions that force failures)
- *E019: Prime counting and a PNT baseline* (prime density and "random prime" heuristics)

CHAPTER

SIX

EXPERIMENT STATUS

This page tracks the generation and manual editing status of experiment pages.

Experiment	Generated	Last edited	Notes
e001_taylor_error_landscapes	31.12.2025		
e002_even_perfect_growth	31.12.2025		
e003_abundancy_index_landscape	31.12.2025		
e004_sigma_benchmark	31.12.2025		
e005_odd_perfect_filter_pipeline	31.12.2025		
e006_near_misses	31.12.2025		
e007_mersenne_growth	31.12.2025		
e008_lucas_lehmer_scan	31.12.2025		
e009_small_factor_scan	31.12.2025		
e010_perfect_numbers_from_mersenne	31.12.2025		
e011_mersenne_prime_heuristic	31.12.2025		
e012_fermat_pseudoprimes	31.12.2025		
e013_prime_polynomial_counterexample	31.12.2025		
e014_primorial_pm1_counterexamples	31.12.2025		
e015_wilson_test_infeasibility	31.12.2025		
e016_trial_division_vs_mr	31.12.2025		
e017_sieve_memory_segmented	31.12.2025		
e018_mr_base_choice_counterexamples	31.12.2025		
e019_prime_density_pnt	31.12.2025	11.01.2026	
e020_pi_vs_li_numeric	31.12.2025		
e021_pi_explicit_bounds_sanity	31.12.2025		
e022_prime_race_mod4	31.12.2025		
e023_residue_classes_modq	31.12.2025		
e024_ulam_spiral	31.12.2025	11.01.2026	
e025_prime_gaps_nonmonotone	31.12.2025		
e026_normalized_prime_gaps	31.12.2025		
e027_record_gaps_vs_log2	31.12.2025		
e028_jumping_champions	31.12.2025		
e029_twin_primes_heuristic	31.12.2025		
e030_cousin_sexy_pairs	31.12.2025		
e031_admissibility_mod_obstructions	31.12.2025		
e032_triplets_quadruplets_counts	31.12.2025		
e033_bounded_gaps_vs_twins	31.12.2025		
e034_twin_window_variance	31.12.2025		
e035_primes_in_ap	31.12.2025		
e036_prime_ap_search	31.12.2025		
e037_prime_free_intervals_factorial	31.12.2025		
e038_bertrand_postulate	31.12.2025		
e039_sophie_germain_safe_primes	31.12.2025		
e040_palindromic_primes_11	31.12.2025		

continues on next page

Table 1 – continued from previous page

Experiment	Generated	Last edited	Notes
e041_fermat_numbers_counterexample	31.12.2025		
e042_repunit_primes_scan	31.12.2025		
e043_pollard_rho_variability	31.12.2025		
e044_ss_vs_mr	31.12.2025		
e045_mr_deterministic_64bit	31.12.2025		
e046_prime_testing_pipeline	31.12.2025		
e047_fermat_numbers	31.12.2025		
e048_carmichael_numbers	31.12.2025		
e049_wieferich_primes	31.12.2025		
e050_primorials_euclid_numbers	31.12.2025		
e051_semiprimes_factorization	31.12.2025		
e052_totient_ratio_landscape	31.12.2025		
e053_inverse_totient_multiplicity	31.12.2025		
e054_mobius_squarefree_density	31.12.2025		
e055_mertens_random_walk	31.12.2025		
e056_liouville_vs_mertens	31.12.2025		
e057_erdos_kac_omega_distribution	31.12.2025		
e058_divisor_count_records	31.12.2025		
e059_abundancy_sigma_over_n	31.12.2025		
e060_jordan_totient_family	31.12.2025		
e061_chebyshev_psi_prime_powers	31.12.2025		
e062_carmichael_lambda_vs_phi	31.12.2025		
e063_dirichlet_convolution_identities	31.12.2025		
e064_dirichlet_character_tables	31.12.2025		
e065_dirichlet_orthogonality	31.12.2025		
e066_character_partial_sums	31.12.2025		
e067_gauss_sums	31.12.2025		
e068_lseries_vs_euler_product	31.12.2025		
e069_l1_convergence	31.12.2025		
e070_primes_in_progressions_counts	31.12.2025		
e071_progressions_pnt_error	31.12.2025		
e072_prime_race_mod4	31.12.2025		
e073_prime_race_mod3	31.12.2025		
e074_prime_race_mod8_leaderboard	31.12.2025		
e075_race_statistic_distribution	31.12.2025		
e076_theta_in_progressions	31.12.2025		
e077_indicator_via_characters	31.12.2025		
e078_max_character_sums	31.12.2025		
e079_primitive_characters_conductors	31.12.2025		
e080_bias_fraction_curve	31.12.2025		
e081_sign_changes_table	31.12.2025		
e082_zeta_series_convergence	31.12.2025		
e083_series_vs_euler_product	31.12.2025		
e084_critical_line_magnitude	31.12.2025		
e085_eta_acceleration	31.12.2025		
e086_hardy_Z_near_zeros	31.12.2025		
e087_gram_points	31.12.2025		
e088_zero_counting_rvm	31.12.2025		
e089_zeta_heatmap_logabs	31.12.2025		
e090_functional_equation_residual	31.12.2025		
e091_euler_product_critical_line	31.12.2025		
e092_mobius_series_inverse_zeta	31.12.2025		
e093_log_derivative_von_mangoldt	31.12.2025		
e094_omega_vs_bigomega	31.12.2025		
e095_squarefree_omega_equals_bigomega	31.12.2025		

continues on next page

Table 1 – continued from previous page

Experiment	Generated	Last edited	Notes
e096_tau_record_holders	31.12.2025		
e097_sigma_over_n_abundance	31.12.2025		
e098_sigma_over_n_alpha_extremals	31.12.2025		
e099_jordan_totient_atlas	31.12.2025		
e100_carmichael_lambda_vs_phi	31.12.2025		
e101_reduced_residues_structure	31.12.2025		
e102_convolution_identities	31.12.2025		
e103_chebyshev_psi_prime_powers	31.12.2025		
e104_von_mangoldt_statistics	31.12.2025		
e105_mertens_function_scaling	31.12.2025		
e106_dirichlet_characters_real_vs_complex	31.12.2025		
e107_dirichlet_conductor_histogram	31.12.2025		
e108_dirichlet_orthogonality_heatmap	31.12.2025		
e109_gauss_sums_prime_modulus	31.12.2025		
e110_L_series_partial_sums	31.12.2025		
e111_L_euler_product_vs_series	31.12.2025		
e112_prime_race_modq	31.12.2025		
e113_first_prime_per_residue	31.12.2025		
e114_zeta_eta_truncation_error	31.12.2025		
e115_hardy_Z_sign_changes_scan	31.12.2025		
e116_zero_count_vs_rvm	31.12.2025		
e117_functional_equation_check	31.12.2025		
e118_euler_product_breakdown	31.12.2025		
e119_psi_minus_x_smoothed	31.12.2025		
e120_pretentious_distance_proxy	31.12.2025		
e121_multiplicativity_stress_tests	31.12.2025		
e122_arithmetic_function_atlas	31.12.2025		
e123_arithmetic_correlation_matrix	31.12.2025		
e124_klauber_triangle	01.01.2026	11.01.2026	
e125_sacks_spiral	01.01.2026	11.01.2026	
e126_hexagonal_number_spiral	01.01.2026	11.01.2026	
e127_quadratic_prime_streak_atlas	02.01.2026		
e128_quadratic_modular_obstructions	02.01.2026		
e129_euler_lucky_constants	02.01.2026		

CHAPTER

SEVEN

BACKGROUND

This section provides mathematical foundations for the experiments.

7.1 Cheat sheet

Quick, experiment-oriented reminders of core concepts and common pitfalls. Entries are kept **alphabetical by topic title** so this page stays easy to scan. Each topic is intentionally short and points to the relevant background pages and references.

7.1.1 Chebyshev functions $\theta(x)$ and $\psi(x)$

- $\theta(x) = \sum_{p \le x} \log p$ (primes only).
- $\psi(x) = \sum_{n \le x} \Lambda(n) = \sum_{p^k \le x} \log p$ (includes prime powers).
- Weighted counts are often smoother than $\pi(x)$ and connect more directly to analytic theory.

See also: *von Mangoldt \Lambda(n) and Chebyshev functions refresher*. Refs: [Apostol, 1976, Montgomery and Vaughan, 2006].

7.1.2 Congruent integers

- $a \equiv b \pmod{m}$ means $m \mid (a - b)$.
- Congruences are statements about residue classes, not about a unique representative.

See also: *Divisibility and modular arithmetic (Phase 2 core)*. Refs: [Hardy and Wright, 2008].

7.1.3 Dirichlet character $\chi \bmod q$

- A Dirichlet character χ modulo q is periodic mod q, completely multiplicative, and $\chi(n) = 0$ when $\gcd(n, q) > 1$.
- The **principal character** χ_0 satisfies $\chi_0(n) = 1$ for $\gcd(n, q) = 1$ and 0 otherwise.
- Values lie on the unit circle (for units) and encode multiplicative structure of residues mod q.

See also: *Dirichlet characters refresher*. Refs: [Davenport, 2000, Niven *et al.*, 1991].

7.1.4 Dirichlet L-function $L(s, \chi)$

- For $\Re(s) > 1$: $L(s, \chi) = \sum_{n \ge 1} \chi(n) n^{-s} = \prod_{p \nmid q} (1 - \chi(p) p^{-s})^{-1}$.
- The Euler product highlights the prime connection; the series is often easier numerically.
- Near $s = 1$, naive truncation can converge slowly; smoothing/damping is common in experiments.

See also: *Dirichlet L-functions refresher*. Refs: [Davenport, 2000, Montgomery and Vaughan, 2006].

7.1.5 Dirichlet's theorem and PNT(AP) (experiment baseline form)

- If $\gcd(a, q) = 1$, then there are infinitely many primes $p \equiv a \pmod{q}$ (Dirichlet).
- For fixed q and $\gcd(a, q) = 1$: $\pi(x; q, a) \sim \mathrm{li}(x)/\varphi(q)$ (PNT in AP).
- Error term used in Phase 2 plots: $E(x; q, a) = \pi(x; q, a) - \mathrm{li}(x)/\varphi(q)$.

See also: *Dirichlet's theorem and PNT(AP) in the form used by the experiments*. Refs: [Iwaniec and Kowalski, 2004, Lejeune Dirichlet, 1837].

7.1.6 Euler's totient $\varphi(q)$

- $\varphi(q)$ counts invertible residue classes: $\varphi(q) = \#(\mathbb{Z}/q\mathbb{Z})^\times$.
- If $q = \prod p_i^{k_i}$ then $\varphi(q) = q \prod_i (1 - 1/p_i)$.
- In Phase 2, $1/\varphi(q)$ is the "fair share" factor for residues in PNT(AP) baselines.

See also: *Euler's totient function \varphi(n) refresher*, *Divisibility and modular arithmetic (Phase 2 core)*. Refs: [Hardy and Wright, 2008].

7.1.7 Extended Euclidean algorithm (egcd)

- Computes integers (x, y) with $ax + by = \gcd(a, b)$.
- Key for modular inverses: if $\gcd(a, m) = 1$ then $ax \equiv 1 \pmod{m}$.

See also: *Divisibility and modular arithmetic (Phase 2 core)*. Refs: [Niven *et al.*, 1991].

7.1.8 Failure cases (common pitfalls)

- Forgetting to restrict to $\gcd(a, q) = 1$ when discussing PNT(AP) or prime races.
- Mixing sampling grids (linear vs log) across related race experiments.
- Comparing truncated Euler products with different prime cutoffs (not comparable).
- Treating "bias" on a finite range as a theorem; it is an observed phenomenon at that scale.

See also: *Prime number races refresher*, *Dirichlet's theorem and PNT(AP) in the form used by the experiments*.

7.1.9 Gauss sum $\tau(\chi)$

- Basic Gauss sum: $\tau(\chi) = \sum_{a=0}^{q-1} \chi(a) \exp(2\pi i a/q)$.
- For primitive χ, the magnitude satisfies $|\tau(\chi)| = \sqrt{q}$ (key Phase 2 pattern).
- Often interpreted as a finite Fourier transform of the character.

See also: *Gauss sums refresher*. Refs: [Berndt *et al.*, 1998, Davenport, 2000].

7.1.10 Greatest common divisor (gcd)

- $\gcd(a, b)$ is the largest integer dividing both.
- $\gcd(a, q) = 1$ is the "invertible mod q" condition used throughout Phase 2.

See also: *Divisibility and modular arithmetic (Phase 2 core)*. Refs: [Hardy and Wright, 2008].

7.1.11 Modular exponentiation (`pow`, square-and-multiply)

- Compute $a^e \bmod m$ efficiently in $O(\log e)$ multiplications.
- In Python: `pow(a, e, m)` (requires $e \geq 0$, and $m > 0$ in practice).

See also: *Primality testing: guarantees, error bounds, and what to report*.

7.1.12 Modular inverse

- The inverse of a mod m exists iff $\gcd(a, m) = 1$.
- In Python 3.8+: `pow(a, -1, m)` returns the inverse (raises if none exists).

See also: *Divisibility and modular arithmetic (Phase 2 core)*.

7.1.13 Modulo

- "n mod m" is a representative of the residue class of n modulo m.
- Be consistent about representatives (usually $0, 1, \ldots, m-1$) when indexing arrays/plots.

See also: *Divisibility and modular arithmetic (Phase 2 core)*.

7.1.14 Orthogonality (Dirichlet characters)

- Over the reduced residues, characters satisfy discrete orthogonality relations.
- Practical interpretation: averaging over characters can isolate a specific residue-class signal.
- Used as a correctness check (character tables and indicator reconstructions).

See also: *Dirichlet characters refresher*. Refs: [Davenport, 2000, Montgomery and Vaughan, 2006].

7.1.15 Prime counting approximations ($x/\log x$, $\mathrm{li}(x)$)

- $x/\log x$ is a rough first-order approximation to $\pi(x)$.
- $\mathrm{li}(x)$ is the standard smooth baseline used in the experiments (and in PNT(AP) baselines).

See also: *Prime counting approximations: (x), Li(x), and R(x)*. Refs: [Apostol, 1976].

7.1.16 Prime counting function $\pi(x)$

- $\pi(x) = \#\{p \le x : p \text{ prime}\}$.
- For progressions: $\pi(x; q, a)$ counts primes $\le x$ with $p \equiv a \pmod{q}$.

See also: *Prime numbers refresher*, *Dirichlet's theorem and PNT(AP) in the form used by the experiments*.

7.1.17 Prime counting in residue classes $\pi(x; q, a)$

- Definition: $\pi(x; q, a) = \#\{p \le x : p \equiv a \pmod{q}\}$.
- Only reduced residues ($\gcd(a, q) = 1$) participate in equidistribution statements.
- Error term used in plots: $E(x; q, a) = \pi(x; q, a) - \mathrm{li}(x)/\varphi(q)$.

See also: *Dirichlet's theorem and PNT(AP) in the form used by the experiments*.

7.1.18 Prime definition

- A prime is an integer $p \ge 2$ with exactly two positive divisors: 1 and p.
- In experiments, always specify whether you include $p = 2$ separately when using odd-only sieves/tests.

See also: *Prime numbers refresher*.

7.1.19 Prime mask / sieve idea

- A sieve produces a boolean mask `is_prime[n]` for $0 \le n \le N$.
- Masks support fast bulk computations: prime lists, residue-class counts, race differences.

See also: *Primality testing: guarantees, error bounds, and what to report*, *Prime numbers refresher*.

7.1.20 Prime race difference $\Delta(x; q, a, b)$

- Define $\Delta(x; q, a, b) = \pi(x; q, a) - \pi(x; q, b)$.
- Because baselines cancel, races compare error terms implicitly: $\Delta = E(x; q, a) - E(x; q, b)$.
- "Leader fraction" depends on sampling: a log grid weights decades more evenly than a linear grid.

See also: *Prime number races refresher*, *Dirichlet's theorem and PNT(AP) in the form used by the experiments*. Refs: [Granville and Martin, 2006, Rubinstein and Sarnak, 1994].

7.1.21 Reduced residue system (units mod q)

- $(\mathbb{Z}/q\mathbb{Z})^\times$ is the set of residues a mod q with $\gcd(a, q) = 1$.
- It has size $\varphi(q)$ and is the domain on which characters behave multiplicatively.

See also: *Divisibility and modular arithmetic (Phase 2 core)*, *Dirichlet characters refresher*.

7.1.22 Residue class

- A residue class modulo m is the set $\{n : n \equiv a \pmod{m}\}$ for some a.
- In code, we store a representative (often $a \in \{0, \ldots, m-1\}$) but the math object is the class.

See also: *Divisibility and modular arithmetic (Phase 2 core)*.

7.1.23 Sampling choices (linear vs log)

- Linear-in-x sampling overweights large x values in "time-in-lead" style statistics.
- Log-in-x sampling gives each decade similar weight and is usually more stable for race distributions.

See also: *Prime number races refresher*, *Exploratory visualizations for arithmetic functions*.

7.1.24 von Mangoldt function $\Lambda(n)$

- $\Lambda(n) = \log p$ if $n = p^k$ for some prime p and $k \geq 1$, else 0.
- Summing $\Lambda(n)$ up to x yields $\psi(x)$ (Chebyshev's second function).

See also: *von Mangoldt \Lambda(n) and Chebyshev functions refresher*. Refs: [Apostol, 1976].

7.2 Arithmetic functions refresher

This page is a *beginner-friendly* refresher for experiments about **arithmetic functions** (i.e. functions $f : \mathbb{N} \to \mathbb{C}$ that encode number-theoretic structure).

For deeper treatments, see [Apostol, 1976], [Niven *et al.*, 1991], and [Tenenbaum, 2015].

7.2.1 What "arithmetic function" usually means

An **arithmetic function** is any function of a positive integer n. Common themes:

- prime factorization drives the behavior of many functions
- *multiplicativity* lets you reduce to prime powers
- *averages* and *summatory functions* reveal global structure

Examples you will meet often:

- Euler's totient $\varphi(n)$ (coprime residues)
- Möbius $\mu(n)$ and Mertens $M(x) = \sum_{n \leq x} \mu(n)$
- divisor counts $d(n)$ and sums $\sigma_k(n)$

- prime-factor counting $\omega(n), \Omega(n)$
- von Mangoldt $\Lambda(n)$ (prime powers)
- Carmichael's $\lambda(n)$ (group exponent mod n)

7.2.2 Multiplicative vs. additive

Multiplicative

An arithmetic function f is **multiplicative** if

$$f(1) = 1, \qquad f(ab) = f(a)f(b) \text{ whenever } \gcd(a, b) = 1.$$

If $n = \prod p_i^{\alpha_i}$ then multiplicativity gives

$$f(n) = \prod f(p_i^{\alpha_i}).$$

Typical: $\varphi, \mu, \sigma_k, d, J_k, \lambda$.

Additive

A function g is **additive** if

$$g(ab) = g(a) + g(b) \text{ whenever } \gcd(a, b) = 1,$$

and **completely additive** if the relation holds for all a, b (no gcd condition). Typical: $\Omega(n)$ is completely additive; $\omega(n)$ is additive.

7.2.3 Why averages matter

Many experiments compare:

- pointwise behavior of $f(n)$
- cumulative behavior $\sum_{k \le n} f(k)$
- distribution of $f(n)$ over ranges

Analytic/probabilistic number theory focuses on these averages; see [Montgomery and Vaughan, 2006] and [Tenenbaum, 2015].

7.2.4 Common experiment patterns

- **Value distribution:** histograms of $f(n)$ for $n \le N$
- **Scatter vs. factorization features:** compare $f(n)$ with $\log n$, $\omega(n)$, etc.
- **Normal order:** show "typical size" vs. rare extremes (e.g. Erdős–Kac behavior)
- **Summatory oscillations:** $M(x)$, $\sum_{n \le x} \lambda(n)$, etc.
- **Extremal orders:** highly composite numbers maximize $d(n)$ (see [Ramanujan, 1915])

7.2.5 Experiments in this repository

- **E121** — Multiplicativity stress tests across core arithmetic functions.
- **E123** — Correlation matrix of arithmetic functions on 1..N (empirical relationships).

7.3 Average orders and the Erdős–Kac viewpoint

Many arithmetic functions are "wild" pointwise but have predictable average behavior. This page gives a short refresher on *average order*, *normal order*, and the Erdős–Kac phenomenon. See [Tenenbaum, 2015] and [Erdős and Kac, 1940].

7.3.1 Average order vs. normal order

- **Average order:** a function $A(x)$ such that

$$\sum_{n \le x} f(n) \approx \sum_{n \le x} A(n)$$

  or f has mean value described by A.

- **Normal order:** a function $N(n)$ such that "most" integers satisfy

$$f(n) \approx N(n).$$

A classic example: for most n, $\omega(n)$ is close to $\log \log n$.

7.3.2 Erdős–Kac theorem (informal statement)

Let $\omega(n)$ be the number of distinct prime factors. Erdős and Kac proved that the normalized variable

$$\frac{\omega(n) - \log \log n}{\sqrt{\log \log n}}$$

has an approximately standard normal distribution over $n \le x$ as $x \to \infty$. [Erdős and Kac, 1940]

7.3.3 Experiment ideas

- for a chosen N, compute $\omega(n)$ for $n \le N$ and histogram the normalized values
- compare the empirical mean/variance with $\log \log N$
- explore how the fit improves as N grows

7.3.4 Experiments in this repository

- **E094** — Erdős–Kac side-by-side for and Ω (normal order / scaling).

7.4 Carmichael numbers

A Carmichael number is a composite n such that

$$a^{n-1} \equiv 1 \pmod{n}$$

for every integer a coprime to n. They are **absolute Fermat pseudoprimes**, so they defeat the naive Fermat primality test. [Prime Pages (UTM), 2025, Wikipedia contributors, 2025]

7.4.1 Key facts

- **Korselt's criterion (1899):** n is Carmichael iff (i) n is squarefree and (ii) for every prime $p \mid n$, we have $(p-1) \mid (n-1)$. [Conrad, 2004, Wikipedia contributors, 2025]
- **At least three prime factors:** No Carmichael number is the product of only two primes. [Wikipedia contributors, 2025]
- **Infinitely many:** Alford–Granville–Pomerance proved there are infinitely many Carmichael numbers. [Alford *et al.*, 1994]
- **Historical origin:** Carmichael studied these numbers in the early 20th century. [Carmichael, 1912]

7.4.2 What to experiment with

- **Counterexample search:** Find Carmichael numbers by generating squarefree products and testing Korselt's criterion.
- **"Fermat test is not enough":** Show that all bases a with $\gcd(a, n) = 1$ pass Fermat for a Carmichael n, while a Miller–Rabin test quickly finds witnesses.
- **Distribution experiments:** Count Carmichael numbers up to X and compare growth to simple heuristics.
- **Construction families:** Implement Chernick-style constructions and see when factors are prime.

7.4.3 References

See Alford *et al.* [1994], Conrad [2004], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

7.5 Carmichael's $\lambda(n)$ function refresher

Carmichael's function $\lambda(n)$ is the exponent of the multiplicative group modulo n. It refines Euler's theorem by giving the *smallest* universal exponent. See [Erdős *et al.*, 1991] and [Niven *et al.*, 1991].

7.5.1 Definition

Let $(\mathbb{Z}/n\mathbb{Z})^\times$ be the multiplicative group of units modulo n. The **exponent** of a finite group G is lcm of the orders of all elements. Carmichael's function $\lambda(n)$ is the exponent of $(\mathbb{Z}/n\mathbb{Z})^\times$.

Equivalently, $\lambda(n)$ is the smallest positive integer such that for all integers a with $\gcd(a, n) = 1$:

$$a^{\lambda(n)} \equiv 1 \pmod{n}.$$

Always $\lambda(n) \mid \varphi(n)$.

7.5.2 Prime power formula (useful in code)

For odd primes p and $k \geq 1$:

$$\lambda(p^k) = \varphi(p^k) = p^{k-1}(p-1).$$

For 2^k:

$$\lambda(2) = 1, \quad \lambda(4) = 2, \quad \lambda(2^k) = 2^{k-2} \text{ for } k \geq 3.$$

For general $n = \prod p_i^{\alpha_i}$:

$$\lambda(n) = \operatorname{lcm}\big(\lambda(p_i^{\alpha_i})\big).$$

7.5.3 Why it is interesting experimentally

- highlights differences between "group size" (φ) and "group exponent" (λ)
- connects to orders modulo n and to cryptographic-style modular arithmetic
- has rich average-order results and rare extreme behavior (see [Erdős *et al.*, 1991])

7.5.4 Experiment ideas

- compare $\lambda(n)$ vs. $\varphi(n)$ on ranges
- visualize distribution of $\varphi(n)/\lambda(n)$
- record-breakers for large $\varphi(n)/\lambda(n)$

7.5.5 Experiments in this repository

- **E100** — Carmichael $\lambda(n)$ vs Euler $\varphi(n)$: ratios, typical size, and extremes.

7.6 Dirichlet characters refresher

This page is a *lightweight* background for experiments about primes in residue classes, Dirichlet L-functions, and prime races.

7.6.1 Core definitions

Fix an integer modulus $q \geq 1$. A **Dirichlet character modulo** q is a function $\chi : \mathbb{Z} \to \mathbb{C}$ such that:

1. (Periodicity) $\chi(n+q) = \chi(n)$ for all n.
2. (Multiplicativity) $\chi(mn) = \chi(m)\chi(n)$ for all m, n.
3. (Support) $\chi(n) = 0$ if $\gcd(n, q) > 1$.
4. For $\gcd(n, q) = 1$, we have $|\chi(n)| = 1$ (so values are roots of unity).

The **principal character** χ_0 modulo q is:

$$\chi_0(n) = \begin{cases} 1, & \gcd(n,q) = 1, \\ 0, & \gcd(n,q) > 1. \end{cases}$$

A character is **primitive** if it does not "factor through" a smaller modulus (its conductor equals q).

Orthogonality (the workhorse identity)

Let χ, ψ be Dirichlet characters modulo q. Then (over a reduced residue system):

$$\sum_{\substack{a \bmod q \\ \gcd(a,q)=1}} \chi(a)\overline{\psi(a)} = \begin{cases} \varphi(q), & \chi = \psi, \\ 0, & \chi \neq \psi. \end{cases}$$

A dual identity (summing over characters) is:

$$\sum_{\chi \bmod q} \chi(a)\overline{\chi(b)} = \begin{cases} \varphi(q), & a \equiv b \pmod q,\ \gcd(a,q) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

These are the algebraic "Fourier rules" that make Dirichlet's argument work.

Example: modulus $q = 4$

There are two characters modulo 4:

- χ_0 (principal).
- The nontrivial character χ_4:

$$\chi_4(n) = \begin{cases} 0, & 2 \mid n, \\ 1, & n \equiv 1 \pmod 4, \\ -1, & n \equiv 3 \pmod 4. \end{cases}$$

This single character already explains the classic race between primes 1 mod 4 and 3 mod 4.

7.6.2 What experiments usually measure

- How $\sum_{n \le N} \chi(n)$ behaves (cancellation, max partial sums).
- How orthogonality isolates a residue class.
- How many primitive characters exist for a given modulus, and how they "look" as tables.

7.6.3 Practical numerical caveats

- Always define $\chi(n) = 0$ when $\gcd(n, q) > 1$ (otherwise identities break).
- Represent χ as a lookup table on residues $0, \ldots, q-1$ and reduce via `n % q`.
- Be careful with complex floats: for most experiments, values are in $\{0, \pm 1\}$ or small roots of unity, so you can use exact complex numbers.

7.6.4 References

See ../references.

[Davenport, 2000, Ireland and Rosen, 1990, Lejeune Dirichlet, 1837, Serre, 1973]

7.6.5 Experiments in this repository

- **E106** — Character gallery: real vs complex characters; conjugate pairing.
- **E107** — Primitive vs imprimitive via conductor (induced characters).
- **E108** — Orthogonality relations as a heatmap; numeric sanity checks.

7.7 Dirichlet convolution refresher

Dirichlet convolution is a key "algebra" on arithmetic functions and shows up in many proofs and experiments. See [Apostol, 1976] and [Tenenbaum, 2015].

7.7.1 Definition

For arithmetic functions $f, g : \mathbb{N} \to \mathbb{C}$, their **Dirichlet convolution** is

$$(f * g)(n) = \sum_{d \mid n} f(d)\, g\left(\frac{n}{d}\right).$$

The function $1(n) \equiv 1$ acts like an identity in many formulas, and the **delta function** $\varepsilon(n)$ (with $\varepsilon(1) = 1$ and $\varepsilon(n) = 0$ for $n > 1$) is the convolution identity:

$$f * \varepsilon = f.$$

7.7.2 Möbius inversion

If

$$F(n) = \sum_{d \mid n} f(d) \quad \Longleftrightarrow \quad F = f * 1,$$

then **Möbius inversion** says

$$f = F * \mu.$$

This is one of the main reasons $\mu(n)$ appears everywhere.

7.7.3 Classic identities

- Sum of divisors:

$$\sigma(n) = (1 * \mathrm{id})(n),$$

  where $\mathrm{id}(n) = n$.

- Totient:

$$\varphi = \mu * \mathrm{id}.$$

- Jordan totient:

$$J_k = \mu * \mathrm{id}^k.$$

- von Mangoldt:

$$\Lambda = \mu * \log,$$

  in the sense of Dirichlet series / logarithmic derivatives of $\zeta(s)$.

7.7.4 Dirichlet series viewpoint (why it's computationally useful)

If $F(s) = \sum_{n\geq 1} f(n)n^{-s}$ and $G(s) = \sum_{n\geq 1} g(n)n^{-s}$ converge absolutely, then

$$\sum_{n\geq 1} \frac{(f * g)(n)}{n^s} = F(s)\,G(s).$$

This "turns convolution into multiplication" and underlies many analytic estimates; see [Montgomery and Vaughan, 2006].

7.7.5 Experiments in this repository

- **E102** — Dirichlet convolution identity zoo (*1=* , 1=id, 1*1=* , *id*1= , …).
- **E121** — Multiplicativity stress tests and convolution sanity checks (random coprime tests).

7.8 Dirichlet eta function η(s)

The **Dirichlet eta function** is the alternating Dirichlet series

$$\eta(s) = \sum_{n\geq 1} \frac{(-1)^{n-1}}{n^s},$$

which converges for (\Re(s)>0) (by alternating-series arguments). It is related to the Riemann zeta function by

$$\eta(s) = \left(1 - 2^{1-s}\right)\zeta(s).$$

7.8.1 Key ideas

- **Faster convergence:** for real (s>1), the alternating series often converges more rapidly than the plain -series.
- **Analytic continuation:** the identity above provides an analytic continuation of (s) away from the factor (1-2^{1-s}).
- **Numerical gateway:** (s) is a convenient "entry point" for simple (s) numerics without complex contour methods.

7.8.2 Why it matters in this project

When we want quick, small-scale experiments (e.g. comparing partial sums, smoothing, or convergence acceleration), (s) provides stable numerics with minimal infrastructure.

7.8.3 Experiments in this repository

- **E114** — (1/2+it) via (s): truncation/precision stability map.

7.8.4 References

See ../references.

[Edwards, 1974, Titchmarsh, 1986, Wikipedia contributors, 2025]

7.9 Dirichlet L-functions refresher

Dirichlet characters package congruence information; Dirichlet L-functions turn that into analytic objects whose zeros control prime distribution in residue classes.

7.9.1 Core definitions

Let χ be a Dirichlet character modulo q. The **Dirichlet L-function** is

$$L(s,\chi) = \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s}, \qquad \Re(s) > 1.$$

For $\Re(s) > 1$, it has an **Euler product**:

$$L(s,\chi) = \prod_{p \nmid q} \left(1 - \frac{\chi(p)}{p^s}\right)^{-1}.$$

For the principal character:

$$L(s,\chi_0) = \zeta(s) \prod_{p \mid q} \left(1 - p^{-s}\right).$$

The decisive analytic fact (used in Dirichlet's theorem) is:

$$L(1,\chi) \neq 0 \quad \text{for every nonprincipal character } \chi.$$

7.9.2 What experiments usually visualize or measure

- Convergence of partial sums $\sum_{n \le N} \chi(n) n^{-s}$ for various s.
- Convergence of partial Euler products over primes.
- Sensitivity near $s = 1$ (slow convergence) and how to stabilize it.

7.9.3 Practical numerical caveats

- Near $s = 1$, both series and Euler products converge painfully slowly.
- For Euler products, compute via logs:

  $$\log L(s,\chi) \approx -\sum_{p \le P} \log\left(1 - \chi(p) p^{-s}\right)$$

  to reduce catastrophic multiplication error.
- If you compare characters, always keep the same prime cutoff P to make plots comparable.

7.9.4 References

See ../references.

[Davenport, 2000, Lejeune Dirichlet, 1837, Serre, 1973, Washington, 1997]

7.9.5 Experiments in this repository

- **E110** — Dirichlet L-series partial sums at s=1 and s=1/2 (principal vs nonprincipal).
- **E111** — Euler product vs Dirichlet series truncations for L(s,): error vs cutoff.

7.10 Divisibility and modular arithmetic (Phase 2 core)

This page is the *toolbox layer* for elementary number theory experiments: **divisibility**, **gcd**, **congruences**, **residue classes**, and **modular inverses** appear in primality testing, factorization, and prime-distribution checks.

References for standard results: [Apostol, 1976, Niven *et al.*, 1991].

7.10.1 Divisibility

For integers a and b with $b \neq 0$, we say b **divides** a (written $b \mid a$) if there exists an integer k such that $a = bk$.

Basic closure rules used constantly:

- If $b \mid a$ and $b \mid c$, then $b \mid (a \pm c)$.
- If $b \mid a$, then $b \mid ac$ for any integer c.
- If $b \mid a$ and $a \mid b$, then $|a| = |b|$.

7.10.2 Greatest common divisor (gcd)

The **greatest common divisor** of a and b, written $\gcd(a, b)$, is the largest positive integer d such that $d \mid a$ and $d \mid b$.

We call a and b **coprime** if $\gcd(a, b) = 1$.

Bézout identity (extended gcd)

There exist integers x, y such that

$$\gcd(a, b) = ax + by.$$

This identity is what enables **modular inverses** and appears directly in many algorithms (e.g., computing inverses for modular division).

Euclidean algorithm

If $a = bq + r$ with $0 \leq r < |b|$, then

$$\gcd(a, b) = \gcd(b, r).$$

Repeatedly applying this step gives the Euclidean algorithm, which runs in time polynomial in the number of digits of the input.

7.10.3 Congruences and residue classes

For integers a, b and modulus $m \geq 1$,

$$a \equiv b \pmod{m} \quad \Longleftrightarrow \quad m \mid (a - b).$$

This is an equivalence relation; its equivalence classes are called the **residue classes modulo** m.

In computations, we usually represent residues by integers in $\{0, 1, \ldots, m-1\}$. That is a *representation choice*; the underlying object is the residue class.

Arithmetic modulo m

If $a \equiv a' \pmod{m}$ and $b \equiv b' \pmod{m}$, then:

- $a + b \equiv a' + b' \pmod{m}$
- $ab \equiv a'b' \pmod{m}$

So addition and multiplication "mod m" are well-defined.

7.10.4 Units and modular inverses

An integer a has a multiplicative inverse modulo m if and only if

$$\gcd(a, m) = 1.$$

In that case, there exists a^{-1} such that

$$aa^{-1} \equiv 1 \pmod{m}.$$

In code, compute a^{-1} via the extended Euclidean algorithm (Bézout coefficients).

Important practical note: if $\gcd(a, m) \neq 1$, then "division by a modulo m" is not defined.

7.10.5 Euler's totient function and Euler's theorem

Let $\varphi(m)$ be the number of integers in $\{1, \ldots, m\}$ that are coprime to m. Then **Euler's theorem** states:

$$\gcd(a, m) = 1 \;\Rightarrow\; a^{\varphi(m)} \equiv 1 \pmod{m}.$$

This generalizes Fermat's little theorem (below) and is a conceptual bridge between congruences and multiplicative structure.

7.10.6 Fermat's little theorem (FLT)

If p is prime and $p \nmid a$, then

$$a^{p-1} \equiv 1 \pmod{p}.$$

FLT motivates Fermat-style primality tests, but there are composite numbers that can still pass such tests (pseudoprimes, especially Carmichael numbers). See *Carmichael numbers*.

7.10.7 Modular exponentiation (what you implement)

Nearly every algorithm in this phase needs fast computation of $a^e \bmod m$.

Use **binary exponentiation** ("square-and-multiply"), which evaluates a^e in $O(\log e)$ modular multiplications.

Practical checklist for implementations:

- Reduce a modulo m once at the start.
- Use repeated squaring; never compute a^e as a huge integer first.
- Use constant-time or side-channel-resistant variants only if you have a security goal (this project typically does not).

7.10.8 Chinese remainder theorem (CRT)

If m and n are coprime, then for any residues $a \pmod m$ and $b \pmod n$, there exists a unique residue $x \pmod{mn}$ such that

$$x \equiv a \pmod m, \qquad x \equiv b \pmod n.$$

Equivalently, the ring $\mathbb{Z}/(mn)\mathbb{Z}$ "splits" into $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ when $\gcd(m, n) = 1$.

CRT is the conceptual reason that many modular phenomena can be studied prime-power by prime-power. It is also a common proof tool for counterexamples and constructions.

7.10.9 How this connects to Phase 2 experiments

- **Primality tests**: Fermat and Miller–Rabin use modular exponentiation and gcd checks.
- **Factorization**: Pollard–ρ uses repeated modular iteration and $\gcd(|x - y|, n)$.
- **Residue classes**: counting primes in residue classes is literally "arithmetic modulo m".

Suggested related pages:

- *Prime numbers refresher*
- *Primality testing: guarantees, error bounds, and what to report*
- *Semiprimes*
- *Carmichael numbers*
- *Prime counting approximations: (x), Li(x), and R(x)*

7.11 Divisor functions $d(n)$ and $\sigma_k(n)$ refresher

Divisor functions measure how many divisors a number has, and how large they are. They are classical examples of multiplicative functions. See [Apostol, 1976].

7.11.1 Counting divisors: $d(n)$

Let $d(n)$ (also written $\tau(n)$) be the number of positive divisors of n.

If $n = \prod p_i^{\alpha_i}$, then

$$d(n) = \prod(\alpha_i + 1).$$

7.11.2 Sum of divisors: $\sigma_k(n)$

For $k \geq 0$,

$$\sigma_k(n) = \sum_{d|n} d^k.$$

The case $k = 1$ is the usual sum-of-divisors function $\sigma(n)$.

For a prime power,

$$\sigma_k(p^\alpha) = 1 + p^k + p^{2k} + \cdots + p^{\alpha k} = \frac{p^{(\alpha+1)k} - 1}{p^k - 1}.$$

Again, multiplicativity gives $\sigma_k(n)$ from prime powers.

7.11.3 Highly composite numbers (extremal behavior)

Numbers that maximize $d(n)$ up to a range are called **highly composite numbers**. Ramanujan's classic paper studies their structure. [Ramanujan, 1915]

7.11.4 Experiment ideas

- visualize $d(n)$ up to N and highlight record-breakers
- compare $\sigma(n)$ to n (abundant / perfect / deficient classification)
- log-scale plots of $d(n)$ to make extremes visible

7.11.5 Experiments in this repository

- **E096** — Record-holders for (n) up to N (highly composite flavor).
- **E097** — (n)/n landscape: deficient / perfect / abundant classification.
- **E098** — Extremals of (n)/n^ across (phase changes / superabundant intuition).

7.11.6 References

See ../references.

[Alaoglu and Erdős, 1944, Apostol, 1976, Lagarias, 2002, Ramanujan, 1915, Robin, 1984]

7.12 Euler's prime-generating polynomial refresher

One of the most famous "too good to be true" examples in experimental number theory is Euler's quadratic

$$f(n) = n^2 + n + 41.$$

For integers $n = 0, 1, \ldots, 39$ it produces **prime numbers**. This is an unusually long initial run for such a simple polynomial, and it remains a classic motivation for computational "prime-value" experiments. [contributors, 2025, Weisstein, 2025]

7.12.1 The basic facts

- For $0 \le n \le 39$, $f(n)$ is prime.
- At $n = 40$,

$$f(40) = 40^2 + 40 + 41 = 1681 = 41^2,$$

  so the prime streak stops immediately afterward. [Weisstein, 2025]

(You will also see the equivalent form $n^2 - n + 41$ in the literature; it is just a shift: $n^2 - n + 41 = f(n-1)$.)

7.12.2 What "prime-producing polynomial" can mean

The phrase *prime-producing* is used in three different (and easily confused) ways:

1. **Prime for every integer input.**
   Apart from constant polynomials (e.g. $f(n) = 2$), this is impossible for integer polynomials. [Hardy and Wright, 2008]
2. **Prime for a long initial run.**
   This is what Euler's polynomial does: it produces primes for many *consecutive* values, but not forever. [contributors, 2025]

3. **Prime infinitely often.**
   Here the goal is not "always prime", but "infinitely many n with $f(n)$ prime". For degree 1, Dirichlet's theorem proves this (arithmetic progressions). For degree ≥ 2 it is open in general, and conjectures (Bunyakovsky, Bateman–Horn) give a framework and quantitative predictions. [Bateman and Horn, 1962, contributors, 2025, Lejeune Dirichlet, 1837]

In experimental projects it helps to state explicitly which of the three you mean, because they lead to very different "success criteria" and plots.

7.12.3 Why a nonconstant integer polynomial cannot be prime for all integers

Let $f \in \mathbb{Z}[x]$ be nonconstant. Pick any integer m with $|f(m)| > 1$ (such an m exists because a nonconstant polynomial is unbounded). Consider the values

$$f\big(m + k\, f(m)\big), \qquad k = 1, 2, 3, \ldots$$

Because $m + kf(m) \equiv m \pmod{f(m)}$ and f has integer coefficients, we get the congruence

$$f\big(m + k\, f(m)\big) \equiv f(m) \pmod{f(m)}.$$

So $f(m)$ divides $f(m + kf(m))$. For all sufficiently large k the absolute value $\big|f(m + kf(m))\big|$ is strictly larger than $|f(m)|$, hence $f(m + kf(m))$ has a nontrivial divisor and is composite. This shows:

> No nonconstant polynomial with integer coefficients can take prime values for all integers.

This simple divisibility trick is one of the reasons why Euler's streak is remarkable but still compatible with the "no prime for all inputs" principle. [Hardy and Wright, 2008]

7.12.4 Why Euler's quadratic has such a long streak

Two complementary explanations are useful:

1) A built-in factor at $n = 40$

The identity

$$f(n) \equiv 0 \pmod{41} \quad \text{when } n \equiv 40 \pmod{41}$$

is not an accident: the first failure happens exactly at $n = 40$ where $f(40) = 41^2$.

More generally, for any prime q, the set of residues $n \bmod q$ for which $f(n) \equiv 0 \pmod{q}$ forms a "local obstruction": every such residue forces a composite value (unless the value equals q itself). Counting and visualizing these local obstructions is often the fastest way to understand why some polynomials are "better" prime generators than others.

2) An algebraic number theory viewpoint (Heegner number 163)

Euler's polynomial is naturally a **norm form** in the quadratic field $\mathbb{Q}(\sqrt{-163})$. Let

$$\omega = \frac{1 + \sqrt{-163}}{2}.$$

Then the (field) norm satisfies

$$N(n + \omega) = (n + \omega)(n + \overline{\omega}) = n^2 + n + 41 = f(n).$$

The discriminant here is -163, and 163 is a **Heegner number**: the ring of integers of $\mathbb{Q}(\sqrt{-163})$ has class number 1, i.e. it enjoys unique factorization. This special property is one reason the prime streak for 41 is unusually long compared to "random" quadratics. [contributors, 2025, Cox, 2013, Weisstein, 2025]

A classical theorem of Rabinowitsch makes the link precise: for a prime p, the polynomial $n^2 + n + p$ produces primes for $n = 0, \ldots, p-2$ **if and only if** the discriminant $1 - 4p$ is the negative of a Heegner number. In particular, $p = 41$ corresponds to $1 - 4p = -163$. [contributors, 2025]

This is also why the so-called *lucky numbers of Euler* are exactly $2, 3, 5, 11, 17, 41$. [contributors, 2025, Weisstein, 2025]

7.12.5 Prime values infinitely often: what we believe (and what is known)

Euler's polynomial fails to be prime eventually, but it is widely believed to take **prime values infinitely often**. Proving this is currently out of reach for essentially any polynomial of degree ≥ 2.

A standard "minimal sanity check" for a polynomial $f \in \mathbb{Z}[x]$ to have infinitely many prime values is:

- **(irreducible)** f should be irreducible over $\mathbb{Q}$,
- **(no fixed prime divisor)** the gcd of the integer values $f(0), f(1), f(2), \ldots$ should be 1.

Bunyakovsky's conjecture asserts that these necessary conditions are also sufficient. [contributors, 2025]

Bateman–Horn goes further and predicts an asymptotic density: roughly, the count of primes among $f(1), \ldots, f(N)$ should be

$$\sim C(f) \int_2^N \frac{dt}{\log f(t)},$$

where $C(f)$ is an explicit constant built from local obstruction data modulo each prime. [Bateman and Horn, 1962]

For experiments, this is extremely useful: it gives you a quantitative "expected curve" to plot against your observed prime counts.

7.12.6 What to experiment with

A few experiment ideas that scale from quick checks to deeper projects:

- **Streak length across Euler's form:** for primes p, measure the length of the initial prime run of $n^2 + n + p$ and compare $p \in \{2, 3, 5, 11, 17, 41\}$ to nearby primes. [contributors, 2025, contributors, 2025]
- **Local obstruction profile:** for each small prime q, count the number of solutions to $f(n) \equiv 0 \pmod{q}$ and visualize the "obstruction density" across q.
- **Bateman–Horn calibration:** empirically estimate $C(f)$ from a truncated Euler product over small primes and compare Bateman–Horn's predicted prime count against the observed count.
- **Baselines:** compare Euler's polynomial against random quadratics $n^2 + an + b$ with the same size of coefficients, filtered by "no fixed prime divisor", so you can see what is truly special and what is selection bias.

7.12.7 Practical numerical caveats

- **Big integers:** $f(n)$ grows like n^2; primality testing dominates runtime quickly.
- **Avoid accidental float conversion:** keep n and $f(n)$ as Python integers throughout.
- **Sampling bias:** focusing only on "nice-looking" polynomials can be misleading; compare against random baselines rather than only showcasing "winners".

7.12.8 References

See ../references.

[Bateman and Horn, 1962, contributors, 2025, contributors, 2025, contributors, 2025, Cox, 2013, Hardy and Wright, 2008, Lejeune Dirichlet, 1837, Weisstein, 2025, Weisstein, 2025, Weisstein, 2025]

7.13 Euler's totient function $\varphi(n)$ refresher

Euler's totient function $\varphi(n)$ counts reduced residues modulo n. Standard references: [Apostol, 1976], [Niven *et al.*, 1991].

7.13.1 Definition

For $n \geq 1$,

$$\varphi(n) = \#\{\, 1 \leq k \leq n : \gcd(k, n) = 1 \,\}.$$

Equivalently: $\varphi(n)$ is the size of the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^{\times}$.

7.13.2 Prime power formula

If p is prime and $k \geq 1$,

$$\varphi(p^k) = p^k - p^{k-1} = p^k \left(1 - \frac{1}{p}\right).$$

Since φ is multiplicative, for $n = \prod p_i^{\alpha_i}$:

$$\varphi(n) = n \prod_{p \mid n} \left(1 - \frac{1}{p}\right).$$

7.13.3 Important properties

- **Euler's theorem:** if $\gcd(a, n) = 1$ then $a^{\varphi(n)} \equiv 1 \pmod{n}$.
- **Average order:** roughly $\sum_{n \leq x} \varphi(n) \sim \frac{3}{\pi^2} x^2$ (analytic methods).
- **Extremes:** $\varphi(n)$ is small when n has many small prime factors.

7.13.4 Computational notes

Given the prime factorization of n, $\varphi(n)$ is cheap to compute via the product formula. For experiments, you can often compute it for all $n \leq N$ efficiently using a sieve-style method.

7.13.5 Experiments in this repository

- **E101** — Reduced residues (ℤ/qℤ)× as an explicit set; verify size (q).

7.14 Explicit formulas: primes ↔ zeros

"Explicit formulas" are identities that connect prime counting (typically via Chebyshev functions like (\psi(x))) to sums over zeros of (s) (or more general L-functions). Conceptually, they are a precise form of the slogan:

> **Primes are controlled by zeros.**

A prototypical form expresses (\psi(x)) as a main term (x) plus oscillatory corrections coming from nontrivial zeros.

7.14.1 Key ideas

- **From Euler product to primes:** differentiating (\log\zeta(s)) connects (s) to the von Mangoldt function (\Lambda(n)).
- **Zeros drive oscillations:** sums over (\rho) (zeros) produce fluctuating terms that explain biases and sign changes seen in finite ranges.
- **Numerical truncation:** in experiments, one typically truncates zero sums and studies stability vs cutoff parameters.

7.14.2 Why it matters in this project

This is the bridge between the "Dirichlet character / L-function" block and the "prime race / bias" phenomena: you can literally watch zero contributions modulate prime-counting curves.

7.14.3 Experiments in this repository

- **E119** — (x) − x oscillations and "explicit-formula intuition" (visual).

7.14.4 References

See ../references.

[Ivić, 1985, Iwaniec and Kowalski, 2004, Schoenfeld, 1976, Titchmarsh, 1986, Wikipedia contributors, 2025]

7.15 Exploratory visualizations for arithmetic functions

Several experiments in this project are intentionally "EDA-like": they produce **heatmaps**, **atlases**, and **correlation matrices** for many functions at once. This page collects best practices so those figures remain interpretable and comparable.

7.15.1 Core definitions

Given a function table $F(n)$ for $n = 1, \ldots, N$ and a list of arithmetic functions $f_1, \ldots, f_m$, two common derived views are:

- **Heatmap atlas:** visualize values $f_j(n)$ as an image (rows = functions, columns = n or blocks of n).
- **Correlation matrix:** compute an empirical correlation between columns $f_i(n)$ and $f_j(n)$ (Pearson or rank-based).

7.15.2 What experiments usually visualize or measure

- "Texture": squarefree bands, prime powers, smooth numbers, record-holders, etc.
- Clustering: which functions behave similarly on $1..N$ (after normalization).
- Stability: how patterns change when N grows or when you sample on a log grid.

7.15.3 Practical numerical caveats

- **Normalize first.** Raw scales differ wildly (e.g. $\sigma(n)$ vs $\mu(n)$). Typical choices: z-score, log1p, or scaling by n^{α}.
- **Beware heavy tails.** Extremal values can dominate Pearson correlation; consider Spearman correlation or winsorization for robustness.
- **Sampling matters.** Linear n emphasizes small structure; log-spaced n emphasizes asymptotic behavior. Record the sampling rule in the figure caption.

7.15.4 References

See ../references.

[Arnold, 2015, Borwein and Bailey, 2008]

7.15.5 Experiments in this repository

- **E122** — Heatmap atlas of , , Ω and related "squarefree texture" features.
- **E123** — Correlation matrix of arithmetic functions (with normalization variants).

7.16 Factorization pipelines (trial division + Pollard rho)

Phase 2 includes experiments that do not only test primality but also attempt to **factor** integers. In practice you rarely use a single algorithm; you build a pipeline:

1. Cheap deterministic filters (small primes, gcd checks).
2. A fast primality test for the remaining cofactor (often Miller-Rabin).
3. A randomized factor finder for hard cases (Pollard rho), with retries.

This page describes a practical, experiment-friendly pipeline and how to report it.

7.16.1 What is being computed

Given an integer $n \geq 2$, factorization aims to write

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

with distinct primes p_i and exponents $e_i \geq 1$.

7.16.2 Deterministic front-end: trial division

Trial division tries to find small prime factors by dividing by primes $p \leq B$.

- It is **deterministic**.
- It is extremely effective as a first stage.
- It gives you natural runtime knobs (`B`, number of primes).

Implementation notes:

- Precompute primes up to B with a sieve.
- Divide repeatedly to capture multiplicities.
- After removing small primes, the remaining cofactor can be 1, prime, or composite.

7.16.3 Probabilistic core: Pollard rho

Pollard's rho is a randomized method that often finds a nontrivial factor quickly. It is the standard next step after trial division for medium-size integers. See [Pollard, 1975] and the practical improvements in [Brent, 1980].

Idea (high level)

Work modulo n and iterate a polynomial map, commonly

$$f(x) = x^2 + c \pmod{n}.$$

The sequence $x_{i+1} = f(x_i)$ eventually cycles modulo a hidden prime divisor p of n. Using gcd computations on differences, you can often extract a factor:

$$d = \gcd(|x_i - x_j|, n).$$

If $1 < d < n$ you found a nontrivial factor.

Why it is probabilistic

- The runtime depends on the chosen function parameter c, the start value, and luck.
- For some choices, the method can stall; you must retry with different parameters.

Because of this, Pollard rho is **probabilistic / heuristic**. You must report it that way.

7.16.4 A practical pipeline

A good experiment pipeline is:

1. **Handle trivial cases:** $n < 2$, even numbers, perfect powers if you want.
2. **Trial division:** divide by primes up to B.
3. **Primality check on the cofactor:** if the remaining cofactor is 1 or prime, stop.
4. **Pollard rho for the remaining composite cofactor:** find a factor d.
5. **Recurse:** factor d and n/d using the same pipeline.
6. **Normalize:** sort factors, combine multiplicities.
7. **Verify:** multiply the result back and check you recover the original n.

The key property for correctness is the final verification step: it turns implementation bugs into test failures.

7.16.5 Reporting checklist

In your report, always include:

- **Deterministic vs probabilistic:**
  - Trial division: deterministic.
  - Pollard rho: probabilistic / heuristic.
- **Runtime knobs:**
  - trial division bound B,
  - max rho iterations per attempt,
  - number of rho restarts,
  - random seed (if used).
- **Correctness cross-check:**
  - for small n compare against a trusted reference (full trial division or a library factorization),
  - verify the product of returned factors equals the original input.

7.16.6 Common pitfalls

- **Repeated factors:** always divide repeatedly (e.g. $n = 12$ has factor 2 twice).
- **Prime cofactors:** do not feed a prime into rho without checking; you can loop unnecessarily.
- **gcd edge cases:** rho can return $d = n$ (failure); treat as "retry".
- **Non-reproducible results:** randomized algorithms should record a seed or a full parameter list.

7.16.7 How this connects to experiments

- Semiprimes and factorization difficulty: *Semiprimes*
- Primality tests used inside pipelines: *Primality testing: guarantees, error bounds, and what to report*
- Modular arithmetic building blocks: *Divisibility and modular arithmetic (Phase 2 core)*

7.16.8 References

See ../references.

7.17 Fermat numbers

Fermat numbers are the integers

$$F_n = 2^{2^n} + 1 \quad (n \geq 0).$$

They grow extremely quickly and are central to a classic "counterexample" story: Fermat conjectured that all F_n are prime, but F_5 is composite (Euler). [Kř\'ıžek *et al.*, 2013, Wikipedia contributors, 2025]

7.17.1 Key facts

- **Coprime property:** $\gcd(F_m, F_n) = 1$ for $m \neq n$. In fact, $F_0 F_1 \cdots F_{n-1} = F_n - 2$. [Kř\'ıžek *et al.*, 2013]
- **Fermat primes:** If F_n is prime, it is called a Fermat prime (known: $n = 0..4$). [The OEIS Foundation Inc., 2025]
- **Pépin test (primality):** For $n \geq 1$, F_n is prime iff $3^{(F_n-1)/2} \equiv -1 \pmod{F_n}$. [Kř\'ıžek *et al.*, 2013]

7.17.2 What to experiment with

- **Factor hunting:** Search for small prime factors of F_n (using modular arithmetic and wheel/primorial filters).
- **Pépin vs. trial division:** Compare performance and failure modes as n increases.
- **Structure of known factors:** Many known prime factors have the form $k \cdot 2^{n+2} + 1$; explore how often such candidates appear.
- **Euclid-style numbers:** Explore $E_n = 1 + \prod_{i=0}^{n} p_i$ and compare "Euclid primes" vs. Fermat primes (they behave very differently).

7.17.3 References

See Kř\'ıžek *et al.* [2013], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

7.18 Gauss sums refresher

Gauss sums are finite Fourier transforms of Dirichlet characters. In Phase 2 they serve two purposes:

1. **A correctness/structure check for characters:** for primitive characters, the Gauss sum magnitude obeys a clean law.
2. **A bridge to analytic number theory:** Gauss sums appear in functional equations for Dirichlet L-functions and in explicit evaluations of some special values.

This page gives the definitions and the specific facts that the experiments use, with enough context to interpret the plots.

7.18.1 Additive characters and the Fourier viewpoint

Fix a modulus $q \geq 1$. The basic additive character modulo q is

$$e_q(n) := \exp\left(\frac{2\pi i n}{q}\right).$$

It is periodic mod q and satisfies $e_q(m+n) = e_q(m)e_q(n)$. You can view $e_q(an)$ as a Fourier mode on the finite group $\mathbb{Z}/q\mathbb{Z}$.

A Dirichlet character χ modulo q is (after restricting to units) a *multiplicative* character on $(\mathbb{Z}/q\mathbb{Z})^\times$. Gauss sums mix the multiplicative character χ with the additive phase $e_q(\cdot)$, i.e. they are "multiplicative objects seen through additive Fourier glasses".

7.18.2 Core definitions

Let χ be a Dirichlet character modulo q. The (basic) **Gauss sum** is

$$\tau(\chi) := \sum_{a=0}^{q-1} \chi(a)\, e_q(a).$$

A more general "twisted" Gauss sum is

$$\tau(\chi, b) := \sum_{a=0}^{q-1} \chi(a)\, e_q(ba) \qquad (b \in \mathbb{Z}).$$

Immediate properties

- If $b \equiv 0 \pmod{q}$, then $\tau(\chi, b) = \sum_a \chi(a)$.
- If $\gcd(b, q) = 1$, then a change of variables implies the relation

$$\tau(\chi, b) = \overline{\chi}(b)\, \tau(\chi).$$

This identity is the main reason experiments can focus on $\tau(\chi)$ without losing information: all invertible twists are just rotations/scalings by a root of unity.

7.18.3 The magnitude law $|\tau(\chi)| = \sqrt{q}$ for primitive characters

The central numerical pattern is:

> If χ is **primitive** modulo q, then $|\tau(\chi)| = \sqrt{q}$.

A good way to remember *why* this is plausible is to compute the squared magnitude and watch orthogonality collapse the double sum. Start with

$$|\tau(\chi)|^2 = \tau(\chi)\,\overline{\tau(\chi)} = \sum_{a=0}^{q-1}\sum_{b=0}^{q-1} \chi(a)\overline{\chi}(b)\, e_q(a-b).$$

For primitive characters, the sum reorganizes into complete additive character sums with perfect cancellation, giving exactly q. For **imprimitive** characters (induced from a smaller conductor), the same cancellation can fail; magnitudes may be smaller (and in some cases can vanish). This is why Phase 2 experiments should either restrict to primitive characters for "$\sqrt{q}$ laws" or explicitly label imprimitive cases.

7.18.4 Quadratic Gauss sums (cleanest special case)

For an odd prime p and the quadratic character (Legendre symbol) $\chi(n) = \left(\frac{n}{p}\right)$, the Gauss sum has a famous closed form:

$$\tau(\chi) = \varepsilon_p \sqrt{p}, \qquad \varepsilon_p = \begin{cases} 1, & p \equiv 1 \pmod 4, \\ i, & p \equiv 3 \pmod 4. \end{cases}$$

So not only is the magnitude $\sqrt{p}$, but the argument depends on p mod 4. This is a strong "sanity target" for Phase 2: if your implementation of χ is correct and you use a consistent residue convention, the plotted points should land on the expected circle with the expected symmetry.

7.18.5 Why Gauss sums matter for Dirichlet L-functions (high level)

For a primitive character χ, the completed L-function satisfies a functional equation of the form

$$\Lambda(s, \chi) = W(\chi)\, \Lambda(1 - s, \overline{\chi}),$$

where the **root number** $W(\chi)$ has absolute value 1 and is built from $\tau(\chi)$ (up to normalization by $\sqrt{q}$ and a parity factor).

You do not need the full functional equation machinery for Phase 2 plots, but it explains why Gauss sums appear naturally whenever you connect characters, L-functions, and oscillations in arithmetic progression counts.

7.18.6 Practical numerical notes for experiments

- **Residue conventions:** be consistent about whether you sum $a = 0, \ldots, q-1$ or $a = 1, \ldots, q$; both are valid but should not be mixed.
- **Non-units:** if you extend χ by 0 on $\gcd(a, q) > 1$, those terms contribute 0, but you must ensure your implementation really does this.
- **Floating-point cancellation:** Gauss sums are dominated by cancellation. Use complex128 and avoid rounding before taking magnitudes.

7.18.7 References

See ../references.

[Berndt *et al.*, 1998, Davenport, 2000, Iwaniec and Kowalski, 2004]

7.18.8 Experiments in this repository

- **E067** — Gauss sums: magnitude vs. $\sqrt{q}$ (Phase 2 core).
- **E109** — Gauss sums $\tau(\chi)$: magnitude law and geometry for characters modulo q.

7.19 Gram points and zero counting

A **Gram point** is (informally) a value (t) where the Riemann–Siegel theta function (\theta(t)) hits an integer multiple of (\pi). These points are useful landmarks when visualizing (Z(t)) and tracking sign changes.

The **Riemann–von Mangoldt formula** gives the asymptotic count of nontrivial zeros up to height (T):

$$N(T) = \frac{T}{2\pi} \log\left(\frac{T}{2\pi}\right) - \frac{T}{2\pi} + O(\log T) \quad (T \to \infty),$$

with refinements that include the argument of on the critical line.

7.19.1 Key ideas

- **Bookkeeping:** zero counting provides a "sanity check" for numerical root-finding: we can compare a computed zero list against (N(T)).
- **Gram blocks / Gram's law:** empirical rules about how zeros sit between consecutive Gram points (useful, but not always true).
- **Practical numerics:** many experiments become clearer when plotted against (\theta(t)) or indexed by Gram points.

7.19.2 Experiments in this repository

- **E116** — Observed zero counts (via sign changes) vs Riemann–von Mangoldt estimate.

7.19.3 References

See ../references.

[Odlyzko, 1992, Titchmarsh, 1986, Wikipedia contributors, 2025, Wikipedia contributors, 2025]

7.20 Hardy's Z-function and the critical line

For real (t), the **Hardy Z-function** is defined (up to standard conventions) by

$$Z(t) = e^{i\theta(t)}\,\zeta\big(\tfrac{1}{2} + it\big)\,,$$

where (\theta(t)) is the Riemann–Siegel theta function. The key feature is that **Z(t) is real-valued for real t**, so zeros of (Z(t)) correspond to zeros of (s) on the critical line.

7.20.1 Key ideas

- **Real signal:** studying sign changes of (Z(t)) is a practical way to bracket zeros on (\Re(s)=\tfrac12).
- **Theta function:** (\theta(t)) captures the "oscillatory phase" of on the critical line and is closely tied to Gram points.
- **Numerical workflows:** many computational approaches to zeta zeros are formulated in terms of (Z(t)), (\theta(t)), and their approximations.

7.20.2 Why it matters in this project

It gives a clean, visualization-friendly path from "complex -values" to "real curves" whose roots can be bracketed and counted.

7.20.3 Experiments in this repository

- **E115** — Hardy Z sign-change scan and zero bracketing (bisection refinement).

7.20.4 References

See ../references.

[Odlyzko, 1992, Titchmarsh, 1986, Wikipedia contributors, 2025, Wikipedia contributors, 2025]

7.21 Heegner numbers

7.21.1 Overview

The **Heegner numbers** are the nine positive integers

$$d \in \{1, 2, 3, 7, 11, 19, 43, 67, 163\}$$

for which the imaginary quadratic field $K = \mathbb{Q}(\sqrt{-d})$ has **class number** $h_K = 1$. Equivalently, the ring of integers $\mathcal{O}_K$ is a **unique factorization domain** (UFD). [contributors, 2025, Weisstein, 2025]

This "exactly nine" phenomenon is one of the landmark classification results in classical algebraic number theory and is tightly connected to:

- the arithmetic of **binary quadratic forms** (Gauss),
- **complex multiplication** of elliptic curves and **singular moduli** (Heegner–Stark),
- and the famous "almost integer" $e^{\pi\sqrt{163}}$.

[Cox, 2013]

7.21.2 1. Imaginary quadratic fields and discriminants

Let $d > 0$ be squarefree and set $K = \mathbb{Q}(\sqrt{-d})$.

1.1 The ring of integers $\mathcal{O}_K$

A frequent pitfall is assuming $\mathcal{O}_K = \mathbb{Z}[\sqrt{-d}]$ always holds. In fact:

- If $d \equiv 1, 2 \pmod 4$, then

$$\mathcal{O}_K = \mathbb{Z}[\sqrt{-d}], \qquad \Delta_K = -4d.$$

- If $d \equiv 3 \pmod 4$, then

$$\mathcal{O}_K = \mathbb{Z}\left[\frac{1+\sqrt{-d}}{2}\right], \qquad \Delta_K = -d.$$

Here Δ_K is the (fundamental) **field discriminant**. [Cox, 2013]

1.2 Heegner discriminants

Because the theory is often indexed by discriminant rather than by d, it is useful to record:

$$\Delta \in \{-4, -8, -3, -7, -11, -19, -43, -67, -163\}$$

These are exactly the **negative fundamental discriminants** with class number 1. [Cox, 2013]

7.21.3 2. Class number and unique factorization

2.1 From element factorization to ideal factorization

The ring $\mathcal{O}_K$ is a **Dedekind domain**. This implies:

- Every nonzero ideal factors uniquely into prime ideals.
- But elements in $\mathcal{O}_K$ may fail to factor uniquely into irreducibles.

The mechanism measuring "how far" we are from principal ideals is the **ideal class group**:

$$\mathrm{Cl}(\mathcal{O}_K) = \frac{\{\text{nonzero fractional ideals of } \mathcal{O}_K\}}{\{\text{principal fractional ideals}\}}.$$

Its cardinality is the **class number**:

$$h_K := \#\mathrm{Cl}(\mathcal{O}_K).$$

[Cox, 2013]

2.2 Why $h_K = 1$ is the same as "UFD" here

For rings of integers in number fields (Dedekind domains), we have the chain of equivalences:

$$h_K = 1 \iff \text{every ideal is principal (PID)} \iff \mathcal{O}_K \text{ is a UFD.}$$

So Heegner numbers classify precisely the imaginary quadratic integer rings with unique factorization. [Cox, 2013]

2.3 A concrete failure of unique factorization

In $\mathbb{Z}[\sqrt{-5}]$ (which corresponds to $d = 5$, **not** a Heegner number), we have:

$$6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5}),$$

and these are genuinely distinct factorizations (up to units), so UFD fails; indeed $h_{\mathbb{Q}(\sqrt{-5})} > 1$. [Cox, 2013]

7.21.4 3. Gauss' binary quadratic forms viewpoint

Binary quadratic forms provide a very concrete model for class groups.

A primitive positive definite binary quadratic form is

$$Q(x, y) = ax^2 + bxy + cy^2, \qquad a, b, c \in \mathbb{Z},$$

with negative discriminant

$$\Delta = b^2 - 4ac < 0.$$

Two forms are equivalent if related by an $\mathrm{SL}_2(\mathbb{Z})$ change of variables. Gauss' composition defines a finite abelian group on equivalence classes, and that group matches the ideal class group (for the order of discriminant Δ; in particular for fundamental Δ it matches $\mathrm{Cl}(\mathcal{O}_K)$). [Cox, 2013]

3.1 Reduced forms and the class number

A positive definite form (a, b, c) is **reduced** if:

$$|b| \le a \le c,$$

and if $|b| = a$ or $a = c$, then additionally $b \ge 0$.

A key fact: the number of reduced forms of discriminant Δ equals the class number $h(\Delta)$. So $h(\Delta) = 1$ means: **there is exactly one reduced form** for that discriminant. [Cox, 2013]

3.2 The unique reduced forms for the Heegner discriminants

For each Heegner discriminant, the single reduced form can be written explicitly:

- $\Delta = -4$: $x^2 + y^2$
- $\Delta = -8$: $x^2 + 2y^2$
- $\Delta = -3$: $x^2 + xy + y^2$
- $\Delta = -7$: $x^2 + xy + 2y^2$
- $\Delta = -11$: $x^2 + xy + 3y^2$
- $\Delta = -19$: $x^2 + xy + 5y^2$
- $\Delta = -43$: $x^2 + xy + 11y^2$
- $\Delta = -67$: $x^2 + xy + 17y^2$
- $\Delta = -163$: $x^2 + xy + 41y^2$

You can verify, for example, that $x^2 + xy + 41y^2$ has discriminant $1 - 4 \cdot 1 \cdot 41 = -163$. This "single reduced form" phenomenon is another face of class number one. [Cox, 2013]

7.21.5 4. The class-number-one theorem (why exactly nine?)

The theorem states that there are **exactly nine** imaginary quadratic fields of class number one. Historically:

- **Heegner (1952)** proved the result using modular functions and the emerging ideas of **complex multiplication**. [Heegner, 1952]
- A gap in the original proof was later clarified and the argument completed/strengthened in later work (notably by **Stark**), leading to the modern accepted classification. [Stark, 1967, Stark, 1969]
- The broader Diophantine toolkit around these questions is closely linked to effective bounds from linear forms in logarithms (Baker's theory). [Baker, 1966]

A widely used modern reference that connects the class group, quadratic forms, and complex multiplication in one narrative is Cox. [Cox, 2013]

7.21.6 5. Why Heegner numbers create "almost integers"

The most famous numerical surprise is:

$$e^{\pi\sqrt{163}} \approx 262537412640768743.999999999999\ldots$$

This is not an accident; it comes from **complex multiplication** and the q-expansion of the modular j-invariant.

5.1 The j-invariant and its q-series

Let $j(\tau)$ be the modular j-invariant. It has a Fourier expansion in

$$q = e^{2\pi i\tau}$$

of the form

$$j(\tau) = q^{-1} + 744 + 196884\,q + \cdots,$$

with integer coefficients. [Diamond and Shurman, 2005]

When τ is an **imaginary quadratic** point in the upper half-plane (a CM point), $j(\tau)$ is an **algebraic integer**. More precisely, $j(\tau)$ generates the **Hilbert class field** of $K = \mathbb{Q}(\tau)$. When $h_K = 1$, the Hilbert class field is just K itself, and the special values become exceptionally explicit. [Cox, 2013]

5.2 Why $d = 163$ is the showpiece

Take

$$\tau = \frac{1 + \sqrt{-163}}{2}.$$

Then $\mathrm{Im}(\tau) = \frac{\sqrt{163}}{2}$, so

$$|q| = e^{-2\pi \,\mathrm{Im}(\tau)} = e^{-\pi\sqrt{163}},$$

which is astronomically small. Also $\mathrm{Re}(\tau) = \frac{1}{2}$, so q is *negative real* to extremely high precision:

$$q = e^{2\pi i(\frac{1}{2} + i\frac{\sqrt{163}}{2})} = e^{\pi i}\, e^{-\pi\sqrt{163}} = -e^{-\pi\sqrt{163}}.$$

Therefore $q^{-1} \approx -e^{\pi\sqrt{163}}$ and the expansion implies

$$j(\tau) = q^{-1} + 744 + O(q)$$

is extremely close to $q^{-1} + 744$.

For this specific CM point, complex multiplication yields the celebrated exact value

$$j\left(\frac{1 + \sqrt{-163}}{2}\right) = -640320^3.$$

Substituting $q^{-1} \approx -e^{\pi\sqrt{163}}$ gives

$$e^{\pi\sqrt{163}} \approx 640320^3 + 744,$$

and the error is on the order of $e^{-\pi\sqrt{163}}$, explaining why the approximation is absurdly accurate. [contributors, 2025, Cox, 2013, Weisstein, 2025]

7.21.7 6. Analytic class number formula (bridge to L-functions)

A second "high-level" way to see class numbers is the Dirichlet class number formula. For a negative fundamental discriminant $\Delta < 0$:

$$h(\Delta) = \frac{w\sqrt{|\Delta|}}{2\pi} L(1, \chi_\Delta).$$

Here:

- w is the number of roots of unity in $\mathcal{O}_K$; for imaginary quadratic fields, $w = 2$ except for $\Delta = -4$ (Gaussian integers, $w = 4$) and $\Delta = -3$ (Eisenstein integers, $w = 6$).
- $\chi_\Delta(n) = \left(\frac{\Delta}{n}\right)$ is the Kronecker symbol (a quadratic Dirichlet character),
- and

$$L(1, \chi_\Delta) = \sum_{n=1}^{\infty} \frac{\chi_\Delta(n)}{n}.$$

This formula is the doorway from "class number" into Dirichlet characters and L-functions, and it is a natural starting point for computational experiments around prime races / residue classes later. [Cox, 2013]

7.21.8 7. Computation-friendly viewpoints (good experiment hooks)

7.1 Counting reduced forms (elementary and visual)

For a negative fundamental discriminant Δ, enumerate all integer triples (a, b, c) with:

$$b^2 - 4ac = \Delta, \quad a > 0, \quad |b| \le a \le c,$$

apply the reduced-form tie-break rules, and count reduced forms. That count is $h(\Delta)$.

For the Heegner discriminants, you will find exactly one reduced class.

7.2 "Almost integers" via truncated q-series

Choose $d \in \{19, 43, 67, 163\}$ and set $\tau = (1 + \sqrt{-d})/2$. Compute $q = e^{2\pi i \tau}$ numerically and compare:

$$q^{-1} + 744 \quad \text{vs.} \quad j(\tau)$$

approximating $j(\tau)$ by truncating its q-expansion. The gap shrinks dramatically as d increases, with $d = 163$ the most striking.

7.21.9 8. Common confusion: Heegner numbers vs Heegner points

- **Heegner numbers**: the nine d for which $\mathbb{Q}(\sqrt{-d})$ has class number 1.
- **Heegner points**: CM points on modular curves / elliptic curves used in deep results about ranks of elliptic curves (e.g. Gross–Zagier and Kolyvagin).

They share the same complex multiplication background, but they are different objects.

7.21.10 References

- Heegner [1952]
- Stark [1967]
- Stark [1969]
- Baker [1966]
- Cox [2013]
- Diamond and Shurman [2005]
- Weisstein [2025]
- contributors [2025]

7.22 Jordan totient $J_k(n)$ refresher

Jordan's totient function generalizes Euler's totient. It is multiplicative and appears naturally in group-counting problems. See [Apostol, 1976] and [Tenenbaum, 2015].

7.22.1 Definition

For a fixed integer $k \ge 1$, the **Jordan totient** function $J_k(n)$ can be defined by

$$J_k(n) = n^k \prod_{p \mid n} \left(1 - \frac{1}{p^k}\right).$$

For $k = 1$, this is Euler's totient: $J_1(n) = \varphi(n)$.

7.22.2 Divisor-sum identity

A useful identity is

$$\sum_{d|n} J_k(d) = n^k.$$

In Dirichlet convolution language:

$$J_k * 1 = \mathrm{id}^k,$$

hence $J_k = \mu * \mathrm{id}^k$.

7.22.3 Experiment ideas

- compare $J_k(n)$ as k varies
- visualize $J_2(n)$ and $J_3(n)$ over $n \le N$
- compare $J_k(n)/n^k$ as a "prime factor penalty"

7.22.4 Experiments in this repository

- **E099** — Jordan totients J_k: atlas, identities (J_1=), and scaling J_k(n)/n^k.

7.23 Landau's problems refresher

At the 1912 International Congress of Mathematicians (Cambridge), Edmund Landau highlighted four "basic" open problems about prime numbers. They became known as **Landau's problems**. [Pintz, 2009, Wikipedia contributors, 2026]

The striking aspect is that the statements are easy to explain to a beginner, but none of them has been proved so far.

7.23.1 The four problems (informal statements)

Landau's list is usually presented as:

1. **Goldbach (binary):** Every even integer ≥ 4 is a sum of two primes.
2. **Twin primes:** There are infinitely many primes p such that $p + 2$ is also prime.
3. **Legendre's conjecture:** For every $n \ge 1$, there is a prime between n^2 and $(n + 1)^2$.
4. **Primes of the form $n^2 + 1$:** There are infinitely many primes among $n^2 + 1$.

Landau's original phrasing and historical context are discussed nicely in Pintz's survey. [Pintz, 2009]

7.23.2 Why these problems are "simple but hard"

A recurring theme is that **primes behave like random numbers in many respects**, but not enough is known to turn probabilistic intuition into proofs.

Two typical obstacles:

- **Parity barrier:** Many sieve methods can show "almost primes" (numbers with few prime factors), but they struggle to force *exactly one* prime factor.
- **Correlation control:** Conjectures like "twin primes" require proving that primality events at two nearby integers are correlated often enough.

7.23.3 What experiments often do

Landau's problems are perfect targets for "experimental math" because you can explore the *shape* of the evidence:

- **Goldbach:** empirical coverage, number of representations, typical smallest prime in a representation.
- **Twin primes:** counts up to x, normalized by $x/(\log x)^2$ (heuristics).
- **Legendre:** check prime gaps near squares; visualize primes in intervals $[n^2, (n+1)^2]$.
- n^2+1 **primes:** count prime values of n^2+1 for $n \le N$; compare against heuristic $\sim C\,N/\log N$ (Bateman–Horn-style intuition).

7.23.4 Practical numerical caveats

- **Cost growth:** naive primality tests become slow as ranges grow; use a fast deterministic/probabilistic test (e.g. Miller–Rabin) and cache/sieve where possible.
- **Bias in small ranges:** for small N, "constants" in asymptotics are hard to see; show uncertainty bands or multiple scales.
- **Overflow:** n^2 grows quickly; use Python integers and avoid intermediate floats.

7.23.5 References

See ../references.

[Guy, 2004, Pintz, 2009, Wikipedia contributors, 2026]

7.24 Liouville function $\lambda(n)$ refresher

The Liouville function is a simple, oscillatory completely multiplicative function built from $\Omega(n)$ (the total number of prime factors with multiplicity). See [Tenenbaum, 2015].

7.24.1 Definition

Define

$$\lambda(n) = (-1)^{\Omega(n)}.$$

So $\lambda(n) = 1$ if n has an even total number of prime factors (with multiplicity), and $\lambda(n) = -1$ otherwise.

Examples:

- $\lambda(1) = 1$
- $\lambda(2) = -1$
- $\lambda(4) = \lambda(2^2) = 1$
- $\lambda(12) = \lambda(2^2 \cdot 3) = -1$

7.24.2 Key properties

- **Completely multiplicative:** $\lambda(ab) = \lambda(a)\lambda(b)$ for all a, b.
- Connected to the Möbius function, but without the "squarefree = 0" behavior.

7.24.3 Summatory function experiments

A standard experiment is the summatory function

$$L(x) = \sum_{n \le x} \lambda(n),$$

which oscillates and has deep connections to primes and zeta-function methods.

7.24.4 Experiment ideas

- compare $L(x)$ with a random walk
- compare $\lambda(n)$ and $\mu(n)$ on squarefree numbers
- visualize $\lambda(n)$ as a ± 1 texture over the integer grid

7.25 Mersenne numbers and primes refresher

This page is a *beginner-friendly* refresher for experiments about **Mersenne numbers** and **Mersenne primes**.

7.25.1 Core definitions

The **Mersenne numbers** are the integers

$$M_n = 2^n - 1 \quad (n \in \mathbb{N}).$$

A **Mersenne prime** is a Mersenne number that is prime:

$$M_p \text{ is a Mersenne prime} \iff 2^p - 1 \text{ is prime.}$$

A key (easy) fact is that if $2^n - 1$ is prime, then n must be prime.
So, in experiments, you typically only test M_p for *prime* exponents p. [contributors, 2025]

7.25.2 Key theorem / test: Lucas–Lehmer (why Mersenne primes are "computable")

For an odd prime exponent p, define $M_p = 2^p - 1$ and a sequence $\{s_k\}$ by

$$s_0 = 4, \qquad s_{k+1} = s_k^2 - 2.$$

Compute this sequence *modulo* M_p at each step, and then:

$$M_p \text{ is prime} \iff s_{p-2} \equiv 0 \pmod{M_p}.$$

This is the **Lucas–Lehmer test**. It's the workhorse behind most practical Mersenne-prime checks (and historically central to GIMPS). [contributors, 2025, Mersenne Research, 2024]

7.25.3 What experiments typically visualize

- **Growth with the exponent:** number of bits / digits of M_p as p grows.
- **Prime vs. composite behavior:** the Lucas–Lehmer residue s_{p-2} mod M_p across many prime exponents.
- **Factor patterns for composites:** quickly finding small factors of M_p (to avoid running LLT when a trivial factor exists).
- **Connections to perfect numbers:** if M_p is prime, then $2^{p-1}(2^p - 1)$ is an even perfect number (Euclid–Euler). [Caldwell, n.d.]

For curated sequences (lists of exponents / known values), OEIS is a convenient "ground truth" reference. [Inc., 2025, Inc., 2025]

7.25.4 Practical numerical caveats

- **Always reduce modulo M_p in Lucas–Lehmer.**
  If you don't, the intermediate values explode in size (each squaring roughly doubles the bit-length).
- **Test the exponent first.**
  If p is composite, M_p is automatically composite, so there's no point running LLT.
- **Huge integers are fine, but you must be intentional.**
  Python's big integers won't overflow, but performance depends on bit-length and on the efficiency of modular squaring.
- **Separate "demo scale" from "real scale."**
  For educational experiments, keep p modest (e.g., $p \leq 10^5$ is already huge for pure Python LLT). For large exponents, you'd rely on specialized implementations and careful FFT-based multiplication.

7.25.5 References

See ../references.

[Caldwell, 2021, contributors, 2025, contributors, 2025, Inc., 2025, Inc., 2025]

7.26 Möbius function $\mu(n)$ and Mertens function $M(x)$ refresher

The Möbius function $\mu(n)$ is the main tool for inversion over divisors. The Mertens function $M(x) = \sum_{n \leq x} \mu(n)$ is a classic "oscillatory" summatory function. See [Apostol, 1976], [Tenenbaum, 2015].

7.26.1 Möbius function

Define $\mu(1) = 1$. For $n > 1$:

- $\mu(n) = 0$ if n is divisible by the square of a prime (not squarefree)
- otherwise $\mu(n) = (-1)^k$ where k is the number of distinct prime factors of n

So for squarefree $n = p_1 p_2 \cdots p_k$,

$$\mu(n) = (-1)^k.$$

7.26.2 Möbius inversion (most important identity)

If

$$F(n) = \sum_{d|n} f(d),$$

then

$$f(n) = \sum_{d|n} \mu(d)\, F\left(\frac{n}{d}\right).$$

7.26.3 Mertens function and a famous counterexample

The **Mertens function** is

$$M(x) = \sum_{n \leq x} \mu(n).$$

A historic conjecture was $|M(x)| < \sqrt{x}$ for all $x \geq 1$ (the **Mertens conjecture**). It was disproved by Odlyzko and te Riele. [Odlyzko and te Riele, 1985]

7.26.4 Why $M(x)$ is experiment-friendly

- $M(x)$ exhibits sign changes and irregular growth ("random walk-like" behavior).
- It is deeply connected to the Riemann zeta function and the Riemann hypothesis.

Typical experiment: plot $M(x)$ for growing x and compare to envelopes like $\pm\sqrt{x}$ or $\pm x^{1/2}\log x$.

7.26.5 Experiments in this repository

- **E105** — Mertens function M(x) at multiple scales (including rescaling views).

7.27 Partition function $p(n)$ refresher

The partition function $p(n)$ counts the number of ways to write n as a sum of positive integers, ignoring order (e.g. $4 = 4 = 3+1 = 2+2 = 2+1+1 = 1+1+1+1$ gives $p(4) = 5$). See [Andrews, 1984].

7.27.1 Definition

A **partition** of n is a multiset of positive integers with sum n. The partition function $p(n)$ is the number of such partitions.

Conventions:

- $p(0) = 1$ (empty partition)
- $p(n) = 0$ for $n < 0$

7.27.2 Generating function

The ordinary generating function is

$$\sum_{n\geq 0} p(n)q^n = \prod_{m\geq 1}\frac{1}{1-q^m}.$$

This identity is the starting point for many algorithms and proofs.

7.27.3 Growth

$p(n)$ grows very rapidly (roughly like $\exp(C\sqrt{n})$), so experiments often use log-scales or focus on modular patterns.

7.27.4 Experiment ideas

- compute $p(n)$ for $n \leq N$ via dynamic programming and plot $\log p(n)$
- explore congruences (e.g. Ramanujan-type congruences)
- compare exact values to asymptotic approximations (later experiment)

7.28 Perfect numbers refresher

This page is a *beginner-friendly* refresher for experiments about **perfect numbers**. You only need basic number theory facts (divisors, primes) to follow it.

7.28.1 Core definitions

For a positive integer n, let

- $d \mid n$ mean "d divides n",
- $\sigma(n) = \sum_{d\mid n} d$ be the **sum-of-divisors function**,

- $s(n) = \sum_{d|n,\ d<n} d = \sigma(n) - n$ be the sum of **proper** divisors.

A number n is **perfect** iff its proper divisors sum to itself:

$$n \text{ is perfect} \iff s(n) = n \iff \sigma(n) = 2n.$$

Examples:

- 6 is perfect because $1 + 2 + 3 = 6$.
- 28 is perfect because $1 + 2 + 4 + 7 + 14 = 28$.

7.28.2 Key theorem: all even perfect numbers (Euclid–Euler)

A classic result completely characterizes **even** perfect numbers:

Euclid–Euler theorem.
An integer n is an even perfect number **iff**

$$n = 2^{p-1}(2^p - 1)$$

where $2^p - 1$ is prime (a **Mersenne prime**).

So every known perfect number is generated from a Mersenne prime exponent p. This is the main "generator" you'll use in experiments. [Caldwell, n.d., Voight, 1998]

Why σ matters (multiplicativity)

If n factors as

$$n = \prod_{i=1}^{k} p_i^{a_i},$$

then

$$\sigma(n) = \prod_{i=1}^{k} \sigma\left(p_i^{a_i}\right) = \prod_{i=1}^{k} \frac{p_i^{a_i+1} - 1}{p_i - 1}.$$

This formula turns "sum all divisors" into a fast computation once you know the prime factorization.

7.28.3 The big open question: odd perfect numbers

It is **unknown** whether any **odd** perfect numbers exist. A large literature proves *constraints* (congruences, size bounds, number of prime factors, etc.). Experiments can explore these constraints (and why they make brute-force search unrealistic). [Guy, 2004, Ochem and Rao, 2014, Stone, 2024]

7.28.4 What experiments typically visualize

Typical "lab" questions you can turn into plots and tables:

- **Verification by computation:** compute $\sigma(n)$ (via factorization) and check $\sigma(n) = 2n$ for candidates.
- **Generator experiment:** produce even perfect numbers from known Mersenne exponents p and confirm perfection.
- **Growth:** number of digits / bit length of $2^{p-1}(2^p - 1)$ as a function of p.
- **Divisor-function behavior:** compare $\sigma(n)/n$ (abundancy index) for random n vs. perfect numbers.
- **Odd constraints (toy models):** test necessary conditions on odd n and see how restrictive they are.

For data (lists of perfect numbers and exponents), OEIS is a convenient reference. [OEIS Foundation Inc., 2025]

7.28.5 Practical numerical caveats

Even with correct math, computation has a few traps:

- **Factorization dominates.** Computing $\sigma(n)$ via divisors is slow unless you factor n. For large n, factorization becomes infeasible; prefer the Euclid–Euler generator for even perfect numbers.
- **Big integers are fine but expensive.** Python integers won't overflow, but operations on huge numbers scale with the number of bits (so be mindful in loops and plotting).
- **Prime testing vs. proof.** Testing that $2^p - 1$ is prime is nontrivial for large p. For experiments, use a curated list of known Mersenne prime exponents (e.g., from OEIS / GIMPS) rather than trying to discover new ones from scratch. [Mersenne Research, Inc. (GIMPS), 2024, Mersenne Research, Inc. (GIMPS), 2025]
- **Be explicit about definitions.** Some sources define "perfect" via $\sigma(n) = 2n$; others via proper divisors. Use one convention consistently in code and docs.

7.28.6 References

See ../references.

[Caldwell, n.d., Stone, 2024, Voight, 1998, OEIS Foundation Inc., 2025]

7.29 Pretentious number theory refresher

"Pretentious" number theory replaces some zero-driven arguments by measuring how closely a multiplicative function *pretends* to be a simple model (typically n^{it} or a Dirichlet character). It is especially effective for **comparative** experiments: *which model explains the data best?*

7.29.1 Core definitions

Let f, g be multiplicative functions bounded by 1 in magnitude on primes. The **pretentious distance** up to x is

$$\mathbb{D}(f, g; x)^2 = \sum_{p \le x} \frac{1 - \Re\big(f(p)\overline{g(p)}\big)}{p}.$$

Interpretation:

- $\mathbb{D}(f, g; x)$ small means $f(p)$ and $g(p)$ "agree" on most primes (in a $1/p$-weighted sense).
- $\mathbb{D}(f, g; x)$ large means f cannot be explained well by g on primes.

A common model family is $g(n) = \chi(n)n^{it}$ with a Dirichlet character χ and real t.

7.29.2 What experiments usually visualize or measure

- Fit t to minimize $\mathbb{D}(f, n^{it}; x)$ and plot the best-fit $t(x)$.
- Compare distances $\mathbb{D}(f, \chi; x)$ across characters χ to see which congruence structure matches f.
- Use the distance to explain why partial sums of $f(n)$ may be large or small.

7.29.3 Practical numerical caveats

- Always define the prime cutoff (use the same x for fair comparisons).
- Distance is dominated by small primes; if you want to study "large-prime behavior," consider plotting contributions by prime ranges.
- When fitting t, the objective can have shallow minima; report robustness (e.g., multiple initial guesses).

7.29.4 References

See ../references.

[Granville, 2009, Granville and Soundararajan, 2007]

7.29.5 Experiments in this repository

- **E120** — Pretentious distance atlas for core multiplicative functions (, , /n, …).

7.30 Primality testing: guarantees, error bounds, and what to report

This page explains how we talk about primality tests in **py-mathx-lab**: what is guaranteed, what is probabilistic, and what must be recorded in experiment reports.

Core references: [Crandall and Pomerance, 2005, Rabin, 1980].

7.30.1 Deterministic vs probabilistic (project language)

A primality test can have different guarantees:

- **Deterministic**: returns the correct answer for every input in its stated domain.
- **Probabilistic (one-sided error)**:
  - If it returns **composite**, that is *always correct*.
  - If it returns **probably prime**, that can be wrong with some probability.

Project reporting rule: every experiment that uses a probabilistic test must state:

- "probabilistic vs deterministic" (and the domain if deterministic),
- bases / rounds used,
- a conservative error probability statement (when available),
- and known counterexamples / failure modes when applicable.

(Practically: put this into `out/e###/report.md` as a short, explicit section.)

7.30.2 Fermat probable prime test (FPP)

For odd $n > 2$ and a base a with $\gcd(a, n) = 1$, Fermat's congruence says:

$$a^{n-1} \equiv 1 \pmod{n} \quad \text{for prime } n.$$

If the congruence fails, n is composite (a **witness** was found).
If it passes, n is a **Fermat probable prime** to base a.

Why Fermat alone is not reliable

There exist composite numbers that pass Fermat's test for many bases. In particular, **Carmichael numbers** pass the test for *all* bases coprime to n. So Fermat is best treated as a **cheap pre-filter**, not a strong claim of primality.

See *Carmichael numbers*.

Known counterexamples (Fermat)

- $341 = 11 \cdot 31$ is a classic base-2 pseudoprime.
- $561 = 3 \cdot 11 \cdot 17$ is the smallest Carmichael number.

7.30.3 Miller–Rabin (strong probable prime test)

Write

$$n - 1 = d\,2^s \quad (d \text{ odd}).$$

For a chosen base a, Miller–Rabin tests a short chain of squarings modulo n and returns:

- **composite** (a witness was found), or
- **strong probable prime** to base a.

Standard error bound (what you may claim)

For any odd composite n, at most one quarter of bases $a \in \{2, \ldots, n-2\}$ can make n look prime to the Miller–Rabin test. Therefore, if you test k **independent random bases**,

$$\Pr(\text{composite passes all } k \text{ rounds}) \le 4^{-k}.$$

This is the classical Rabin bound. [Rabin, 1980]

Practical "deterministic" statements

Sometimes you will see statements like "Miller–Rabin is deterministic for 64-bit integers using a fixed base set." Such claims depend on **external results** that specify an input range and a particular base set.

In this project:

- If you use **random bases**, treat MR as probabilistic and report 4^{-k}.
- If you use a **fixed base list**, report it as an engineering choice, and only call it deterministic if your documentation *also* cites a theorem covering your input range.

7.30.4 Pipeline viewpoint (how tests combine)

In experiments, primality tests often appear as components in a larger pipeline:

1. Handle trivial cases: $n < 2$, even numbers, perfect squares.
2. Trial division up to a small bound (fast removal of tiny factors).
3. Miller–Rabin as a strong "probable prime" filter (with stated guarantee).
4. If composite but no factor found: a factor-finding step (e.g. Pollard–ρ) and recursion.

See *Semiprimes* and *Prime numbers refresher* for context.

7.30.5 Report checklist (copy/paste into `report.md`)

Use a short section in `out/e###/report.md` that states:

- **Status**: deterministic vs probabilistic
- **Method**: Fermat / MR / combination
- **Bases / rounds**:
  - fixed list (write the list), or
  - random rounds k (state how bases are sampled)
- **Error statement**:
  - MR random rounds: $P(\text{false prime}) \le 4^{-k}$
  - Fermat: no uniform bound; mention Carmichael numbers
- **Correctness cross-check**:

  - compare decisions to a trusted reference on a CI-safe range
- **Known counterexamples / failure modes**:
  - include at least one concrete counterexample when relevant (e.g. 341, 561)

7.30.6 References

- [Crandall and Pomerance, 2005]
- [Rabin, 1980]

7.31 Prime counting approximations: π(x), Li(x), and R(x)

The prime-counting function (\pi(x)) counts primes up to (x). Two standard smooth approximations are:

- the **logarithmic integral** (\mathrm{Li}(x)), and
- **Riemann's R function** (R(x)) (a Möbius-weighted combination of (\mathrm{Li}(x^{1/n}))).

These approximations are central for numerical experiments around the Prime Number Theorem, error terms, and the visual "shape" of prime races.

7.31.1 Key ideas

- **PNT baseline:** (\pi(x)\sim \frac{x}{\log x}), and (\mathrm{Li}(x)) is often an excellent smooth baseline.
- **R(x) as a refinement:** R(x) folds in prime-power information and is closely connected to explicit-formula viewpoints.
- **Error terms:** plotting (\pi(x)-\mathrm{Li}(x)) or (\psi(x)-x) reveals oscillations and sign changes.

7.31.2 References

See ../references.

[Titchmarsh, 1986, Wikipedia contributors, 2025]

7.32 Prime counting: explicit bounds (not just asymptotics)

Prime counting experiments often start from approximations (PNT-style) such as $x/\log x$. For Phase 2, we also want **explicit inequalities**: formulas that are proven to hold for all x in a stated range.

This page focuses on how to use explicit bounds correctly in experiments:

- put the bound formula in the report,
- name the theorem/source,
- verify it numerically on the plotted range,
- and avoid overselling asymptotic statements at small x.

7.32.1 The prime counting function

Let

$$\pi(x) = \#\{p \le x : p \text{ prime}\}.$$

For approximations, see *Prime counting approximations: (x), Li(x), and R(x)*.

7.32.2 Example: Dusart-style explicit bounds

A convenient family of explicit bounds uses expansions in $1/\log x$. For example, Dusart gives (among many results) inequalities of the form:

$$\pi(x) \geq \frac{x}{\log x}\left(1 + \frac{1}{\log x}\right) \quad \text{for } x > 599,$$

and

$$\pi(x) \leq \frac{x}{\log x}\left(1 + \frac{1.2762}{\log x}\right) \quad \text{for } x > 1.$$

(See Theorem 6.9 in [Dusart, 2010].)

These bounds are not the tightest known, but they are easy to implement and they demonstrate the discipline: **formula + validity range**.

7.32.3 Stronger bounds exist

Classical work such as Rosser–Schoenfeld provides explicit inequalities for $\pi(x)$ and related functions. See [Rosser and Schoenfeld, 1962]. More recent refinements and alternative explicit estimates are discussed in [Dusart, 2018].

7.32.4 How to use a bound in an experiment

1) Put the theorem statement in the report

In `out/e###/report.md`, include:

- the exact inequality you implemented,
- the stated validity range,
- and the precise meaning of symbols (e.g. natural log).

2) Verify numerically on your plotted range

If you plot $\pi(x)$ and a bound $B(x)$ on $x \in [A, B]$, then verify the inequality on that same grid:

- check `pi(x) <= B(x)` (or the other direction) at each grid point,
- report the first violation (if any) as a witness.

This does **not** prove the theorem, but it catches implementation mistakes.

3) State the range where the bound becomes meaningful

A bound can be true but useless for small x. To report "where it becomes meaningful", pick an operational criterion. For example:

- relative gap `|B(x) - pi(x)| / pi(x) <= 0.05`, or
- relative gap compared to $x/\log x$, or
- a fixed vertical error tolerance.

Then report the smallest x in your sampled range where the criterion holds. Label this as **finite-range behavior**.

4) Avoid asymptotic overclaims

Statements like "$\pi(x) \sim x/\log x$" are asymptotic. They describe what happens as $x \to \infty$. In reports and captions:

- keep the phrase "asymptotic" for the theory statement,
- and use "finite-range behavior" for what your plot shows.

7.32.5 How this connects to experiments

- Approximations and PNT-style plots: *Prime counting approximations: (x), Li(x), and R(x)*
- Arithmetic progressions and prime races: *Dirichlet's theorem and PNT(AP) in the form used by the experiments*, *Prime number races refresher*

7.32.6 References

See ../references.

7.33 Prime-factor counting: $\omega(n)$ and $\Omega(n)$ refresher

Prime-factor counts are central examples of additive arithmetic functions. They are key in probabilistic number theory and in "typical behavior" experiments. See [Tenenbaum, 2015].

7.33.1 Definitions

Let $n = \prod p_i^{\alpha_i}$.

- $\omega(n)$: number of **distinct** prime factors

$$\omega(n) = \#\{p : p \mid n\}.$$

- $\Omega(n)$: number of prime factors **with multiplicity**

$$\Omega(n) = \sum_i \alpha_i.$$

Examples:

- $12 = 2^2 \cdot 3$ has $\omega(12) = 2$ and $\Omega(12) = 3$.

7.33.2 Additivity

If $\gcd(a, b) = 1$ then

$$\omega(ab) = \omega(a) + \omega(b),$$

and Ω is even completely additive:

$$\Omega(ab) = \Omega(a) + \Omega(b) \quad \text{for all } a, b.$$

7.33.3 Typical size: Erdős–Kac phenomenon

A famous theorem of Erdős and Kac says (informally) that $\omega(n)$ behaves like a normal random variable with mean and variance $\log\log n$ after normalization. [Erdős and Kac, 1940]

This is why histograms of $\omega(n)$ over ranges often look Gaussian.

7.33.4 Experiment ideas

- histogram of $\omega(n)$ for $n \le N$ and compare to a normal curve
- compare $\omega(n)$ vs. $\log\log n$ ("normal order")
- scatter $\Omega(n)$ vs. $\omega(n)$ to see multiplicities

7.33.5 Experiments in this repository

- **E094** — (n) vs Ω(n): Erdős–Kac side-by-side (distribution + scaling).
- **E095** — Squarefree conditioning ((n)≠0): forces (n)=Ω(n).
- **E122** — Heatmap atlas of , , Ω (visual textures / patterns).

7.34 Prime number races refresher

A **prime number race** compares prime counts in different residue classes, e.g.

$$\pi(x; 4, 3) \text{ vs. } \pi(x; 4, 1).$$

Although $\pi(x; q, a) \sim \mathrm{Li}(x)/\varphi(q)$ suggests "no winner," finite ranges often show persistent preferences (biases).

How this repository measures a race (Phase 2 convention). For each reduced residue class a (with $\gcd(a, q) = 1$) we compare $\pi(x; q, a)$ to the baseline $\mathrm{li}(x)/\varphi(q)$ and define the error term

$$E(x; q, a) := \pi(x; q, a) - \frac{\mathrm{li}(x)}{\varphi(q)}.$$

Race curves are typically differences such as $\Delta(x) = \pi(x; q, a) - \pi(x; q, b)$; the baseline cancels, so $\Delta(x) = E(x; q, a) - E(x; q, b)$. For the exact definitions and the baseline/error plots used throughout Phase 2, see *Dirichlet's theorem and PNT(AP) in the form used by the experiments*.

7.34.1 The classic example: Chebyshev's bias mod 4

Empirically, for many ranges of x one observes

$$\pi(x; 4, 3) > \pi(x; 4, 1),$$

even though both are asymptotically equal.

Rubinstein–Sarnak analyze this phenomenon through the lens of L-function zeros and propose models for the distribution of race leaders.

7.34.2 Practical notes for experiments

- Bias claims depend on *how you sample* x (linear vs log-grid); be explicit.
- With small cutoffs (say $x \le 10^7$), you'll see "apparent stability" that may later flip; that's expected.
- Counting primes is the bottleneck; keep an eye on runtime and memory, and record parameters in your manifest.

7.34.3 References

See ../references.

[Granville and Martin, 2006, Rubinstein and Sarnak, 1994]

7.34.4 Experiments in this repository

- **E112** — Prime race curves $\pi(x; q, a) - \pi(x; q, b)$: sign changes and time-in-lead.

7.35 Prime numbers refresher

This page is a *beginner-friendly* refresher for experiments about **prime numbers**. It is intentionally broad: it covers the prime-related ideas most commonly explored with computation (distribution, sieves, primality testing, gaps, and prime patterns).

7.35.1 Core definitions

Primes and composites

A positive integer $p \geq 2$ is **prime** if its only positive divisors are 1 and p. If $n \geq 2$ is not prime, it is **composite**.

Equivalently,

$$p \text{ is prime} \iff \forall\, a, b \in \mathbb{N} :\ p = ab \Rightarrow (a = 1 \text{ or } b = 1).$$

Greatest common divisor and coprimality

The **greatest common divisor** of integers a, b, written $\gcd(a, b)$, is the largest positive integer dividing both. We say a and b are **coprime** if $\gcd(a, b) = 1$.

Coprimality appears constantly in modular arithmetic and in many prime-related proofs.

Congruences (modular arithmetic)

For integers a, b and $m \geq 2$,

$$a \equiv b \pmod{m}$$

means m divides $a - b$. Many prime experiments (residue classes, sieves, primality tests) are naturally expressed modulo m.

Prime factorization (the "atomic" viewpoint)

The **Fundamental Theorem of Arithmetic** says every integer $n \geq 2$ factors uniquely (up to ordering) as

$$n = \prod_{i=1}^{k} p_i^{\alpha_i},$$

where the p_i are distinct primes and the α_i are positive integers. This makes primes the "building blocks" of the integers. [Hardy and Wright, 2008]

7.35.2 Key quantitative language

Infinitely many primes

Euclid's classic argument shows there are infinitely many primes. This matters experimentally because it justifies questions like "how often do primes occur as numbers grow?" [Hardy and Wright, 2008]

The prime counting function and the Prime Number Theorem

Let $\pi(x)$ be the number of primes $\leq x$. The **Prime Number Theorem** (PNT) states

$$\pi(x) \sim \frac{x}{\log x} \quad (x \to \infty).$$

Informally: primes thin out, and the "typical" gap size near x is about $\log x$. [Apostol, 1976]

For experiments, two very common comparisons are:

- $\pi(x)$ vs. $x/\log x$,
- $\pi(x)$ vs. $\mathrm{Li}(x)$ (the logarithmic integral), which is often a better approximation in practice. [Apostol, 1976, Rosser and Schoenfeld, 1962]

Explicit bounds (useful sanity checks)

Many sources provide explicit inequalities for $\pi(x)$, the nth prime p_n, and related functions. These bounds are useful to validate computations and pick experiment ranges safely. [Axler, 2019, Dusart, 2018, Rosser and Schoenfeld, 1962]

7.35.3 Computational building blocks

Generating primes: sieves

If you need *all* primes up to a bound N, a sieve is usually best.

Sieve of Eratosthenes (idea). Start with a boolean array "possibly prime," then repeatedly cross out multiples of each found prime. Time is roughly $O(N \log \log N)$, memory is $O(N)$ booleans.

Segmented sieve. For large N, store only a window $[L, R]$ at a time; this keeps memory bounded while preserving speed. Segmented sieves are often the right choice once N grows beyond typical RAM-friendly sizes. [Crandall and Pomerance, 2005]

Testing primality (single numbers)

If you need to test the primality of individual (possibly large) integers, common strategies are:

- **Trial division** up to $\sqrt{n}$ (great for small n, too slow for large),
- **Probabilistic tests** such as **Miller–Rabin**, fast and extremely reliable with good parameters, [Rabin, 1980]
- **Deterministic polynomial-time** primality testing (AKS), important theoretically but rarely used for practical large-number work. [Agrawal *et al.*, 2004]

In an experiment repo, it is common to use Miller–Rabin for speed and then either: (1) accept "probable prime" status (with a stated error bound) or (2) confirm with stronger checks for the sizes you care about. [Crandall and Pomerance, 2005]

Factorization (often the bottleneck)

Many arithmetic functions become easy once a number is factored, but factoring large integers is hard. Even "toy" factorization methods (e.g., Pollard's rho) make excellent experiments because performance varies wildly with input structure. [Crandall and Pomerance, 2005]

7.35.4 Prime gaps and prime patterns

Prime gaps

Let p_n be the nth prime. The **prime gap** sequence is

$$g_n = p_{n+1} - p_n.$$

Heuristically, "typical" gaps near p are size $\log p$, but gaps vary a lot. Two natural experimental viewpoints are:

- **local statistics**: histogram of g_n in a range,
- **record gaps**: the largest gap seen up to x (maximal gaps / first occurrences). [Caldwell, n.d., Cramér, 1936, Nicely, 1999]

A common normalization is $g_n / \log p_n$, which helps compare different scales.

Twin, cousin, and sexy primes (prime pairs)

A **prime pair** is a pair of primes $(p, p + d)$ with fixed even difference d:

- **twin primes**: $d = 2$,
- **cousin primes**: $d = 4$,
- **sexy primes**: $d = 6$.

These are all instances of "prime constellations."

Prime k-tuples (constellations)

A **prime k-tuple** is a set of offsets $\{h_1, \ldots, h_k\}$ such that $p + h_i$ are simultaneously prime. Hardy–Littlewood heuristics predict asymptotic counts for many such patterns (including twins). [Hardy and Littlewood, 1923]

Modern sieve breakthroughs show **bounded prime gaps** exist (there are infinitely many g_n below some constant), but we still do not know whether the twin prime conjecture is true. [Maynard, 2015, Zhang, 2014, D. H. J. Polymath, 2014]

7.35.5 Primes in structure (residue classes and progressions)

Residue classes

Primes (except 2) are odd, so they lie in residue classes modulo 2. More generally, you can study how primes distribute among residue classes modulo q. Experiments here include "prime races" and small biases in counts.

Arithmetic progressions of primes

A classical result (Dirichlet's theorem) says that if $\gcd(a, q) = 1$, then the arithmetic progression

$$a,\ a + q,\ a + 2q, \ldots$$

contains infinitely many primes. A landmark modern result (Green–Tao) shows primes contain arbitrarily long arithmetic progressions. [Green and Tao, 2008]

For experiments, you can stay at an elementary level (searching for 3-term or 4-term prime APs in ranges) while still getting meaningful patterns.

7.35.6 Other prime families you may include (optional)

These are often nice "side quests" that still fit under "Prime Numbers":

- **Sophie Germain primes**: p prime and $2p + 1$ prime,
- **safe primes**: q prime with $(q - 1)/2$ prime,
- **Mersenne primes**: $2^p - 1$ prime (you already have a dedicated page), [contributors, 2025]
- **palindromic / repunit primes**: "pattern-based" primes useful for search experiments.

7.35.7 What experiments typically visualize

Typical "prime numbers lab" plots and tables include:

- **Prime density:** $\pi(x)$ vs. $x/\log x$, and error curves $\pi(x) - x/\log x$. [Apostol, 1976]
- **Bounds and approximations:** compare $\pi(x)$ with explicit inequalities (sanity checks). [Dusart, 2018, Rosser and Schoenfeld, 1962]
- **Sieve scaling:** runtime/memory vs. N for classic vs. segmented sieves. [Crandall and Pomerance, 2005]
- **Gaps:** histograms of g_n, normalized gaps $g_n/\log p$, record gaps vs. $\log^2 x$. [Cramér, 1936, Nicely, 1999]
- **Prime pairs:** counts of twin/cousin/sexy primes in $[1, x]$, compared to Hardy–Littlewood style heuristics. [Hardy and Littlewood, 1923]
- **Bounded gaps story:** "what bound was known when?" (Zhang → Maynard → Polymath) plus empirical gap data. [Maynard, 2015, Zhang, 2014, D. H. J. Polymath, 2014]
- **Primality tests:** speed vs. integer size (bits), and false-positive rates for weak tests vs. Miller–Rabin. [Agrawal *et al.*, 2004, Rabin, 1980]

- **Progressions:** frequency of 3-term prime APs in ranges; maximal AP length in a window. [Green and Tao, 2008]

For "ground truth" sequences and curated lists, OEIS and PrimePages are practical reference points. [Caldwell, n.d., OEIS Foundation Inc., 2025, OEIS Foundation Inc., 2025]

7.35.8 Practical numerical caveats

- **Pick the right primitive:** if you need *many* primes up-to N, use a sieve; if you need primality of a few large numbers, use primality testing.
- **Memory matters for sieves:** a naive boolean list of length N can be huge; segmented sieves avoid this.
- **State what "prime" means in code:** if you use Miller–Rabin, your results are "probable primes" unless you add a deterministic guarantee for your size range. [Rabin, 1980]
- **Avoid floating-point traps:** use natural logs, guard against $\log(0)$, and remember that $\log x$ changes slowly.
- **Be precise with pair-counting:** decide whether you count pairs by their smaller prime p (common), and avoid double-counting.
- **Separate discovery from verification:** for pattern-hunting (e.g., long APs), do fast screening first, then verify candidates carefully.

7.35.9 References

See ../references.

[Agrawal *et al.*, 2004, Apostol, 1976, Cramér, 1936, Crandall and Pomerance, 2005, Dusart, 2018, Green and Tao, 2008, Hardy and Littlewood, 1923, Hardy and Wright, 2008, Maynard, 2015, Rabin, 1980, Rosser and Schoenfeld, 1962, Zhang, 2014, D. H. J. Polymath, 2014]

7.36 Dirichlet's theorem and PNT(AP) in the form used by the experiments

This page is the Phase 2 background for experiments about primes in residue classes and prime races. It states Dirichlet's theorem and the prime number theorem in arithmetic progressions (PNT(AP)) *exactly in the normalization used by the experiments*:

- baseline: $\mathrm{li}(x)/\varphi(q)$,
- error plots: $\pi(x;q,a) - \mathrm{li}(x)/\varphi(q)$ (and derived race differences).

7.36.1 The counting function $\pi(x;q,a)$

Fix integers $q \geq 1$ and a. Define

$$\pi(x;q,a) := \#\{p \leq x : p \text{ prime and } p \equiv a \pmod q\}.$$

Only **reduced residue classes** participate in the classical equidistribution theorems:

- If $\gcd(a,q) = 1$, then the congruence class $a \bmod q$ contains infinitely many primes.
- If $\gcd(a,q) > 1$, then the class contains at most one prime (the common divisor itself), so it is not part of the PNT(AP) story.

Let

$$(\mathbb{Z}/q\mathbb{Z})^\times = \{a \bmod q : \gcd(a,q) = 1\}$$

denote the reduced residue system, and let $\varphi(q) = \#(\mathbb{Z}/q\mathbb{Z})^\times$ be Euler's totient.

7.36.2 The baseline $\mathrm{li}(x)/\varphi(q)$

The logarithmic integral is

$$\mathrm{li}(x) := \mathrm{PV} \int_0^x \frac{dt}{\log t}.$$

In numerical work one often uses the "offset" version

$$\mathrm{Li}(x) := \int_2^x \frac{dt}{\log t},$$

and treats the two as essentially the same baseline for large x (they differ by a constant). Phase 2 uses the baseline in the form

$$\mathrm{baseline}(x; q) := \frac{\mathrm{li}(x)}{\varphi(q)}.$$

Interpretation: if primes are "evenly spread" across the $\varphi(q)$ reduced residue classes, then each class should get about a $1/\varphi(q)$ share of the total prime mass predicted by $\mathrm{li}(x)$.

7.36.3 Dirichlet's theorem (existence of primes in each reduced residue class)

Theorem (Dirichlet, 1837). If $\gcd(a, q) = 1$, then there are infinitely many primes p such that $p \equiv a \pmod{q}$.

This is the minimal statement the experiments rely on: every reduced residue class shows up forever, so "race leaders" can change infinitely often in principle.

Why characters and L-functions enter (one paragraph)

The proof introduces Dirichlet characters χ mod q and their L-functions

$$L(s, \chi) = \sum_{n \geq 1} \frac{\chi(n)}{n^s} \qquad (\Re(s) > 1),$$

and shows that for **nonprincipal** characters χ, the value $L(1, \chi)$ is nonzero. Using character orthogonality, one can express the indicator of a residue class a mod q as an average over characters, which lets one "filter primes by congruence class" and prove that every reduced class contains infinitely many primes.

7.36.4 Prime number theorem in arithmetic progressions (equidistribution)

Dirichlet's theorem says primes exist in each reduced class; PNT(AP) says they are *asymptotically equidistributed*.

Theorem (PNT(AP), fixed modulus form). Fix $q \geq 1$ and let $\gcd(a, q) = 1$. Then, as $x \to \infty$,

$$\pi(x; q, a) \sim \frac{\mathrm{li}(x)}{\varphi(q)}.$$

Equivalently, the difference

$$E(x; q, a) := \pi(x; q, a) - \frac{\mathrm{li}(x)}{\varphi(q)}$$

is "small compared to" $\mathrm{li}(x)/\varphi(q)$ in the limit $x \to \infty$.

7.36.5 The experiment's error-term plots

Most Phase 2 plots are based on $E(x;q,a)$. Typical visualizations include:

- **Raw error:** plot $E(x;q,a)$ vs x.
- **Multiple residues:** plot $E(x;q,a)$ for several $a \in (\mathbb{Z}/q\mathbb{Z})^\times$ on the same axes.
- **Race differences (baseline cancels):** for two residues a, b define

$$\Delta(x;q,a,b) := \pi(x;q,a) - \pi(x;q,b) = E(x;q,a) - E(x;q,b).$$

So, race experiments are directly about comparing error terms across residues.

What size should $E(x;q,a)$ have? (qualitative)

For the ranges used in computational experiments, it is normal that $E(x;q,a)$ oscillates and does not look "small" pointwise. The key phenomenon is not monotone convergence but **oscillation around the baseline**.

A useful benchmark statement (not required for running the experiments, but helpful for interpretation) is: under GRH one expects roughly

$$E(x;q,a) = O\big(\sqrt{x}\,\log x\big)$$

for fixed q (and more refined uniform statements are known with restrictions on q). This heuristic explains why, even when $\pi(x;q,a)$ is close to the baseline in relative terms, the raw difference can still have visible swings.

7.36.6 How L-function zeros explain oscillations (high level)

A guiding principle is: **zeros of Dirichlet L-functions drive oscillations in residue-class prime counts**. Very roughly, character orthogonality lets you write "error terms in a class" as sums over contributions from nonprincipal characters, and analytic number theory relates those contributions to the zeros of $L(s,\chi)$. You do not need to compute zeros for Phase 2, but this explains why race plots can show long-lasting biases and sign changes.

7.36.7 Practical numerical notes for experiments

- Always restrict to a with $\gcd(a,q) = 1$ when discussing equidistribution or races.
- Sampling matters: using a linear grid in x vs a log grid in x weights different ranges differently and can change "leader fractions".
- If you approximate $\mathrm{li}(x)$ numerically, document the method and keep it consistent across experiments (baseline consistency matters more than ultimate accuracy for qualitative plots).

7.36.8 References

See ../references.

[Davenport, 2000, Iwaniec and Kowalski, 2004, Lejeune Dirichlet, 1837, Montgomery and Vaughan, 2006]

7.36.9 Experiments in this repository

- **E070** — $\pi(x;q,a)$ for several reduced residue classes (baseline comparison).
- **E071** — $E(x;q,a) = \pi(x;q,a) - \mathrm{li}(x)/\varphi(q)$ error-term plots.
- **E072–E075** — Prime races and race distributions (differences of $\pi(x;q,a)$).
- **E081** — Effect of modulus on PNT(AP) error behavior (comparative error plots).

7.37 Primorials

The primorial of a prime p_k is

$$p_k\# = \prod_{i=1}^{k} p_i$$

(the product of the first k primes). Primorials are a natural "smoothing knob" in experimental number theory: they amplify small prime structure and appear in constructions like Euclid numbers $p_k\# \pm 1$. [Prime Pages (UTM), 2025, The OEIS Foundation Inc., 2025]

7.37.1 Key facts

- **Growth:** $\log(p_k\#) = \sum_{i \le k} \log p_i$; by the prime number theorem this is asymptotic to p_k (in a coarse sense). [Hardy and Wright, 2008]
- **Euclid numbers:** $p_k\# + 1$ is coprime to all primes $\le p_k$, but is usually composite (so: "Euclid's proof does *not* generate primes"). [Prime Pages (UTM), 2025]

7.37.2 What to experiment with

- **Euclid-number factorization:** For $E_k = p_k\# \pm 1$, try to find small factors and compare +1 vs. -1.
- **Primorial wheels:** Use primorials to build "wheel sieves" and benchmark against a simple sieve.
- **Primorial primes / plus/minus primes:** Scan for primes of the form $p_k\# \pm 1$ and visualize rarity.

7.37.3 References

See Prime Pages (UTM) [2025], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

7.38 Quadratic polynomials (algebraic) refresher

A **quadratic polynomial** over a field F is a degree-2 polynomial

$$f(x) = ax^2 + bx + c, \qquad a \neq 0, \quad a, b, c \in F.$$

Over $F = \mathbb{R}$ or $\mathbb{C}$ this is the familiar object from school algebra, but many experiments treat f over other fields (like $\mathbb{Q}$ or finite fields $\mathbb{F}_p$). [Dummit and Foote, 2004, Wikipedia contributors, 2026]

7.38.1 Completing the square

A standard normal form is obtained by completing the square:

$$ax^2 + bx + c = a\left(x + \frac{b}{2a}\right)^2 - \frac{b^2 - 4ac}{4a}.$$

This shows that (over a field of characteristic $\neq 2$) every quadratic is a shifted/scaled square plus a constant term.

7.38.2 Discriminant and factorization

The **discriminant** is

$$\Delta = b^2 - 4ac.$$

Over a field where square roots make sense, the roots are

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}.$$

Algebraically, this means:

- f **factors** over F iff Δ is a square in F (when $\text{char}(F) \neq 2$).
- If Δ is not a square, the polynomial is **irreducible** over F.

Finite field case ($\mathbb{F}_p$, odd prime p)

Over $\mathbb{F}_p$, "Δ is a square" means Δ is a quadratic residue mod p. This is the bridge between algebra and number theory: quadratic residues, Legendre symbols, and Gauss sums.

7.38.3 Why quadratics show up in prime experiments

Quadratics are the simplest non-linear polynomials, so they are a natural testbed for "prime values of polynomials":

- Example: $n^2 + 1$ (Landau's 4th problem).
- Example: $n^2 + n + 41$ (Euler's famous prime-producing streak).

Even when a polynomial produces many primes early on, modular obstructions (e.g., always divisible by some prime for some residue class) inevitably appear.

7.38.4 Practical numerical caveats

- **Stable quadratic formula:** over floating point numbers, use a stable variant when $b^2 \gg 4ac$ to avoid catastrophic cancellation.
- **Exact arithmetic:** for modular experiments, keep everything as integers mod p; don't mix floats.
- **Normalization:** many plots look cleaner after shifting/scaling to the monic form $x^2 + Bx + C$ (when working over a field with a^{-1}).

7.38.5 References

See ../references.

[Dummit and Foote, 2004, Wikipedia contributors, 2026]

7.39 Riemann zeta function ζ(s)

The **Riemann zeta function** (s) is the Dirichlet series

$$\zeta(s) = \sum_{n \geq 1} \frac{1}{n^s},$$

which converges for (\Re(s)>1) and extends (by analytic continuation) to a meromorphic function on (\mathbb{C}) with a single simple pole at (s=1).

7.39.1 Key ideas

- **Euler product (primes):** for (\Re(s)>1),

  $$\zeta(s) = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}},$$

  linking (s) directly to prime distribution.
- **Functional equation:** (s) satisfies a symmetry relating (s) and (1-s), which is central for studying zeros.
- **Zeros:** (s) has *trivial zeros* at negative even integers and *nontrivial zeros* in the critical strip (0<\Re(s)<1), conjecturally all on the critical line (\Re(s)=\tfrac12) (Riemann Hypothesis).

7.39.2 Why it matters in this project

Many "analytic prime number theory" numerics (prime counting approximations, explicit formulas, prime races, etc.) are most naturally expressed in terms of (s), its logarithmic derivative, and its zeros.

7.39.3 Experiments in this repository

- **E117** — -factor functional-equation consistency checks on a grid of s values.
- **E118** — Partial Euler product for (s): where it breaks as Re(s) decreases.

7.39.4 References

See ../references.

[Edwards, 1974, Ivić, 1985, Odlyzko, 1992, Titchmarsh, 1986, Wikipedia contributors, 2025]

7.40 Semiprimes

A semiprime is an integer that is the product of two primes (not necessarily distinct), i.e.

$$n = pq \quad (p, q \text{ prime}).$$

Semiprimes sit at the center of practical factorization hardness and are the basic objects behind RSA-style moduli. [Rivest *et al.*, 1978, The OEIS Foundation Inc., 2025]

7.40.1 Key facts

- **Sequence:** The semiprimes form OEIS sequence A001358. [The OEIS Foundation Inc., 2025]
- **RSA context:** RSA uses $N = pq$ and the difficulty of factoring N (when p, q are large and random-looking). [Rivest *et al.*, 1978]
- **Special cases:** $p = q$ yields squares of primes; $p \neq q$ yields "RSA-like" semiprimes.

7.40.2 What to experiment with

- **Distribution:** Count semiprimes up to X and compare empirical growth with simple heuristic approximations.
- **Factorization methods:** Benchmark trial division, Pollard's rho, and (for larger sizes) ECM bindings if you later allow optional deps.
- **RSA-style generation:** Generate balanced semiprimes (p, q same bitlength) and compare factorization difficulty vs unbalanced ones.
- **Smoothness:** Explore how often $p - 1$ or $q - 1$ is smooth (motivates $p - 1$ factoring methods).

7.40.3 References

See Rivest *et al.* [1978], The OEIS Foundation Inc. [2025].

7.41 Taylor series refresher

This page is a *beginner-friendly* refresher for experiments that use Taylor polynomials. You only need basic calculus (derivatives) to follow it.

7.41.1 Taylor polynomial

Assume f has enough derivatives near x_0 (this is true for $\sin(x)$, $\cos(x)$, polynomials, exponentials, etc.). The Taylor polynomial of degree n around x_0 is

$$T_n(x; x_0) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k.$$

Intuition: $T_n(x; x_0)$ is the polynomial that matches $f(x_0)$ and the first n derivatives at x_0. It is usually accurate when x is close to x_0.

For $f(x) = \sin(x)$, the derivatives cycle, and around $x_0 = 0$ this becomes

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots.$$

7.41.2 Truncation error and the remainder

The approximation error is the remainder

$$R_n(x; x_0) = f(x) - T_n(x; x_0).$$

Under standard conditions, Taylor's theorem gives a remainder representation. One common form is the (Lagrange) remainder:

$$R_n(x; x_0) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)^{n+1} \quad \text{for some } \xi \text{ between } x \text{ and } x_0.$$

This is the key qualitative message for experiments like E001:

- the factor $(x - x_0)^{n+1}$ makes the method **local** (good near x_0, potentially bad far away),
- increasing n helps most where $|x - x_0|$ is small.

7.41.3 What experiments typically visualize

In a numerical experiment, you often look at

- absolute error: $|R_n(x; x_0)|$
- relative error: $|R_n(x; x_0)|/|f(x)|$ (careful near zeros of f)

and plot them across a domain to see where the approximation is reliable.

7.41.4 Practical numerical caveats

Even when the mathematics are correct, computation can mislead:

- large $|x - x_0|$ and high n can produce huge intermediate terms,
- subtractive cancellation can reduce accuracy,
- floating-point rounding can dominate before the theoretical truncation error does.

A common "extension" experiment is to repeat the same plots using higher precision arithmetic to separate *truncation error* from *rounding error*.

7.41.5 Introductory reading

If you want a longer, *beginner-friendly* treatment (beyond this refresher), these are good starting points:

- A quick overview / definitions and examples: [Wikipedia contributors, 2025].
- A rigorous calculus textbook with a clean presentation of Taylor's theorem and remainders: [Apostol, 1991].
- A proof-oriented classic (slower, deeper): [Spivak, 2008].
- For the numerical viewpoint (truncation vs. rounding error): [Burden *et al.*, 2015].

7.41.6 References

See ../references.

[Apostol, 1991, Burden *et al.*, 2015, Spivak, 2008, Wikipedia contributors, 2025]

7.42 von Mangoldt $\Lambda(n)$ and Chebyshev functions refresher

The von Mangoldt function concentrates on prime powers and is central in analytic number theory. Chebyshev functions summarize prime distributions and are used in proofs of the Prime Number Theorem. See [Montgomery and Vaughan, 2006] and [Tenenbaum, 2015].

7.42.1 von Mangoldt function

Define

$$\Lambda(n) = \begin{cases} \log p, & \text{if } n = p^k \text{ for a prime } p \text{ and } k \geq 1, \\ 0, & \text{otherwise.} \end{cases}$$

So $\Lambda(n)$ "picks out" primes and prime powers.

7.42.2 Chebyshev functions

Two common Chebyshev functions:

- $\theta(x) = \sum_{p \leq x} \log p$
- $\psi(x) = \sum_{n \leq x} \Lambda(n)$

Since ψ sums $\log p$ over prime powers, it is smoother and often easier in analysis.

7.42.3 Why these matter

- $\psi(x) \sim x$ is essentially equivalent to the Prime Number Theorem.
- Λ is tied to the logarithmic derivative of $\zeta(s)$ (via Dirichlet series).

7.42.4 Experiment ideas

- plot $\psi(x)$ and compare to x
- plot $\theta(x)$ and compare to x
- compare contributions from primes vs. higher prime powers in $\psi(x)$

7.42.5 Experiments in this repository

- **E103** — Chebyshev (x) jumps at prime powers (Λ support).
- **E104** — Von Mangoldt Λ statistics (support, value distribution, summatory behavior).
- **E119** — (x) − x oscillations (prime-weighted deviation view).

7.43 Wieferich primes

A Wieferich prime (base 2) is a prime p such that

$$2^{p-1} \equiv 1 \pmod{p^2}.$$

Only two are currently known: 1093 and 3511. [The OEIS Foundation Inc., 2025, Wikipedia contributors, 2025]

7.43.1 Key facts

- **Origin (Fermat's Last Theorem, first case):** Wieferich proved in 1909 that if the first case of FLT fails for an odd prime exponent p, then p must be a Wieferich prime (base 2). [Wieferich, 1909]
- **Rarity:** Heuristics suggest Wieferich primes are very sparse; whether infinitely many exist is open. [Wikipedia contributors, 2025]
- **Generalization:** One can define Wieferich primes for any base a via $a^{p-1} \equiv 1 \pmod{p^2}$. [Prime Pages (UTM), 2025]

7.43.2 What to experiment with

- **Fast search:** Implement modular exponentiation (pow with mod) to test primes up to a bound and recover $1093, 3511$.
- **Near-Wieferich primes:** Measure the p-adic valuation $v_p(2^{p-1} - 1)$ and look for unusually large values.
- **Connections:** Explore overlaps with other "rare prime" phenomena (e.g., Wall–Sun–Sun, Wilson) as *tags* rather than hard coupling.

7.43.3 References

See Wieferich [1909], The OEIS Foundation Inc. [2025], Wikipedia contributors [2025].

CHAPTER

EIGHT

API REFERENCE AND BIBLIOGRAPHY

The full API reference with auto-generated documentation is available in the **HTML version** of this documentation.

For the PDF, we provide a brief overview of the main modules:

- **mathxlab.exp** — CLI helpers, seeding, logging, and report writing for experiments.
- **mathxlab.experiments** — Stable registry for enumerating and running experiments.
- **mathxlab.nt** — Arithmetic functions, Dirichlet machinery, and zeta/L-related utilities.
- **mathxlab.num** — Numerical series helpers.
- **mathxlab.plots** — Plotting helpers used across experiments and reports.
- **mathxlab.utils** — Shared utilities.
- **mathxlab.viz** — Visualization backend wrappers (Matplotlib).

BIBLIOGRAPHY

[1] Experimental mathematics (project euclid archive). URL: https://projecteuclid.org/journals/experimental-mathematics (visited on 2025-12-22).

[2] Experimental mathematics lab (university of luxembourg). URL: https://math.uni.lu/eml/ (visited on 2025-12-22).

[3] Experimental mathematics website. URL: https://www.experimentalmath.info (visited on 2025-12-22).

[4] Experimental mathematics (journal). 1992. URL: https://www.tandfonline.com/journals/uexm20.

[5] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. URL: https://annals.math.princeton.edu/2004/160-2/p12 (visited on 2025-12-27), doi:10.4007/annals.2004.160.781.

[6] L. Alaoglu and Paul Erdős. On highly composite and similar numbers. *Transactions of the American Mathematical Society*, 56(3):448–469, 1944. doi:10.1090/S0002-9947-1944-0011087-2.

[7] W. R. Alford, Andrew Granville, and Carl Pomerance. There are infinitely many carmichael numbers. *Annals of Mathematics*, 139(3):703–722, 1994. URL: https://annals.math.princeton.edu/1994/139-3/p10.

[8] George E. Andrews. *The Theory of Partitions*. Volume 2 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1984. ISBN 978-0-521-63766-4. doi:10.1017/CBO9780511608650.

[9] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer, 1976. URL: https://link.springer.com/book/10.1007/978-1-4757-5579-4 (visited on 2025-12-27), doi:10.1007/978-1-4757-5579-4.

[10] Tom M. Apostol. *Calculus, Volume 1*. John Wiley & Sons, 2 edition, 1991. ISBN 9780471000051. URL: https://books.google.com/books/about/Calculus\TU\textbackslash{}_Volume\TU\textbackslash{}_1.html?id=o2D4DwAAQBAJ.

[11] Vladimir I. Arnold. *Experimental Mathematics*. MSRI Mathematical Circles Library. American Mathematical Society, 2015. ISBN 9780821894163. URL: https://bookstore.ams.org/msri-13/ (visited on 2025-12-22).

[12] Jeremy Avigad and Rebecca Morris. The concept of “character” in dirichlet's theorem on primes in an arithmetic progression. *Archive for History of Exact Sciences*, 68(3):265–326, 2014. URL: https://arxiv.org/abs/1209.3657 (visited on 2025-12-29), doi:10.1007/s00407-013-0126-0.

[13] Christian Axler. New estimates for the nth prime number. *Journal of Integer Sequences*, 22(4):Article 19.4.2, 2019. URL: https://cs.uwaterloo.ca/journals/JIS/VOL22/Axler/axler17.pdf (visited on 2025-12-27).

[14] Christian Axler. Effective estimates for some functions defined over primes. *Integers*, 24:A34, 2024. URL: https://math.colgate.edu/~integers/y34/y34.pdf (visited on 2025-12-27).

[15] David H. Bailey and Jonathan M. Borwein. Experimental mathematics: examples, methods and implications. *Notices of the American Mathematical Society*, 52(5):502–514, 2005. URL: https://www.ams.org/notices/200505/fea-borwein.pdf.

[16] David H. Bailey, Jonathan M. Borwein, and Richard E. Crandall. Ten problems in experimental mathematics. *Experimental Mathematics*, 13(2):193–207, 2004.

[17] Alan Baker. Linear forms in the logarithms of algebraic numbers. I. *Mathematika*, 13(2):204–216, 1966. URL: https://doi.org/10.1112/S0025579300003971 (visited on 2026-01-06), doi:10.1112/S0025579300003971.

[18] Paul T. Bateman and Roger A. Horn. A heuristic asymptotic formula concerning the distribution of prime numbers. *Mathematics of Computation*, 16(79):363–367, 1962. Introduces the Bateman–Horn heuristic for primes represented by polynomial sequences. doi:10.1090/S0025-5718-1962-0148632-7.

[19] Albert H. Beiler. *Recreations in the Theory of Numbers: The Queen of Mathematics Entertains*. Dover Publications, New York, 2nd ed. edition, 1966. ISBN 0486210960. ISBN-13: 9780486210964. URL: https://www.worldcat.org/fr/title/recreations-in-the-theory-of-numbers-the-queen-of-mathematics-entertains/oclc/905454754 (visited on 2026-01-04).

[20] Bruce C. Berndt, Ronald J. Evans, and Kenneth S. Williams. *Gauss and Jacobi Sums*. Wiley-Interscience, 1998. ISBN 9780471128076.

[21] Jonathan Borwein, Alf van der Poorten, Jeffrey Shallit, and Wadim Zudilin. *Neverending Fractions: An Introduction to Continued Fractions*. Volume 23 of Australian Mathematical Society Lecture Series. Cambridge University Press, 2014. ISBN 9780521186490. URL: https://www.cambridge.org/core/books/neverending-fractions/A3900DAB483D65CE6CB960A6B71226EE (visited on 2025-12-22).

[22] Jonathan M. Borwein. The experimental mathematician: the pleasure of discovery and the role of proof. *International Journal of Computers for Mathematical Learning*, 10:75–108, 2005. doi:10.1007/s10758-005-6244-3.

[23] Jonathan M. Borwein. *The Crucible: An Introduction to Experimental Mathematics*. A K Peters, Wellesley, MA, USA, 2009. ISBN 9781568813438.

[24] Jonathan M. Borwein and David H. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A K Peters, Wellesley, MA, USA, 2 edition, 2008. ISBN 9781568814421.

[25] Jonathan M. Borwein, David H. Bailey, and Roland Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*. A K Peters, Natick, MA, USA, 2004. ISBN 9781568811369. doi:10.1201/9781439864197.

[26] Jonathan M. Borwein, David H. Bailey, Roland Girgensohn, David Luke, and Victor H. Moll. *Experimental Mathematics in Action*. A K Peters, Wellesley, MA, USA, 2007. ISBN 9781568812717. URL: https://carmamaths.org/resources/jon/Preprints/Books/EMA/ema.pdf (visited on 2025-12-22).

[27] Jonathan M. Borwein and Richard P. Brent. *Inquiries into Experimental Mathematics*. A K Peters, Wellesley, MA, USA, 2004. ISBN 9781568812113.

[28] Richard P. Brent. An improved monte carlo factorization algorithm. *BIT Numerical Mathematics*, 20(2):176–184, 1980. URL: https://doi.org/10.1007/BF01933190 (visited on 2026-01-10), doi:10.1007/BF01933190.

[29] Dirk Brockmann. Prime time: the distribution of primes along number spirals. Online, 2019. Explorable explanation including the Klauber triangle and Sacks spiral definitions. URL: https://www.complexity-explorables.org/explorables/prime-time/ (visited on 2026-01-01).

[30] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, 10 edition, 2015. ISBN 9781305253667. URL: https://www.cengage.com/c/numerical-analysis-10e-faires/9781305253667/.

[31] Chris Caldwell. Characterizing all even perfect numbers. n.d. PrimePages (The Prime Database), proof note on the Euclid–Euler characterization. URL: https://t5k.org/notes/proofs/EvenPerfect.html (visited on 2025-12-25).

[32] Chris K. Caldwell. Mersenne primes: history, theorems and lists. Online, 2021. PrimePages reference page collecting history, key theorems, and curated tables related to Mersenne primes and perfect numbers. URL: https://primes.utm.edu/mersenne/ (visited on 2025-12-27).

[33] Chris K. Caldwell. Prime gaps and related records. Online, n.d. PrimePages (The Prime Database), curated notes and links on prime gaps and record computations. URL: https://t5k.org/notes/primegaps.html (visited on 2025-12-27).

[34] R. D. Carmichael. On composite numbers p which satisfy the fermat congruence a^P-1 \equiv 1 (mod p). *American Mathematical Monthly*, 19(2):22–27, 1912. URL: https://www.jstor.org/stable/2974142.

[35] Keith Conrad. Korselt's criterion for carmichael numbers. https://kconrad.math.uconn.edu/blurbs/ugradnumthy/carmichael.pdf, 2004. Online lecture note; accessed 2025-12-28.

[36] Wikipedia contributors. Bunyakovsky conjecture. 2025. Wikipedia, revision as of 2025-12-19 (permanent link). URL: https://en.wikipedia.org/w/index.php?oldid=1328429627&title=Bunyakovsky_conjecture (visited on 2026-01-06).

[37] Wikipedia contributors. Heegner number. 2025. Wikipedia, revision as of 2025-09-17 (permanent link). URL: https://en.wikipedia.org/w/index.php?oldid=1311960098&title=Heegner_number (visited on 2026-01-06).

[38] Wikipedia contributors. Lucas–lehmer primality test — wikipedia, the free encyclopedia. Online, 2025. Revision as of 01:16, 31 October 2025. Abstract: Description and correctness of the Lucas–Lehmer test used to prove primality of Mersenne numbers $M_p=2^p-1$. URL: https://en.wikipedia.org/w/index.php?oldid=1319643028&title=Lucas%E2%80%93Lehmer_primality_test (visited on 2025-12-27).

[39] Wikipedia contributors. Lucky numbers of euler. 2025. Wikipedia, revision as of 2025-01-03 (permanent link). URL: https://en.wikipedia.org/w/index.php?oldid=1267053719\TU\textbackslash{}&title=Lucky\TU\textbackslash{}_numbers\TU\textbackslash{}_of\TU\textbackslash{}_Euler (visited on 2026-01-02).

[40] Wikipedia contributors. Mersenne prime — wikipedia, the free encyclopedia. Online, 2025. Revision as of 20:10, 11 December 2025. Abstract: Overview of Mersenne primes (numbers of the form 2^p-1), their connection to perfect numbers, and known results and records. URL: https://en.wikipedia.org/w/index.php?oldid=1326946346&title=Mersenne_prime (visited on 2025-12-27).

[41] Wikipedia contributors. Ulam spiral — wikipedia, the free encyclopedia. Online, 2025. Revision as of 16:55, 25 October 2025. Visualization of primes arranged in a spiral; discusses prime-rich lines and links to quadratic polynomials. URL: https://en.wikipedia.org/w/index.php?oldid=1318727705&title=Ulam_spiral (visited on 2026-01-01).

[42] David A. Cox. *Primes of the Form $x^2 + ny^2$: Fermat, Class Field Theory, and Complex Multiplication*. John Wiley & Sons, 2 edition, 2013. ISBN 9781118390184.

[43] Harald Cramér. On the order of magnitude of the difference between consecutive prime numbers. *Acta Arithmetica*, 2(1):23–46, 1936. URL: https://www.impan.pl/en/publishing-house/journals-and-series/acta-arithmetica/all/2/1/93277/on-the-order-of-magnitude-of-the-difference-between-consecutive-prime-numbers (visited on 2025-12-27), doi:10.4064/aa-2-1-23-46.

[44] Richard Crandall and Carl Pomerance. *Prime Numbers: A Computational Perspective*. Springer, 2005. URL: https://link.springer.com/book/10.1007/978-1-4684-9316-0 (visited on 2025-12-27), doi:10.1007/978-1-4684-9316-0.

[45] Harold Davenport. *Multiplicative Number Theory*. Volume 74 of Graduate Texts in Mathematics. Springer, New York, 3 edition, 2000. ISBN 978-0-387-95097-6. URL: https://link.springer.com/book/10.1007/978-1-4757-5927-3 (visited on 2025-12-29), doi:10.1007/978-1-4757-5927-3.

[46] Fred Diamond and Jerry Shurman. *A First Course in Modular Forms*. Volume 228 of Graduate Texts in Mathematics. Springer, 2005. ISBN 9780387232294. URL: https://doi.org/10.1007/978-0-387-27226-9 (visited on 2026-01-06), doi:10.1007/978-0-387-27226-9.

[47] Marcus du Sautoy. *The Number Mysteries: A Mathematical Odyssey through Everyday Life*. HarperCollins Publishers, may 2011. ISBN 9780007309863. Imprint: 4th Estate; ISBN-10: 0007309864. URL: https://www.harpercollins.com.au/9780007309863/the-number-mysteries-a-mathematical-odyssey-through-everyday-life/ (visited on 2026-01-04).

[48] David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley & Sons, 3 edition, 2004. ISBN 9780471452348.

[49] Pierre Dusart. Estimates of some functions over primes without R.H. arXiv preprint, 2010. URL: https://arxiv.org/abs/1002.0442 (visited on 2026-01-10), arXiv:1002.0442.

[50] Pierre Dusart. Explicit estimates of some functions over primes. *The Ramanujan Journal*, 45(1):227–251, 2018. URL: https://link.springer.com/article/10.1007/s11139-016-9839-4 (visited on 2025-12-27), doi:10.1007/s11139-016-9839-4.

[51] Harold M. Edwards. *Riemann's Zeta Function*. Princeton University Press, 1974. ISBN 9780691113852.

[52] Paul Erdős and Mark Kac. The gaussian law of errors in the theory of additive number theoretic functions. *American Journal of Mathematics*, 62:738–742, 1940. doi:10.2307/2371483.

[53] Paul Erdős, Carl Pomerance, and Eric Schmutz. Carmichael's lambda function. *Acta Arithmetica*, 58(4):363–385, 1991. URL: http://eudml.org/doc/206359.

[54] John Friedlander and Henryk Iwaniec. *Opera de Cribro*. Volume 57 of Colloquium Publications. American Mathematical Society, 2010. URL: https://bookstore.ams.org/coll-57/ (visited on 2025-12-27), doi:10.1090/coll/057.

[55] M. Gardner. *Martin Gardner's Sixth Book of Mathematical Diversions from Scientific American*. Scientific American. University of Chicago Press, 1983. ISBN 9780226282503. URL: https://books.google.ch/books?id=fUViQgAACAAJ.

[56] Andrew Granville. Pretentiousness in analytic number theory. *Journal de Théorie des Nombres de Bordeaux*, 21(1):159–173, 2009. doi:10.5802/jtnb.664.

[57] Andrew Granville and Greg Martin. Prime number races. *The American Mathematical Monthly*, 113(1):1–33, 2006. URL: https://arxiv.org/abs/math/0408319 (visited on 2025-12-29), doi:10.1080/00029890.2006.11920275.

[58] Andrew Granville and Kannan Soundararajan. Large character sums: pretentious characters and the Pólya–Vinogradov theorem. *Journal of the American Mathematical Society*, 20(2):357–384, 2007. doi:10.1090/S0894-0347-06-00549-1.

[59] Ben Green and Terence Tao. The primes contain arbitrarily long arithmetic progressions. *Annals of Mathematics*, 167(2):481–547, 2008. URL: https://annals.math.princeton.edu/2008/167-2/p01 (visited on 2025-12-27), doi:10.4007/annals.2008.167.481.

[60] Richard K. Guy. *Unsolved Problems in Number Theory*. Springer, 3 edition, 2004. ISBN 9780387208602. URL: https://link.springer.com/book/10.1007/978-0-387-26677-0, doi:10.1007/978-0-387-26677-0.

[61] Heini Halberstam and Hans-Egon Richert. *Sieve Methods*. Academic Press, 1974. URL: https://www.sciencedirect.com/book/9780123182501/sieve-methods (visited on 2025-12-27).

[62] G. H. Hardy and J. E. Littlewood. Some problems of `partitio numerorum'; III: on the expression of a number as a sum of primes. *Acta Mathematica*, 44:1–70, 1923. URL: https://link.springer.com/article/10.1007/BF02403921 (visited on 2025-12-27), doi:10.1007/BF02403921.

[63] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 6 edition, 2008. ISBN 9780199219858.

[64] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 6 edition, 2008. ISBN 9780199219865. URL: https://global.oup.com/ukhe/product/an-introduction-to-the-theory-of-numbers-9780199219865 (visited on 2025-12-27).

[65] Kurt Heegner. Diophantische analysis und modulfunktionen. *Mathematische Zeitschrift*, 56:227–253, 1952. URL: https://doi.org/10.1007/BF01174749 (visited on 2026-01-06), doi:10.1007/BF01174749.

[66] Christian Hill. The klauber triangle. Online, 2016. Blog post describing the construction and a short Python script. URL: https://scipython.com/blog/the-klauber-triangle/ (visited on 2026-01-01).

[67] P. Hoffman. *Archimedes' Revenge: The Joys and Perils of Mathematics*. Fawcett Crest, 1989. ISBN 9780449217504. URL: https://books.google.ch/books?id=_SDoAAAACAAJ.

[68] Steve Humble. *The Experimenter's A–Z of Mathematics: Math Activities with Computer Support*. Routledge, 2002. ISBN 9781853468179. URL: https://www.routledge.com/The-Experimenters-A-Z-of-Mathematics-Math-Activities-with-Computer-Support/Humble/p/book/9781853468179 (visited on 2026-01-06).

[69] OEIS Foundation Inc. A000043 — mersenne exponents. Online, 2025. OEIS entry for primes p such that 2^p-1 is prime (exponents of Mersenne primes). URL: https://oeis.org/A000043 (visited on 2025-12-27).

[70] OEIS Foundation Inc. A001348 — mersenne numbers: 2^p-1 where p is prime. Online, 2025. OEIS entry for the Mersenne numbers sequence 2^p-1 with prime exponents p. URL: https://oeis.org/A001348 (visited on 2025-12-27).

[71] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory*. Graduate Texts in Mathematics. Springer, New York, 2 edition, 1990. ISBN 978-0-387-97329-6. URL: https://link.springer.com/book/10.1007/978-1-4757-2103-4 (visited on 2025-12-29), doi:10.1007/978-1-4757-2103-4.

[72] Aleksandar Ivić. *The Riemann Zeta-Function: Theory and Applications*. John Wiley & Sons, 1985.

[73] Henryk Iwaniec and Emmanuel Kowalski. *Analytic Number Theory*. Volume 53 of Colloquium Publications. American Mathematical Society, 2004. URL: https://bookstore.ams.org/coll-53/ (visited on 2025-12-27), doi:10.1090/coll/053.

[74] G. J. O. Jameson. *The Prime Number Theorem*. Volume 53 of London Mathematical Society Student Texts. Cambridge University Press, 2003. ISBN 9780521891103. URL: https://www.cambridge.org/core/books/prime-number-theorem/42A4EE8B79DC9D9738416D7514469DD7 (visited on 2026-01-04).

[75] U. H. Kurzweg. Hexagonal-integer-spiral and primes. PDF, 11 2020. Short note proposing a hexagonal integer spiral representation and discussing primes. URL: https://web.mae.ufl.edu/uhk/HEXAGONAL.pdf (visited on 2026-01-01).

[76] Michal Kř\'ıžek, Florian Luca, and Ladislav Šomer. *17 Lectures on Fermat Numbers: From Number Theory to Geometry*. CMS Books in Mathematics. Springer, 2013. ISBN 9781461465250. doi:10.1007/978-1-4614-6526-7.

[77] Jeffrey C. Lagarias. An elementary problem equivalent to the Riemann hypothesis. *The American Mathematical Monthly*, 109(6):534–543, 2002. doi:10.1080/00029890.2002.11919832.

[78] Peter Gustav Lejeune Dirichlet. Beweis des satzes, dass jede unbegrenzte arithmetische progression, deren erstes glied und differenz ganze zahlen ohne gemeinschaftlichen faktor sind, unendlich viele primzahlen enth"alt. *Abhandlungen der K"oniglichen Preussischen Akademie der Wissenschaften zu Berlin*, pages 45–81, 1837. URL: https://sites.mathdoc.fr/cgi-bin/oeitem?id=OE_DIRICHLET__1_313_0 (visited on 2025-12-29).

[79] Shangzhi Li, Falai Chen, Jiansong Deng, Yaohua Wu, and Yunhua Zhang. *Mathematics Experiments*. World Scientific, 2003. ISBN 9789812380500. doi:10.1142/5008.

[80] Daniel A. Marcus. *Number Fields*. Universitext. Springer, Cham, 2 edition, 2018. ISBN 9783319902326. URL: https://link.springer.com/book/10.1007/978-3-319-90233-3 (visited on 2026-01-09), doi:10.1007/978-3-319-90233-3.

[81] James Maynard. Small gaps between primes. *Annals of Mathematics*, 181(1):383–413, 2015. URL: https://annals.math.princeton.edu/2015/181-1/p07 (visited on 2025-12-27), doi:10.4007/annals.2015.181.1.7.

[82] Inc. Mersenne Research. Gimps — the math — primenet. Online, 2024. Background on the mathematics and algorithms used in the GIMPS search strategy (trial factoring, P-1, PRP testing, Lucas–Lehmer, double-checking). URL: https://www.mersenne.org/various/math.php (visited on 2025-12-27).

[83] Inc. Mersenne Research. List of known mersenne prime numbers. Online, 2025. PrimeNet/GIMPS list of known Mersenne primes including discovery metadata and verification method. URL: https://www.mersenne.org/primes/ (visited on 2025-12-27).

[84] H. L. Montgomery. The pair correlation of zeros of the zeta function. *Proceedings of Symposia in Pure Mathematics*, 24:181–193, 1973.

[85] Hugh L. Montgomery and Robert C. Vaughan. *Multiplicative Number Theory I: Classical Theory*. Volume 97 of Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006. ISBN 978-0-521-84903-6. doi:10.1017/CBO9780511618314.

[86] M. Ram Murty and V. Kumar Murty. *Non-vanishing of L-Functions and Applications*. Volume 157 of Progress in Mathematics. Birkhäuser, 1997. ISBN 9783764358013. doi:10.1007/978-3-0348-8956-8.

[87] Thomas R. Nicely. New maximal prime gaps and first occurrences. *Mathematics of Computation*, 68(227):1311–1315, 1999. URL: https://www.ams.org/mcom/1999-68-227/S0025-5718-99-01065-0/ (visited on 2025-12-27), doi:10.1090/S0025-5718-99-01065-0.

[88] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. *An Introduction to the Theory of Numbers*. John Wiley & Sons, 5 edition, 1991. ISBN 978-0-471-62546-0.

[89] Pascal Ochem and Michaël Rao. Lower bounds on odd perfect numbers. 2014. Slides (Montpellier, 2014-07-02). URL: https://www.lirmm.fr/~ochem/opn/opn\TU\textbackslash{}_slide.pdf (visited on 2025-12-25).

[90] Andrew M. Odlyzko. The 10^20-th zero of the Riemann zeta function and 175 million of its neighbors. Unpublished manuscript, 1992. Revision of a 1989 manuscript; numerical computations of high zeros of $\zeta (s)$. URL: https://www-users.cse.umn.edu/~odlyzko/unpublished/zeta.10to20.1992.pdf (visited on 2025-12-30).

[91] Andrew M. Odlyzko and Herman J. J. te Riele. Disproof of the mertens conjecture. *Journal für die reine und angewandte Mathematik*, 357:138–160, 1985. URL: http://eudml.org/doc/152712, doi:10.1515/crll.1985.357.138.

[92] Marko Petkovšek, Herbert S. Wilf, and Doron Zeilberger. *A=B*. A K Peters, Wellesley, MA, USA, 1996. ISBN 9781568810635. URL: https://sites.math.rutgers.edu/~zeilberg/expmath/ (visited on 2025-12-22).

[93] Charles C. Pinter. *A Book of Abstract Algebra*. Dover Publications, 2 edition, 2010. ISBN 9780486474175. URL: https://store.doverpublications.com/products/9780486474175 (visited on 2026-01-09).

[94] J'anos Pintz. Landau's problems on primes. *Journal de Th'eorie des Nombres de Bordeaux*, 2009. Accessed 2026-01-02. URL: https://www.numdam.org/item/10.5802/jtnb.676.pdf.

[95] John M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975. URL: https://doi.org/10.1007/BF01933667 (visited on 2026-01-10), doi:10.1007/BF01933667.

[96] George Pólya. *Mathematics and Plausible Reasoning, Volume I: Induction and Analogy in Mathematics*. Princeton University Press, Princeton, NJ, USA, 1954.

[97] George Pólya. *Mathematics and Plausible Reasoning, Volume II: Patterns of Plausible Inference*. Princeton University Press, Princeton, NJ, USA, 1954.

[98] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980. URL: https://www.sciencedirect.com/science/article/pii/0022314X80900840 (visited on 2025-12-27), doi:10.1016/0022-314X(80)90084-0.

[99] Srinivasa Ramanujan. Highly composite numbers. *Proceedings of the London Mathematical Society*, s2-14(1):347–409, 1915. URL: https://ramanujan.sirinudi.org/Volumes/published/ram15.pdf, doi:10.1112/plms/s2_14.1.347.

[100] Paulo Ribenboim. *The New Book of Prime Number Records*. Springer, 1996. ISBN 9780387944579. URL: https://link.springer.com/book/10.1007/978-1-4612-0759-7 (visited on 2025-12-27), doi:10.1007/978-1-4612-0759-7.

[101] Hans Riesel. *Prime Numbers and Computer Methods for Factorization*. Birkhäuser, 2 edition, 1994. URL: https://link.springer.com/book/10.1007/978-0-8176-8298-9 (visited on 2025-12-27), doi:10.1007/978-0-8176-8298-9.

[102] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.

[103] Guy Robin. Grandes valeurs de la fonction somme des diviseurs et hypothèse de Riemann. *Journal de Mathématiques Pures et Appliquées*, 63:187–213, 1984.

[104] J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64–94, 1962. URL: https://projecteuclid.org/journals/illinois-journal-of-mathematics/volume-6/issue-1/Approximate-Formulas-for-Some-Functions-of-Prime-Numbers/ijm/1255631807.full (visited on 2025-12-27), doi:10.1215/ijm/1255631807.

[105] Michael Rubinstein and Peter Sarnak. Chebyshev's bias. *Experimental Mathematics*, 3(3):173–197, 1994. URL: https://projecteuclid.org/journals/experimental-mathematics/volume-3/issue-3/Chebyshevs-bias/em/1048515870.full (visited on 2025-12-29), doi:10.1080/10586458.1994.10504289.

[106] Lowell Schoenfeld. Sharper bounds for the Chebyshev functions $\theta (x)$ and $\psi (x)$. II. *Mathematics of Computation*, 30(134):337–360, 1976. doi:10.1090/S0025-5718-1976-0457374-6.

[107] Jean-Pierre Serre. *A Course in Arithmetic*. Graduate Texts in Mathematics. Springer, New York, 1973. ISBN 978-0-387-90040-7. URL: https://link.springer.com/book/10.1007/978-1-4684-9884-4 (visited on 2025-12-29), doi:10.1007/978-1-4684-9884-4.

[108] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2 edition, 2009. ISBN 9780521516440. URL: https://www.shoup.net/ntb/ (visited on 2026-01-09), doi:10.1017/CBO9780511814549.

[109] Joseph H. Silverman. *A Friendly Introduction to Number Theory*. Pearson, 4 edition, 2012. ISBN 9780321816191. URL: https://www.math.brown.edu/johsilve/frint.html (visited on 2026-01-09).

[110] Robert M. Solovay and Volker Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977. URL: https://epubs.siam.org/doi/10.1137/0206006 (visited on 2025-12-27), doi:10.1137/0206006.

[111] Michael Spivak. *Calculus*. Publish or Perish, Inc., 4 edition, 2008. ISBN 9780914098911. URL: https://www.amazon.com/Calculus-4th-Michael-Spivak/dp/0914098918.

[112] Harold M. Stark. A complete determination of the complex quadratic fields of class-number one. *Michigan Mathematical Journal*, 14(1):1–27, 1967. URL: https://doi.org/10.1307/mmj/1028999653 (visited on 2026-01-06), doi:10.1307/mmj/1028999653.

[113] Harold M. Stark. On the "gap" in a theorem of heegner. *Journal of Number Theory*, 1(1):16–27, 1969. URL: https://doi.org/10.1016/0022-314X(69)90023-7 (visited on 2026-01-06), doi:10.1016/0022-314X(69)90023-7.

[114] William Stein. *Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach*. Undergraduate Texts in Mathematics. Springer, New York, NY, nov 2010.

ISBN 9781441927521. ISBN-10: 1441927522; Softcover published 19 Nov 2010. URL: https://link.springer.com/book/10.1007/b13279 (visited on 2026-01-04), doi:10.1007/b13279.

[115] Andrew Stone. Improved upper bounds for odd perfect numbers — part i. *Integers: Electronic Journal of Combinatorial Number Theory*, 2024. URL: https://math.colgate.edu/~integers/y114/y114.pdf.

[116] Gérald Tenenbaum. *Introduction to Analytic and Probabilistic Number Theory*. Volume 163 of Graduate Studies in Mathematics. American Mathematical Society, 3 edition, 2015. ISBN 978-1-4704-7821-6.

[117] E. C. Titchmarsh. *The Theory of the Riemann Zeta-Function*. Oxford University Press, 2 edition, 1986. ISBN 9780198533696.

[118] John Voight. Perfect numbers: an elementary introduction. 1998. Lecture notes / survey. Date: May 31, 1998; updated January 27, 2024. URL: https://jvoight.github.io/notes/perfelem-051015.pdf (visited on 2025-12-25).

[119] Lawrence C. Washington. *Introduction to Cyclotomic Fields*. Volume 83 of Graduate Texts in Mathematics. Springer, New York, 2 edition, 1997. ISBN 978-0-387-94762-4. URL: https://link.springer.com/book/10.1007/978-1-4612-1934-7 (visited on 2025-12-29), doi:10.1007/978-1-4612-1934-7.

[120] Eric W. Weisstein. Perfect number. 2003. MathWorld—A Wolfram Web Resource. URL: https://mathworld.wolfram.com/PerfectNumber.html (visited on 2025-12-25).

[121] Eric W. Weisstein. Euler prime. 2025. MathWorld—A Wolfram Web Resource. URL: https://mathworld.wolfram.com/EulerPrime.html (visited on 2026-01-02).

[122] Eric W. Weisstein. Heegner number. 2025. MathWorld—A Wolfram Web Resource. URL: https://mathworld.wolfram.com/HeegnerNumber.html (visited on 2026-01-06).

[123] Eric W. Weisstein. Lucky number of Euler. 2025. MathWorld—A Wolfram Web Resource. URL: https://mathworld.wolfram.com/LuckyNumberofEuler.html (visited on 2026-01-06).

[124] Arthur Wieferich. Zum letzten fermatschen theorem. *Journal für die reine und angewandte Mathematik*, 136:293–302, 1909.

[125] George Woltman and Scott Kurowski. On the discovery of the 45th and 46th known mersenne primes. *The Fibonacci Quarterly*, 46/47(3):194–197, 2009. Abstract PDF. Describes GIMPS methods and reports the discoveries of M37156667 and M43112609. URL: https://www.fq.math.ca/Abstracts/46_47-3/woltman.pdf (visited on 2025-12-27).

[126] Yitang Zhang. Bounded gaps between primes. *Annals of Mathematics*, 179(3):1121–1174, 2014. URL: https://annals.math.princeton.edu/2014/179-3/p07 (visited on 2025-12-27), doi:10.4007/annals.2014.179.3.7.

[127] CyPari2 Developers. Cypari2 documentation: python interface to the pari library. 2026. URL: https://cypari2.readthedocs.io/ (visited on 2026-01-09).

[128] D. H. J. Polymath. Variants of the selberg sieve, and bounded intervals containing many primes. *Research in the Mathematical Sciences*, 1:12, 2014. URL: https://link.springer.com/article/10.1186/s40687-014-0019-3 (visited on 2025-12-27), doi:10.1186/s40687-014-0019-3.

[129] Mersenne Research, Inc. (GIMPS). Mersenne prime discovery: 2^136279841-1 is prime! 2024. GIMPS press release page (52nd known Mersenne prime). URL: https://www.mersenne.org/primes/?press=M136279841 (visited on 2025-12-25).

[130] Mersenne Research, Inc. (GIMPS). Gimps milestones report. 2025. URL: https://www.mersenne.org/report\TU\textbackslash{}_milestones/ (visited on 2025-12-25).

[131] MIT OpenCourseWare. 18.786 number theory ii: class field theory (spring 2016). 2016. Instructor: Dr. Sam Raskin. Lecture notes courtesy of Oron Propp; problem sets included. URL: https://ocw.mit.edu/courses/18-786-number-theory-ii-class-field-theory-spring-2016/ (visited on 2026-01-09).

[132] MIT OpenCourseWare. 18.785 number theory i (fall 2021). 2021. Instructor: Dr. Andrew Sutherland. Includes full lecture notes and problem sets. URL: https://ocw.mit.edu/courses/18-785-number-theory-i-fall-2021/ (visited on 2026-01-09).

[133] OEIS Foundation Inc. A000396: perfect numbers. 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A000396 (visited on 2025-12-25).

[134] OEIS Foundation Inc. A001097: twin primes. Online, 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A001097 (visited on 2025-12-27).

[135] OEIS Foundation Inc. A001223: prime gaps p_n+1-p_n. Online, 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A001223 (visited on 2025-12-27).

[136] OEIS Foundation Inc. A001359: lesser of twin primes. Online, 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A001359 (visited on 2025-12-27).

[137] OEIS Foundation Inc. A023201: primes p such that $p+6$ is also prime (sexy primes). Online, 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A023201 (visited on 2025-12-27).

[138] OEIS Foundation Inc. A046132: primes p such that $p-4$ is also prime (cousin prime pairs). Online, 2025. The On-Line Encyclopedia of Integer Sequences (OEIS). URL: https://oeis.org/A046132 (visited on 2025-12-27).

[139] Prime Pages (UTM). Carmichael number (prime glossary). https://t5k.org/glossary/page.php?sort=CarmichaelNumber, 2025. Online; accessed 2025-12-28.

[140] Prime Pages (UTM). Primorial (prime glossary). https://t5k.org/glossary/page.php?sort=Primorial, 2025. Online; accessed 2025-12-28.

[141] Prime Pages (UTM). Wieferich prime (prime glossary). https://t5k.org/glossary/page.php?sort=WieferichPrime, 2025. Online; accessed 2025-12-28.

[142] SageMath Developers. Sagemath tutorial (sage documentation). 2025. Release 10.8. URL: https://doc.sagemath.org/pdf/en/tutorial/sage_tutorial.pdf (visited on 2026-01-09).

[143] SymPy Developers. Sympy: number theory (sympy.ntheory) documentation. 2026. URL: https://docs.sympy.org/latest/modules/ntheory.html (visited on 2026-01-09).

[144] The OEIS Foundation Inc. A000215: Fermat numbers F(n) = 2^(2^n) + 1. https://oeis.org/A000215, 2025. Online; accessed 2025-12-28.

[145] The OEIS Foundation Inc. A001220: Wieferich primes (base 2). https://oeis.org/A001220, 2025. Online; accessed 2025-12-28.

[146] The OEIS Foundation Inc. A001358: Semiprimes. https://oeis.org/A001358, 2025. Online; accessed 2025-12-28.

[147] The OEIS Foundation Inc. A002110: Primorial numbers n#. https://oeis.org/A002110, 2025. Online; accessed 2025-12-28.

[148] The OEIS Foundation Inc. A002997: Carmichael numbers. https://oeis.org/A002997, 2025. Online; accessed 2025-12-28.

[149] The PARI Group. Pari/gp documentation. 2026. URL: https://pari.math.u-bordeaux.fr/doc.html (visited on 2026-01-09).

[150] Wikipedia contributors. Carmichael number — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?oldid=1328580842&title=Carmichael_number, 2025. Online; accessed 2025-12-28.

[151] Wikipedia contributors. Fermat number — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?oldid=1329504472&title=Fermat_number, 2025. Online; accessed 2025-12-28.

[152] Wikipedia contributors. Harmonic number. Wikipedia, 2025. Permanent revision as of 20:03, 12 December 2025 (UTC). URL: https://en.wikipedia.org/w/index.php?oldid=1327129204\TU\textbackslash{}&title=Harmonic\TU\textbackslash{}_number (visited on 2025-12-25).

[153] Wikipedia contributors. Perfect number. Wikipedia, 2025. Permanent revision as of 13:46, 22 December 2025 (UTC). URL: https://en.wikipedia.org/w/index.php?oldid=1328905011\TU\textbackslash{}&title=Perfect\TU\textbackslash{}_number (visited on 2025-12-25).

[154] Wikipedia contributors. Prime number. Wikipedia, 2025. Permanent revision as of 20:47, 2 December 2025 (UTC). URL: https://en.wikipedia.org/w/index.php?oldid=1325385945\TU\textbackslash{}&title=Prime\TU\textbackslash{}_number (visited on 2025-12-25).

[155] Wikipedia contributors. Prime-counting function — Wikipedia, the free encyclopedia. 2025. Accessed: 2025-12-29. URL: https://en.wikipedia.org/w/index.php?title=Prime-counting_function&oldid=1328904428 (visited on 2025-12-29).

[156] Wikipedia contributors. Primorial — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?oldid=1328293471&title=Primorial, 2025. Online; accessed 2025-12-28.

[157] Wikipedia contributors. Riemann zeta function — Wikipedia, the free encyclopedia. 2025. Accessed: 2025-12-29. URL: https://en.wikipedia.org/w/index.php?title=Riemann_zeta_function&oldid=1326434355 (visited on 2025-12-29).

[158] Wikipedia contributors. Riemann–von mangoldt formula — Wikipedia, the free encyclopedia. 2025. Accessed: 2025-12-29. URL: https://en.wikipedia.org/w/index.php?title=Riemann%E2%80%93von_Mangoldt_formula&oldid=1322838097 (visited on 2025-12-29).

[159] Wikipedia contributors. Taylor series. Wikipedia, 2025. Permanent revision as of 04:10, 24 December 2025 (UTC). URL: https://en.wikipedia.org/w/index.php?oldid=1329166943\TU\textbackslash{}&title=Taylor\TU\textbackslash{}_series (visited on 2025-12-25).

[160] Wikipedia contributors. Wieferich prime — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?oldid=1329186350&title=Wieferich_prime, 2025. Online; accessed 2025-12-28.

[161] Wikipedia contributors. Z function — Wikipedia, the free encyclopedia. 2025. Accessed: 2025-12-29. URL: https://en.wikipedia.org/w/index.php?title=Z_function&oldid=1313086592 (visited on 2025-12-29).

[162] Wikipedia contributors. Landau's problems. 2026. Accessed 2026-01-02. URL: https://en.wikipedia.org/wiki/Landau%27s_problems.

[163] Wikipedia contributors. Quadratic equation. 2026. Accessed 2026-01-02. URL: https://en.wikipedia.org/wiki/Quadratic_equation.