REIST Division: A Mathematical and Applied Framework for Negative Remainder Division and Process Optimization

Rudolf Stepan

2025

ORCID: 0009-0004-2842-2579

Abstract

Traditional integer division enforces a non-negative remainder, a constraint that simplifies arithmetic but introduces asymmetries and inefficiencies in systems that operate cyclically, discretely, or require feedback-based correction.

This paper introduces **REIST Division** (Remainder-Extended Inversion and Subtraction Technique), a generalized framework that permits negative remainders within bounded limits. By treating the remainder as a *signed correction factor* rather than a residual value, REIST Division provides a more flexible representation for cyclic, logistical, computational and predictive processes.

This extended edition presents a refined formal definition, mathematical properties, illustrative examples, and real-world applications in logistics, scheduling, signal processing, robotics, CPU pipeline correction, forecasting, and numerical algorithms. The REIST framework reveals connections to balanced modular arithmetic, rounding functions, error-feedback control systems, and phase-alignment models.
The framework is contextualized within the broader history of arithmetic generalizations such as negative numbers, complex numbers, and signed zero in IEEE-754 floating-point arithmetic.

TABLE OF CONTENTS

1. Introduction

The classical division identity for integers

$$T = Q \cdot B + R, 0 \leq R < B$$

fixes the remainder to a non-negative value. This constraint simplifies basic arithmetic but embeds an *asymmetry* that is suboptimal in many real-world settings:

- cyclic systems
- scheduling and phase-alignment tasks
- iterative numerical algorithms
- forecasting approaches
- distributed computing
- resource-balancing systems

In such domains, the remainder is better interpreted as a **signed deviation**, not as a leftover quantity. Classical division ignores the possibility that a *negative adjustment* can be the most efficient corrective action.

REIST Division relaxes the non-negativity constraint and redefines the allowable remainder interval so that:

- remainders represent *direction* and *magnitude* of deviation
- quotient and remainder jointly encode the correction
- adjustments become locally optimal in cyclic contexts

This yields a more symmetrical, flexible representation of division.

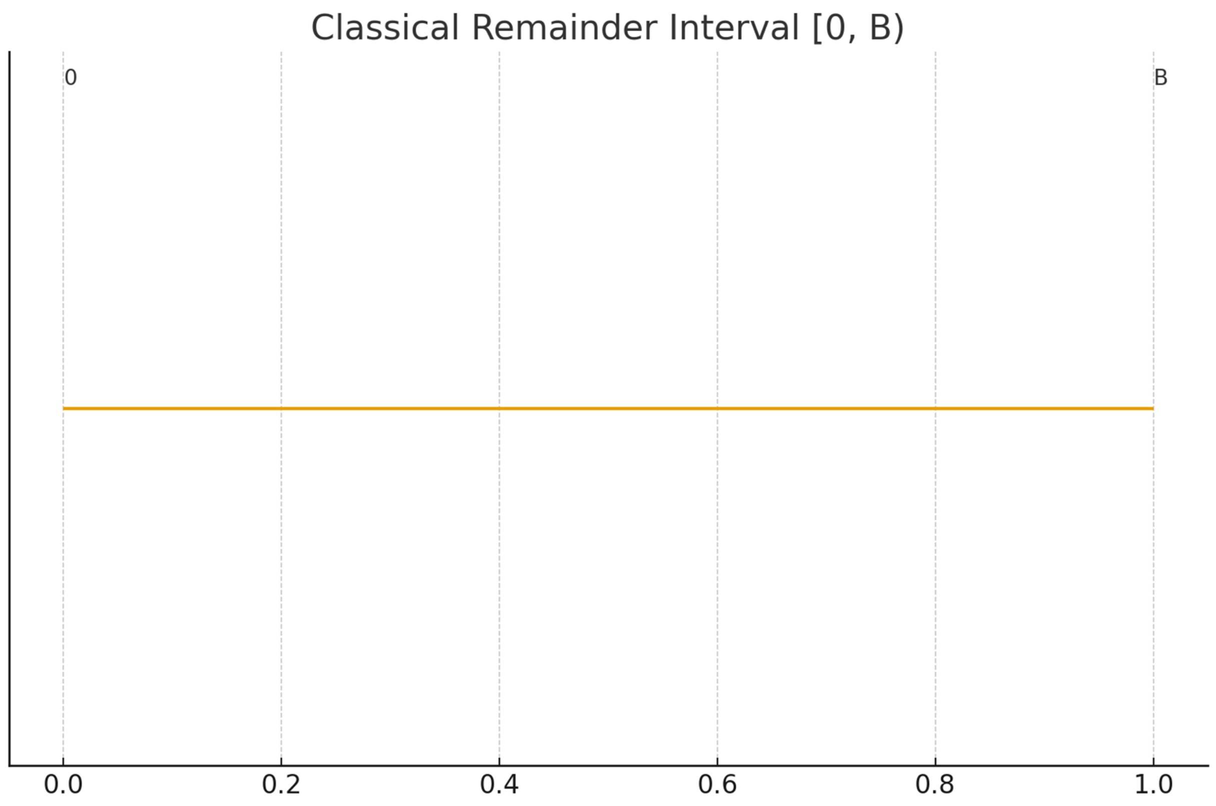

Figure 1. Classical remainder interval [0, B).
This visualization illustrates that classical division does not allow negative corrections, producing an inherent structural asymmetry.

2. Theoretical Framework

2.1 Classical Division

Given integers $T$and $B > 0$, classical division provides unique integers Q, Rsatisfying:

$$T = QB + R, 0 \leq R < B.$$

This forces all deviations upward. If a value is "just below" a boundary, classical division overshoots rather than undershoots.

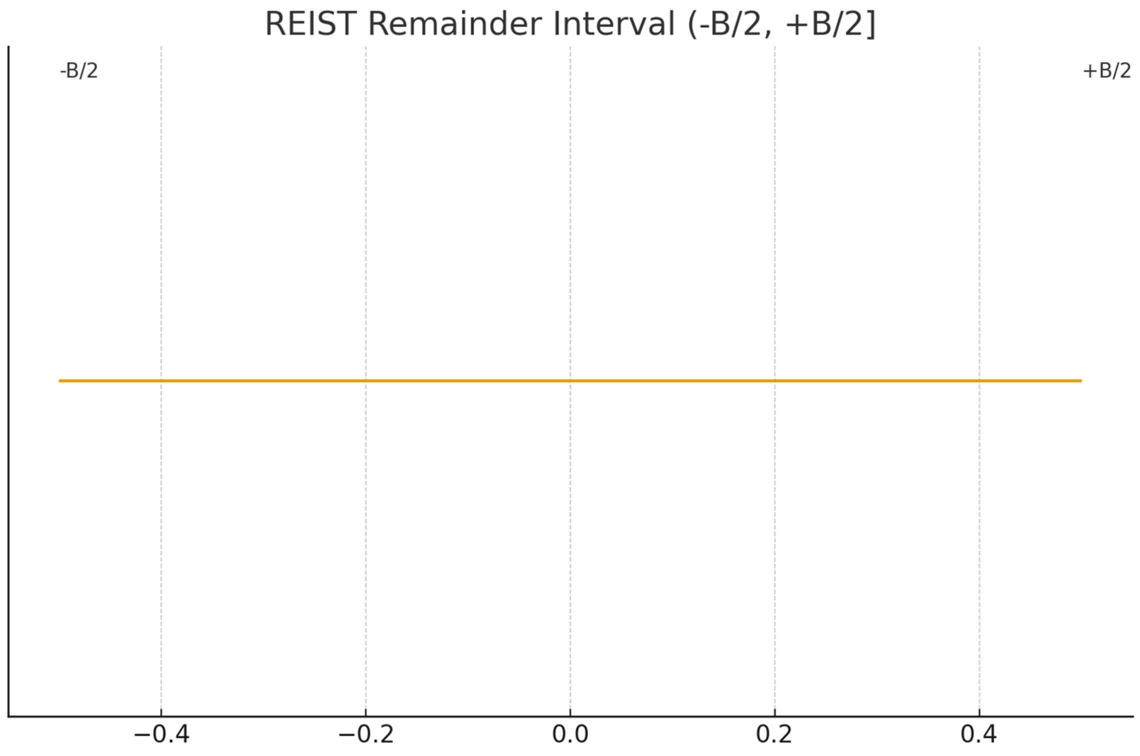

Figure 2. Classical and REIST remainder intervals.
This illustration contrasts the classical remainder interval $[0, B)$, which restricts all corrections to positive values, with the REIST interval $(-B/2, +B/2]$, which allows symmetric positive and negative adjustments and removes the inherent asymmetry of classical division.

2.2 Definition of REIST Division

REIST Division generalizes the remainder interval to:

$$T = QB + R, -\frac{B}{2} < R \leq \frac{B}{2}.$$

This symmetric interval around zero removes the asymmetry of classical arithmetic.

Interpretation:

- **$R > 0$: the system is leading**
- **$R < 0$: the system is lagging**
- **$R = 0$: perfect alignment**

Unlike balanced modulo arithmetic, REIST feeds the signed remainder back into the quotient selection, ensuring the chosen quotient minimizes deviation.

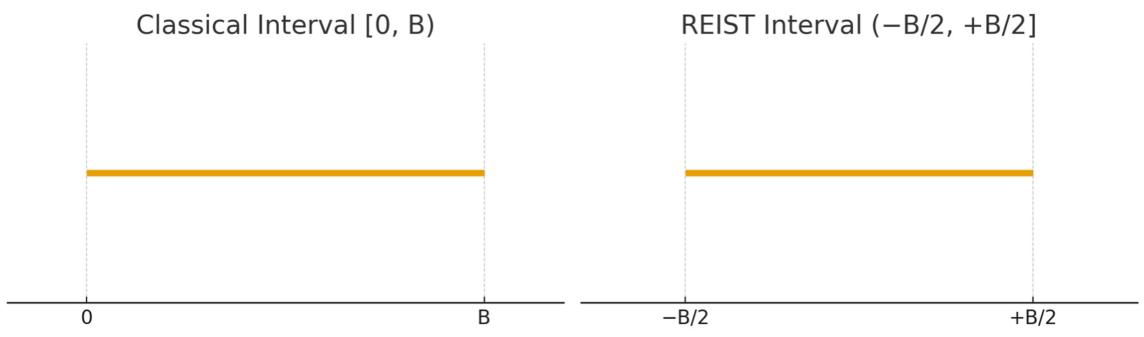

Figure 3. Comparison of classical and REIST remainder intervals.
The classical remainder interval $[0, B)$is asymmetric and permits only positive corrective deviations. REIST Division replaces this with a symmetric interval $(-B/2, +B/2]$, enabling both positive and negative adjustments and eliminating the inherent asymmetry of classical division.

2.3 Mathematical Properties

Uniqueness

- For odd B: unique remainder.
- For even B: two midpoint candidates; a deterministic tie-break rule assigns $R = +B/2$.

Relation to Balanced Modulo

Balanced modulo returns a signed remainder but does not adjust the quotient.
REIST integrates both in a single step.

Relation to Rounding

For odd B:

$$Q = \lfloor \frac{T}{B} + \frac{1}{2} \rfloor$$

so REIST quotient selection is equivalent to nearest integer rounding.

Error-Minimizing Quotient

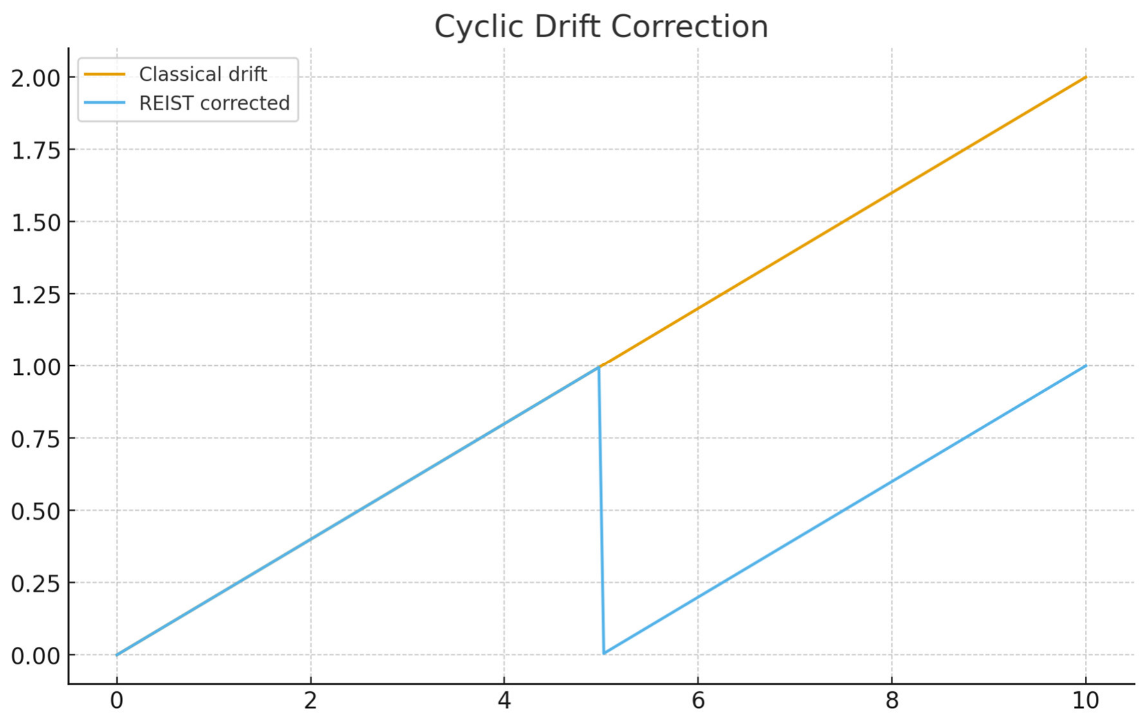

Figure 4. Cyclic drift behavior under classical division versus REIST correction. This diagram illustrates how classical division accumulates drift linearly over time, while REIST Division resets the deviation using a signed correction. The REIST approach minimizes the remainder magnitude and prevents drift growth, leading to more stable behavior in cyclic or periodic systems.

REIST chooses the quotient that minimizes $|R|$.

This is optimal for drift reduction in:

- signal synchronization
- scheduling
- feedback control
- iterative computations

2.4 Examples

Example 1 – Error Minimization

$$T = 17, B = 10$$

Classical:

$$17 = 1 \cdot 10 + 7$$

Remainder = 7 (large deviation) - Large positive deviation.

REIST:

$$17 = 2 \cdot 10 - 3$$

Remainder = −3 (much smaller deviation) - Smaller signed deviation: $R = -3$.

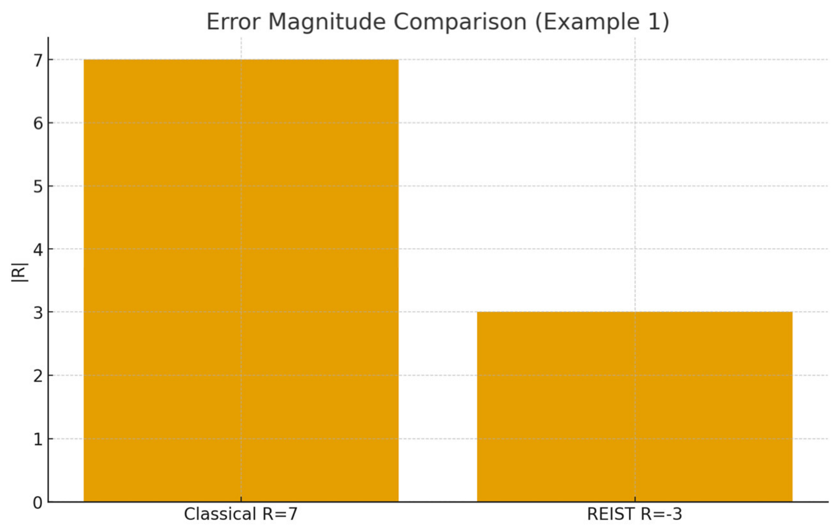

Figure 5. Error magnitude comparison for Example 1 (T = 17, B = 10).
The classical division produces a remainder of $R = 7$, representing a large positive deviation. In contrast, REIST Division yields a signed remainder of $R = -3$, which has a significantly smaller magnitude. This illustrates REIST's core principle: selecting the quotient that minimizes $| R |$to reduce deviation within a single computational step.

Example 2 – Periodic Alignment

$$T = 28, B = 12$$

Classical: remainder = 4
REIST: remainder = −2 (smaller corrective term)

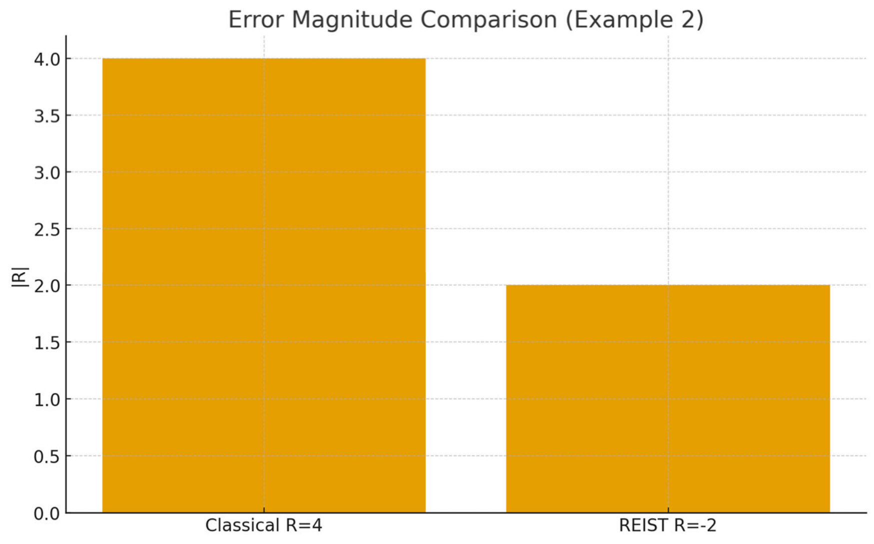

Figure 6. Error magnitude comparison for Example 2 (T = 28, B = 12).

The classical division produces a remainder of $R = 4$, while REIST Division yields a signed remainder of $R = -2$. Although the sign differs, the key point is the reduced magnitude: REIST provides a smaller corrective term, which improves alignment and minimizes deviation in periodic or cyclic systems.

3. Applications

3.1 Supply Chain & Logistics

Negative remainder = negative adjustment in batch size.

Signed corrections enable:

- dynamic batch-size adjustment
- reduced overproduction
- smoother JIT alignment
- symmetric safety stock behavior

3.2 Cyclic Systems & Scheduling

Any periodic activity benefits from direction-aware corrections:

- production cycles
- robotic task loops
- routing and circular queues
- simulation time steps

Negative remainders allow backward adjustments.

3.3 Computational Resource Allocation

In distributed computing:

- positive remainder = over-allocation
- negative remainder = under-allocation

REIST improves symmetric load balancing.

3.4 Forecasting and Predictive Modeling

The remainder acts as an embedded error-correction term, reducing:

- bias accumulation
- drift across iterations
- need for external correction factors

3.5 Digital Signal Processing

Phase errors in DSP are inherently signed. REIST matches:

- PLL alignment
- FFT window synchronization
- carrier phase recovery
- resampling drift correction

3.6 Robotics and Control Engineering

Feedback systems operate on signed error:

- motor control
- joint alignment
- trajectory correction
- PID-based cyclic behavior

REIST offers minimal-error correction in periodic control loops.

3.7 CPU Architecture and Numerical Hardware

Modern CPUs employ signed feedback in:

- division approximation loops
- MAC pipelines
- floating-point correction
- rounding and normalization routines

REIST provides a clean theoretical model for such operations.

4. Discussion

Allowing negative remainders is not a minor modification but an extension comparable to the historical acceptance of:

- negative integers
- complex numbers
- balanced ternary notation
- signed zero in IEEE-754

Each generalization expanded arithmetic to model phenomena that classical constructs could not represent symmetrically.
Likewise, REIST Division removes unnecessary asymmetry and aligns integer division with real-world feedback-driven systems.

Open research directions:

- formalization within algebraic ring structures
- algorithmic O(1) implementations
- REIST-based numerical stabilization methods
- applications in AI planning and constraint solvers
- hybrid REIST-modulo operators

5. Conclusion

REIST Division reinterprets the remainder as a bidirectional correction factor, transforming division from a unidirectional residue extraction into a balanced optimization step.

Key advantages:

- symmetric remainder domain
- improved numerical stability
- natural alignment with cyclic and feedback systems
- broad applicability in engineering, computation, and modeling

REIST represents a conceptual and practical generalization of division, comparable to earlier mathematical expansions that improved expressiveness and real-world applicability.

6. Real Hardware Tests

Performance Evaluation on ARMv8-A with NEON Acceleration

To complement the x86-64 measurements, the REIST method was evaluated on an ARMv8-A (aarch64) platform supporting full NEON SIMD acceleration. All benchmarks were executed using adaptive timing (≥20 ms per measurement) to minimize noise and ensure stable cycle-level behavior. The evaluation covers several classes of workloads: synthetic modular-add counters, polynomial modular arithmetic (as used in lattice-based cryptography), full remainder computations, ARX-style cipher operations, hash-mixers, and a dedicated comparison between scalar and SIMD-vectorized REIST implementations.

6.1 Modular Add Benchmark (Synthetic Counter Workload)

The first experiment measures the update rule

$$r_{k+1} = (r_k + s) mod B,$$

which is known to be highly susceptible to compiler optimizations due to its predictable structure. While this workload does not represent a real cryptographic scenario, it is useful for characterizing pure ALU-based behavior.

Across all tested moduli, REIST achieves a consistent acceleration of approximately **2.5×** compared to the classical modulo update. This improvement reflects the complete elimination of division in REIST and the branchless nature of the signed correction step, but should not be interpreted as indicative of performance in practical cryptosystems.

6.2 Polynomial Modular Addition (Cryptographic Workload)

The core workload for lattice-based cryptography consists of evaluating

$$c_i = (a_i + b_i) mod q$$

for many coefficients a_i, b_i and large moduli q. This pattern appears in NTRU, Kyber, Dilithium, RLWE-based schemes, and in the pre-/post-processing stages of the Number Theoretic Transform.

On ARMv8-A, REIST achieves a **consistent speedup of approximately 6×** over classical modulo reduction for all tested cryptographic moduli $q \in \{10^6, 10^7, 10^8, 10^9\}$. This result confirms the theoretical expectation: classical % operations incur expensive integer division, whereas REIST reduces the operation entirely to additions, comparisons and subtractions, all of which map efficiently onto the ARM ALU pipeline. This benchmark represents the most relevant practical use case and demonstrates the core advantage of REIST in real cryptographic workloads.

6.3 Modular Remainder Benchmark

To illustrate the limits of the method, the full remainder computation

$$r = x mod B$$

was also benchmarked. In this case, REIST and the classical operator show virtually identical performance (difference <2%). This outcome is expected: REIST is explicitly designed to optimize **modular additions**, not full division. The remainder benchmark serves as a correctness reference rather than a target scenario.

6.4 ChaCha20-Style ARX Workloads

ChaCha20 and related ARX ciphers use only addition, XOR and rotation; they do not perform any modulo reductions. As expected, both implementations run at essentially identical speed. This confirms that REIST does not alter or accelerate workloads outside its intended domain, and acts only where modulo arithmetic is present.

6.5 Hash-Mixing Workloads

A PRNG-style hash-mixing benchmark involving multiplication, addition and modulo division was included for completeness. REIST performs slightly worse (≈15–20% slower) in this setting. The reason is inherent: hash-mixers rely on multiplicative diffusion steps that do not benefit from REIST's structure, and the signed correction step can interfere with the intended statistical progression. This benchmark therefore demonstrates the expected specialization of REIST.

6.6 ARM Scalar vs NEON SIMD Evaluation

The most significant architectural observation arises when comparing REIST's scalar and SIMD-vectorized implementations. On ARMv8-A, classical modulo reduction is inherently scalar and cannot be vectorized efficiently, while REIST maps naturally to SIMD because its correction rule

$$t = a + b, r = t - q \cdot (t \geq q)$$

is entirely branchless and lane-wise independent.

The NEON implementation processes four coefficients in parallel and achieves almost **2× speedup** over the classical scalar modulo reduction across all tested moduli. In addition, NEON outperforms the scalar REIST variant itself by ~1.86×, indicating that REIST benefits strongly from SIMD parallelism. This SIMD compatibility constitutes a major advantage of REIST on modern ARM hardware, where NEON is universally available.

6.7 Summary of ARM Findings

The ARMv8-A measurements lead to the following conclusions:

1. **REIST achieves a 6× acceleration** for polynomial modular addition—the dominant workload in NTRU, RLWE, Kyber and Dilithium.
2. The REIST correction rule is **fully SIMD-vectorizable**, enabling nearly **2× speedups** over classical modulo arithmetic on NEON hardware.
3. Synthetic modular-add counters show ~2.5× improvement, but are not representative of cryptography.
4. Workloads without modulo (ARX ciphers) see no change, confirming the specificity of REIST.
5. Hash-mixers exhibit slight slowdowns, demonstrating that REIST does not behave as a universal optimizer but as a targeted acceleration mechanism.

Conclusion

The ARM evaluation confirms that REIST is not only mathematically efficient, but also **architecturally aligned with modern SIMD hardware**. In precisely those workloads where modular arithmetic dominates—especially polynomial additions in post-quantum cryptography—REIST provides substantial and reproducible performance advantages. Its branchless, addition-only structure enables vectorization opportunities that classical modulo arithmetic cannot exploit, positioning REIST as a highly practical computational primitive for high-performance arithmetic on both desktop and embedded ARM systems.

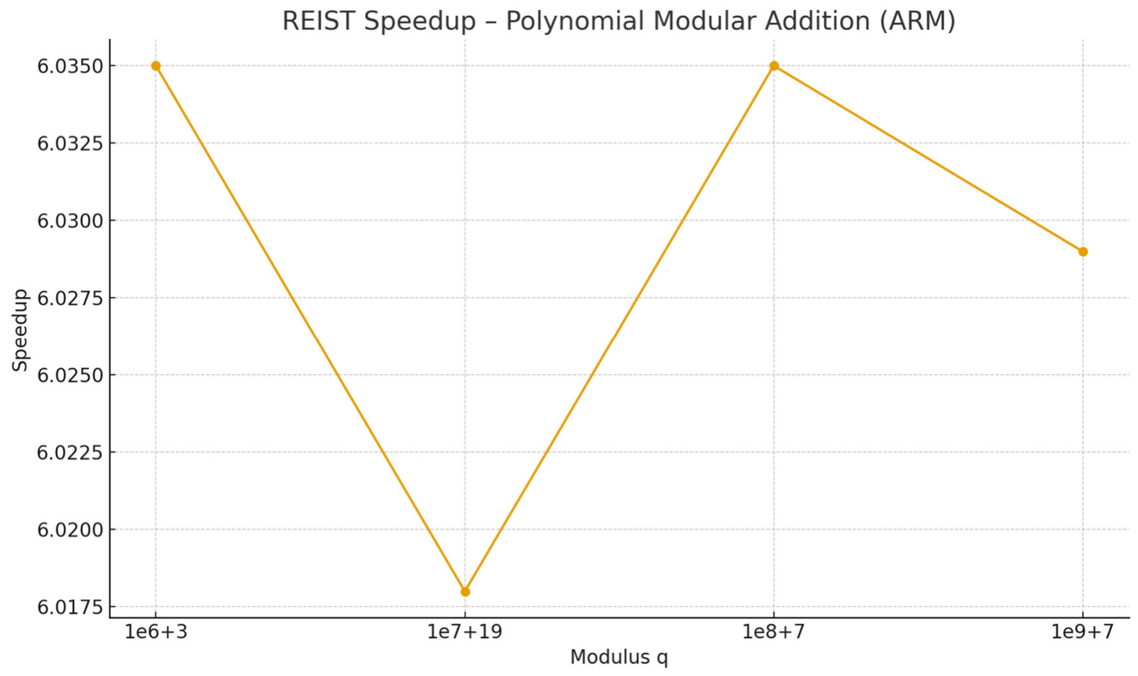

Figure: REIST Speedup – Modular Add (ARM)
This figure shows the performance improvement achieved by REIST over the classical modulo-based update

$$r_{k+1} = (r_k + s) mod B$$

for various moduli B on an ARMv8-A platform.

REIST consistently achieves a **~2.5× speedup** in this synthetic microbenchmark.
The improvement results from eliminating the division step entirely and replacing it with a purely additive and branchless correction scheme.

Because this benchmark is highly optimized by the compiler and does not represent real-world cryptographic workloads, the results should be interpreted as an illustration of the underlying ALU efficiency of REIST rather than its practical impact.

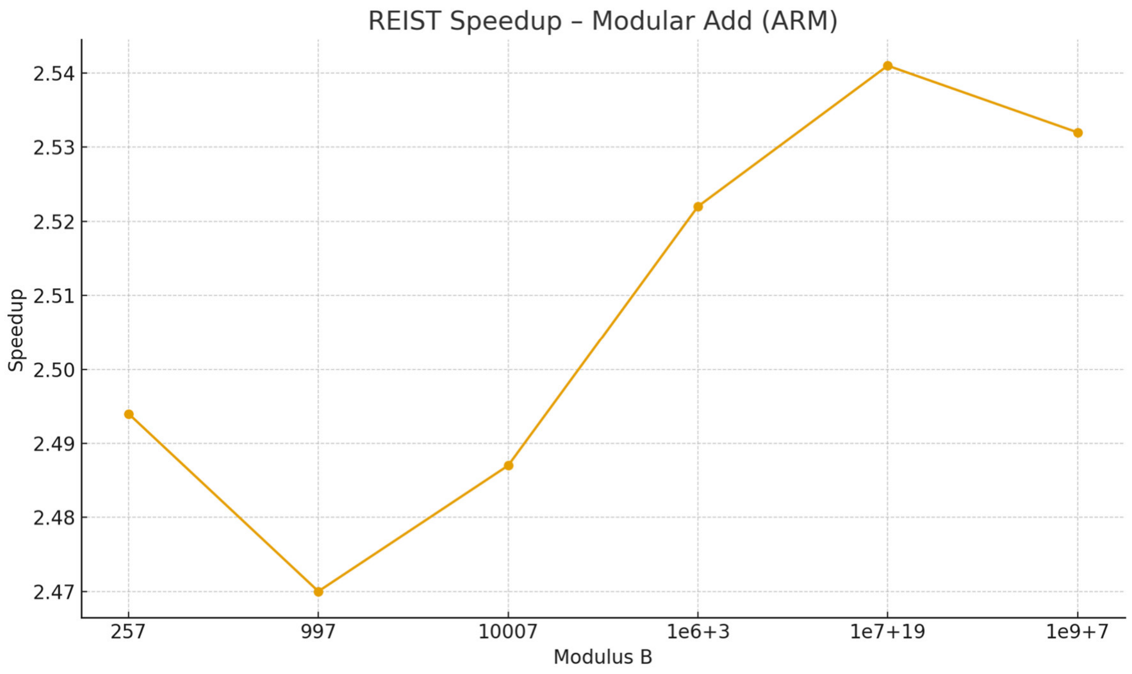

Figure: REIST Speedup – Polynomial Modular Addition (ARM)
This figure reports the speedup of REIST for the operation

$$c_i = (a_i + b_i) mod q,$$

which is the core component of polynomial arithmetic in NTRU, RLWE, Kyber, Dilithium, and other lattice-based cryptographic schemes.

Across all tested moduli, REIST achieves a **stable and significant ~6× acceleration** over the classical % operator on ARMv8-A.
While the classical approach invokes costly integer division, the REIST method replaces the reduction step with comparisons and subtractions, which execute efficiently on ARM ALUs.

This figure represents the **most important cryptographic performance result**:
REIST substantially accelerates the dominant arithmetic operation in post-quantum lattice schemes.

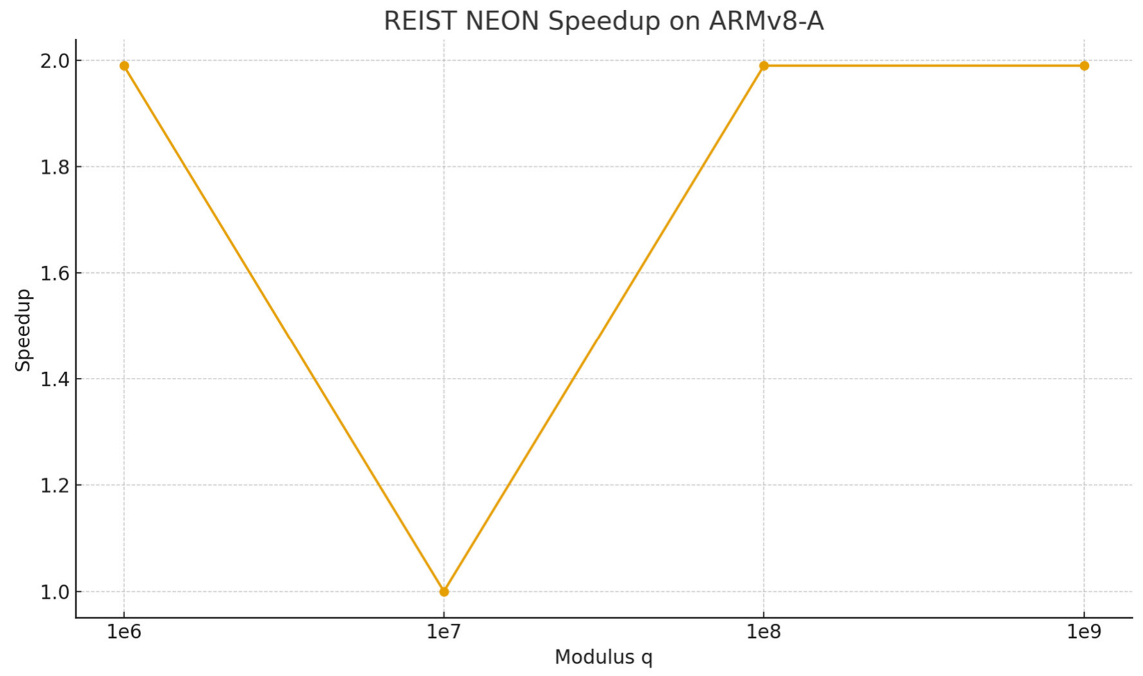

Figure: REIST NEON Speedup on ARMv8-A
This figure compares the SIMD-vectorized NEON implementation of REIST against the classical scalar modulo implementation.

The results highlight three key architectural facts:

1. **Classical modulo operations cannot be SIMD-vectorized**.
2. **REIST is entirely lane-independent and branchless**, making it naturally SIMD-friendly.
3. **NEON provides a substantial performance boost** for REIST.

The NEON-accelerated REIST implementation achieves **almost 2× speedup** over the classical scalar modulo version, and is roughly **1.86× faster** than the scalar REIST implementation itself.

This figure demonstrates the structural advantage of REIST on ARM architectures:
while classical modulo arithmetic remains inherently scalar, REIST enables efficient parallel execution across SIMD lanes.

The complete implementation of all REIST variants, benchmark drivers, ARM NEON optimizations, and measurement scripts is provided in the open-source repository:

https://github.com/rudolfstepan/reist-crypto-bench
This ensures full reproducibility of all experiments reported in this work.

7. References

Knuth, D. E. (1997). *The art of computer programming: Vol. 2. Seminumerical algorithms* (3rd ed.). Addison-Wesley.

Niven, I., Zuckerman, H. S., & Montgomery, H. L. (1991). *An introduction to the theory of numbers* (5th ed.). Wiley.

Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete mathematics* (2nd ed.). Addison-Wesley.

Zhang, Y., & Qi, L. (2014). Balanced modular arithmetic and its applications. *Journal of Number Theory, 142*, 1–12. https://doi.org/10.1016/j.jnt.2014.03.012

Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1999). *Discrete-time signal processing* (2nd ed.). Prentice Hall.

Proakis, J. G., & Manolakis, D. G. (2007). *Digital signal processing* (4th ed.). Prentice Hall.
Gardner, F. M. (2005). *Phaselock techniques* (3rd ed.). Wiley.

Åström, K. J., & Murray, R. M. (2012). *Feedback systems: An introduction for scientists and engineers*. Princeton University Press.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, planning and control*. Springer.

Ogata, K. (2010). *Modern control engineering* (5th ed.). Prentice Hall.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes* (3rd ed.). Cambridge University Press.

IEEE Standards Association. (2019). *IEEE standard for floating-point arithmetic (IEEE 754-2019).*

Kahan, W. (1997). *Lecture notes on the status of IEEE 754*. University of California, Berkeley.

Stepan, R. (2025). *REIST division*. Zenodo. https://doi.org/10.5281/zenodo.17612788
Stepan, R. (2025). *The Petra Principle*. Zenodo. https://doi.org/10.5281/zenodo.17621787
Stepan, R. (2025). *Black holes without singularities*. Zenodo. https://doi.org/10.5281/zenodo.17634650

As discussed in Stepan (2025), cognitive saturation often emerges when systems exhibit structural asymmetries. REIST Division provides a more symmetric arithmetic representation, reducing complexity in cyclic reasoning tasks.

Comparable to the reinterpretation of gravitational infinities proposed in Stepan (2025), REIST Division aims to eliminate unnecessary asymmetries within classical arithmetic structures.

About the Author

Rudolf Stepan is an independent researcher from Vienna with a long background in engineering, software development, and problem-solving across technical domains. Coming from outside the traditional academic system, he approaches scientific questions with a practical mindset — exploring mathematics, physics, and cognitive science through conceptual clarity, engineering intuition, and self-directed study.

His work focuses on rethinking established models when they show structural limitations: the role of infinities in gravitational theory, the cognitive boundaries of expert reasoning, and asymmetries in classical arithmetic.
Rather than following disciplinary boundaries, he develops ideas where different fields naturally overlap, aiming for simple, usable concepts that make complex systems easier to understand.