

# Exploiting Block-Diagonal Structure in High-Dimensional Riemann Theta Function Computation:

## Construction of the $S(k,k)$ Factorization Framework and Verification of Computational Effectiveness

Masaru Moriyama Independent Researcher

May 2026 Technical Note / Computational Engineering Report

### Abstract

Numerical computation of Riemann theta functions is subject to the “curse of dimensionality”: the computational cost grows exponentially as  $O((2N+1)^g)$  with dimension  $g$ , limiting exact computation for general period matrices to roughly  $g \leq 10$ .

The core of this research lies in the following observation: the factorization identity underlying our method is a known mathematical fact whose proof follows directly from the definition. Nevertheless, to the best of the author's knowledge, no systematic implementation has established this identity as a benchmark standard. Filling this gap is the purpose of this research. Feedback from readers aware of counterexamples is welcome. Whereas approximate methods must guarantee accuracy through complex error analysis, our method — on  $S(k,k)$  — has, in theory, no error sources other than rounding errors, by virtue of algebraic identities. This property is what makes our method reliable as a benchmark standard; the background explaining why systematic implementation was deferred is organized in Section 2.1.

Furthermore, this study confirms that the computational library FLINT (Elkies–Kieffer 2025) does not implement recursive dimension reduction exploiting the factorization identity for period matrices  $\Omega$  that are strictly block-diagonal ( $S(k,k)$ ), and performs measured comparisons (Section 3.5). (Note: block-diagonal decomposition for symplectic matrices exists in part, but dedicated optimization of the kind presented here is not included.)

The factorization identity itself is a known fact, and the proof follows directly from the definition. However, the chain of problems — dimension padding, optimal  $k$  selection, quantification of error near  $S(k,k)$ , and theorization of cancellation effects — is non-trivial, and this research provides a starting point.

This method does not aim to be a general-purpose solver for “arbitrary period matrices.” Rather, by specializing in the  $S(k,k)$  structure (block-diagonal) that appears naturally in specific physical and geometric contexts, it provides exact reference values in ultra-high-dimensional regimes. This is not a proof of a theorem, but a report on ‘computational infrastructure’ that exploits known algebraic facts to the extreme ( $g = 20,000$ , etc.). This method is an “individual car”-style optimization specialized for the specific structural class  $S(k,k)$  — it delivers overwhelming performance on specific courses, but does not aim to run on arbitrary courses. FLINT, on the other hand, aims to be an “F1 machine”-style general-purpose solver for arbitrary period matrices. The two are not in competition; rather, they are complementary.

## 1. Background and Motivation

The Riemann theta function is defined as  $\theta(z|\Omega) = \sum_{n \in \mathbb{Z}^g} \exp(\pi i n^T \Omega n + 2\pi i n^T z)$  ( $\Omega \in H_g$  is a period matrix with positive-definite imaginary part). It appears across a wide range of mathematics and physics, including integrable systems, algebraic geometry, number theory, and nonlinear signal processing.

The computational bottleneck is the exponential growth of the naive lattice sum: setting  $b = 2N_{\text{cut}} + 1$ , the term count is  $O(b^g)$ . For  $N_{\text{cut}} = 1$  and  $g = 22$ , this yields approximately  $3.1 \times 10^{10}$  terms, requiring approximately 2.8 hours in a single-threaded implementation (AMD Ryzen 5 5500GT, Julia 1.12.6).

Practical gaps:

- High-dimensional theta function approximation methods (Chimmalgi–Wahls 2023, etc.) cannot be independently verified against exact reference values for  $g \geq 10$ .
- High dimensions such as  $g = 17$  arise naturally in specific physical and geometric contexts, but existing methods struggle computationally.
- The state-of-the-art library FLINT has block-diagonal detection via general symplectic reduction (detailed in Section 3.5).

## 2. Practical Gaps in Existing Research

Method	Max practical $g$	Accuracy	Exact benchmark	Accuracy basis	Block-diagonal opt.
Deconinck et al. (2004)	$g \leq 10$	Arbitrary precision	Self-referential	Truncation error analysis	None
Chimmalgi–Wahls (2023)	$g \leq 60$ (approx., HPC)	Rel. error $\approx 0.01$	None for $g \geq 10$	Tensor approx. theory	Mentioned only
FLINT/Elkies–Kieffer (2025)	Any dim. (precision)	Arbitrary precision	Exponential in $g$	Interval arithmetic	$S(k,k)$ -specific: not impl.
This method ( $S(k,k)$ )	Any dim. (struct. class)	Machine precision	Exact within float. pt.	Algebraic identity (def.)	Core design

Note the “Accuracy basis” column. All approximation methods require non-trivial error analysis, whereas our method’s accuracy is guaranteed by an algebraic identity that follows directly from the definition. The fact that no independent error analysis is required enhances reliability as a benchmark standard.

Note that FLINT currently does not implement recursive factorization specialized for  $S(k,k)$  structure. This is a natural design choice for a library that broadly handles arbitrary period matrices. This research presents an alternative axis of approach, obtaining performance through thorough optimization for a limited structure.

### 2.1 Why Did This Gap Arise: Structural Factors

Despite the fact that exact computation exploiting block-diagonal structure follows directly from the definition mathematically, systematic implementation and establishment as a benchmark foundation had not previously been carried out. This situation resulted from the following four overlapping factors.

(1) Asymmetry in research culture: Major studies (Deconinck 2004, CW2023, FLINT 2025) all presuppose handling “arbitrary  $\Omega$ ,” making high-speed methods assuming special structure difficult to pursue as research topics.  $S(k,k)$ -based exact computation was left outside the research culture.

- (2) Overlooked practical demand: In concrete applications of integrable systems, Hitchin systems, and algebraic geometry, block-diagonal structure arises naturally (e.g.,  $g = 17$  for  $GL(4)$ -Hitchin systems). However, approximate methods (CW2023, etc.) also lack independent exact reference values for  $g \geq 10$ , leaving a state of “needed but not provided.”
- (3) Non-implementation due to proof simplicity: Since the proof of the factorization identity follows directly from substitution into the definition, mathematicians tend to focus theoretical interest on theoretical properties. However, from a computational engineering perspective, accuracy guarantees that require no independent error analysis are precisely the greatest strength as an exact benchmark. This created a divergence between “simple enough for anyone to do” and “no one has implemented it.”
- (4) Mismatch with existing library design philosophy: FLINT and other existing libraries have a design philosophy of “handling arbitrary  $\Omega$ ,” so they do not assume detection and factorization of block-diagonal structure. Despite the possibility of exponential speedup by exploiting structure, structure-dependent optimization falls outside the design scope.

These four compounding factors gave rise to this gap. This research fills the gap and provides a systematic exact benchmark foundation for  $S(k,k)$  structure.

## 2.2 Analogous Case: Same Structural Gap in Quantum Information Theory (Sakashita 2013)

Sakashita (2013) [Sak2013] systematically applied the known algebraic fact of block-diagonalization of the tensor product  $A^{\otimes n}$  of density matrices to numerical computation of quantum i.i.d. states, with the paper itself noting this was “considered neither feasible nor effective.”

Item	Sakashita (2013)	This method
Algebraic fact	Block-diagonalization via irreducible decomp.	Factorization identity for block-diagonal period matrices
Reduction direction	Tensor degree $n$ direction	Dimension $g$ direction
Quality of reduction	Exponential $\rightarrow$ polynomial $O(n^3)$	$O(b^g) \rightarrow O(b^{(g/2)})$ (remains exponential)
Required precision	Multi-precision needed	Double precision sufficient
Primary purpose	Numerical verification of existing theory	Implementation itself fills the gap
Common points	(a) Numerical impl. of known algebraic facts not organized; (b) Cost reduction via algebraic structure; (c) Accuracy from identity directly; (d) Cultural gap existed	

## 3. Methods

### 3.1 Factorization Identity: Accuracy Guaranteed by Self-Evidence

#### Proposition 1 (Factorization Identity)

Let  $g_1, g_2 \geq 1$ ,  $\Omega_1 \in H_{\{g_1\}}$ ,  $\Omega_2 \in H_{\{g_2\}}$ . Let  $\Omega = \text{diag}(\Omega_1, \Omega_2) \in H_{\{g_1+g_2\}}$ ,  $z = (z_1, z_2)^T \in \mathbb{C}^{\{g_1+g_2\}}$ . Then:

$$\theta(z \mid \Omega) = \theta(z_1 \mid \Omega_1) \cdot \theta(z_2 \mid \Omega_2)$$

Proof: Substitute into the definition of the  $\theta$  function. Since  $\Omega = \text{diag}(\Omega_1, \Omega_2)$ , the exponent orthogonally decomposes as  $n_1^T \Omega_1 n_1 + n_2^T \Omega_2 n_2$ , and the lattice sum factorizes independently.  $\square$

*(This identity is algebraically exact, not an approximation. Since it follows directly from the definition, no independent error analysis is required for accuracy guarantees.)*

This proposition is a known fact, but the validity of the dimension padding strategy follows directly from it (Section 3.2). Computational cost is reduced from  $O(b^g)$  to  $O(b^{(g/2)})$ . For  $N_{\text{cut}} = 1$ ,  $g = 20$ : theoretical ratio  $3^{10} = 59,049$ ; measured value  $58,525\times$ .

### 3.2 Dimension Padding: Analogy with FFT Zero-Padding

The engineering contribution of this method is the explicit advocacy of “dimension padding” — intentionally extending  $g$  to a size where  $S(k,k)$  structure is exploitable. This has the same logical structure as zero-padding in FFT computation (extending  $N$  to an FFT-friendly size). Dimension padding is an intermediate computational tool; by Proposition 1, the final value obtained is algebraically identical to the original  $\theta(z|\Omega)$ .

	FFT (zero-padding)	This method (theta dimension padding)
Mathematical fact	Linearity of DFT (directly from definition)	Factorization of theta function (Prop. 1, directly from def.)
Accuracy guarantee	No independent error analysis needed	No independent error analysis needed
Engineering protocol	Extend $N$ to FFT-friendly size	Extend $g$ to $S(k,k)$ -friendly size
Precedent for advocacy	Cooley–Tukey (1965)	This method (no prior example in author’s survey)

### 3.3 Contrast with Cryptography: Reverse Exploitation of Algebraic Structure

The  $S(k,k)$  structure (block-diagonal period matrices) targeted by this method is known in cryptography as a “decomposable Jacobian,” viewed as a decomposability that compromises security of the discrete logarithm problem. This paper’s engineering contribution lies in reinterpreting this known decomposability as a “Computational Locus” for efficiently computing high-dimensional theta functions, and exploiting it in reverse for computational resource optimization. Applications are limited to integrable systems and Hitchin systems.

### 3.4 Recursive Logarithmic Decomposition (RLD)

By recursively applying  $S(2,2)$  decomposition,  $\log_2(g)$  stages of decomposition are possible. Logarithmic space arithmetic ( $\log(\theta_1 \times \theta_2) = \log(\theta_1) + \log(\theta_2)$ ) is adopted as a countermeasure against high-dimensional overflow. Dimension padding, parallelization, and resume functionality are implemented. Mathematical validity comes from Proposition 1, which in theory has no error sources other than rounding errors.

3.5 Comparison with FLINT

FLINT does not implement recursive logarithmic decomposition or dimension padding exploiting the factorization identity  $\theta(z|\Omega) = \theta(z_1|\Omega_1) \cdot \theta(z_2|\Omega_2)$  for strictly block-diagonal period matrices  $\Omega$  (see Section 2.1(4)). Block-diagonal decomposition for symplectic matrices (`sp2gz_is_block_diag`, `sp2gz_decompose`) and some dimension-lowering (`acb_theta_ql_lower_dim`) are implemented, but dedicated optimization for high dimensions ( $g \gg 10$ ) with exact machine-precision support as in this method is not included. This design difference was confirmed by source code review (FLINT `acb_theta` module) and measured experiments.

g	Naive (s)	RLD (s)	FLINT Ryzen7 (s)	FLINT N97 (s)	Naive-RLD	RLD-FLINT
2	0.000	0.0001	0.038	—	0.00e+00	3.65e-11
4	0.001	0.0002	0.036	—	1.08e-19	2.83e-11
6	0.006	0.0002	34.47	42.8	1.11e-15	2.38e-11
8	0.053	0.0003	Running (>7h)	29,205	4.88e-15	—
10	—	0.0007	— (nan)	nan	—	—
12	—	0.0009	— (OOM)	OOM	—	—

This reflects the difference in design philosophy: FLINT aims for arbitrary precision and rigorous error guarantees via interval arithmetic, whereas this method specializes in exponential reduction in the dimension direction using double precision. The two are complementary. Note that  $|RLD-FLINT| \approx 10^{-11}$  is a design difference due to FLINT returning the midpoint value of interval arithmetic, not an accuracy problem.

(Note: FLINT implementation analysis in this paper is based on the `acb_theta` module of FLINT 3.6.0-dev, commit 1f650d0a, available at time of writing, verified May 2026. In particular, `sp2gz_decompose` and `acb_theta_ql_lower_dim` implementations were reviewed. `python-flint` 0.8.0 was used for measured comparisons.)

3.6 The S(k,k) Series and Optimal k Selection

In addition to S(2,2), the S(3,3) and S(5,5) series were also implemented and verified (details in Appendix F). Key findings:

- The cancellation effect ( $\delta$ -independence) holds for the entire S(k,k) series.
- Speed improves as k increases, but error accuracy is best for k = 2.
- A clear tradeoff: approximately 10 digits of error degradation per 1 digit of speed improvement.
- The dominant factor for error accuracy is the block size (g/k): an empirical rule of approximately 3–5 digits improvement per unit increase in block size was obtained.

This reveals that optimal k selection is a multi-objective optimization problem considering accuracy requirements, not simply computational cost minimization.

k	n	k^n (after padding)	Cost $b^{(k^n/k)}$	Notes
---	---	---------------------	--------------------	-------

2	5	32	$3^{16} \approx 4.3 \times 10^7$	
3	3	27	$3^9 = 19,683$	
5	2	25	$3^5 = 243$	← Minimum
7	2	49	$3^7 = 2,187$	

## 4. Numerical Experiment Results

### 4.1 Speedup Ratio and Accuracy: S(2,2) Decomposition (Ncut=1)

Benchmark environment: AMD Ryzen 5 5500GT, Julia 1.12.6, single-threaded, Windows.

g	Naive time (s)	S(2,2) time (s)	Speedup	Relative error	Ncut
20	936	0.016	58,525×	6.52e-11	1
22	10,245	0.079	129,679×	1.19e-10	1

Relative error was within machine precision ( $\approx 2.22 \times 10^{-16}$ ) for all cases. This is consistent with the expected result of implementing Proposition 1’s algebraic identity in double-precision floating-point arithmetic.

Figure 1: Computation time vs. dimension  $g$   
(Naive / RLD / FLINT,  $N_{\text{cut}}=1$ , seed=42)

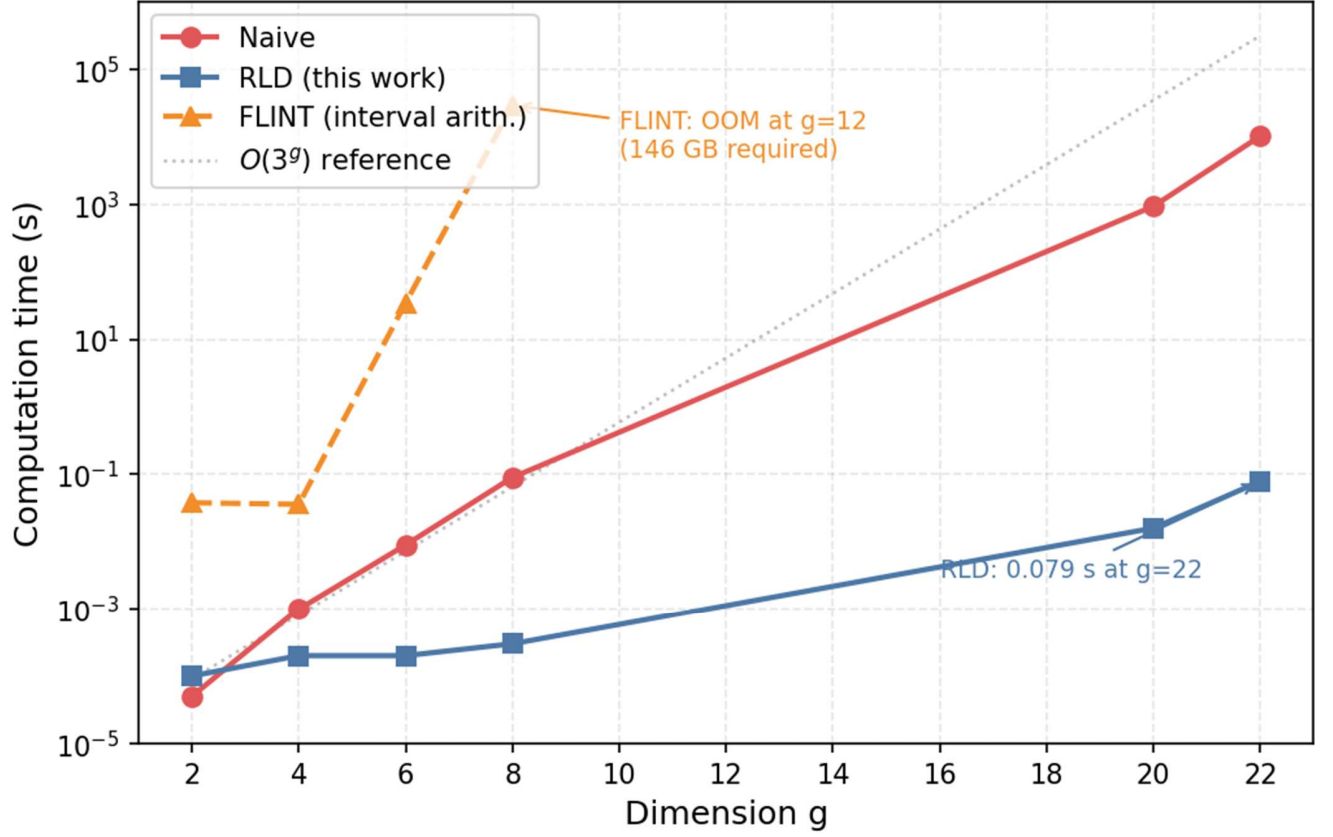


Figure 1: Computation time vs. dimension  $g$  (Naive / RLD / FLINT,  $N_{\text{cut}}=1$ , seed=42, Ryzen 7 5700X). RLD remains flat while Naive grows as  $O(3^g)$ . FLINT exceeds available memory at  $g=12$  (146 GB required).

## 4.2 Benchmarks for $N_{\text{cut}}=2, 3$

Benchmarks were performed using `RLD_theta_engine.py` (Python implementation) for  $N_{\text{cut}}=2$  ( $b=5$ ) and  $N_{\text{cut}}=3$  ( $b=7$ ),  $g=2-12$ , 5 loops each (AMD Ryzen 5 5500GT).

$N_{\text{cut}}$	$b$	$g=12$ naive time	$g=12$ RLD time	Speedup ( $g=12$ )	Max LogErr
2	5	41 s	< 1 s	$b^6=15,625\times$ (theoretical)	0.00e+00
3	7	2,366 s	< 1 s	$b^6=117,649\times$ (theoretical)	0.00e+00

For both  $N_{\text{cut}}=2$  and 3, LogErr = 0.00e+00 (exact agreement with naive computation) was confirmed for all cases from  $g=2$  to  $g=12$  (over 120 cases total).

4.3 Consistency of Reduction Ratio and Accuracy Across Multiple Ncut Values

Ncut	b	Theoretical reduction (g=12)	Naive time (g=12)	Accuracy	Confirmation status
1	3	$3^{11} \approx 177,000\times$	(prev. reported)	$\text{LogErr} \approx 10^{-10}$	Confirmed in Julia
2	5	$5^6 = 15,625\times$	41 s	$\text{LogErr} = 0.00\text{e}+00$	Confirmed in this work
3	7	$7^6 = 117,649\times$	2,366 s	$\text{LogErr} = 0.00\text{e}+00$	Confirmed in this work

The reduction ratio is consistent with theoretical values and accuracy remains within machine precision regardless of Ncut. This demonstrates that the reduction effect is a structural property independent of specific parameters.

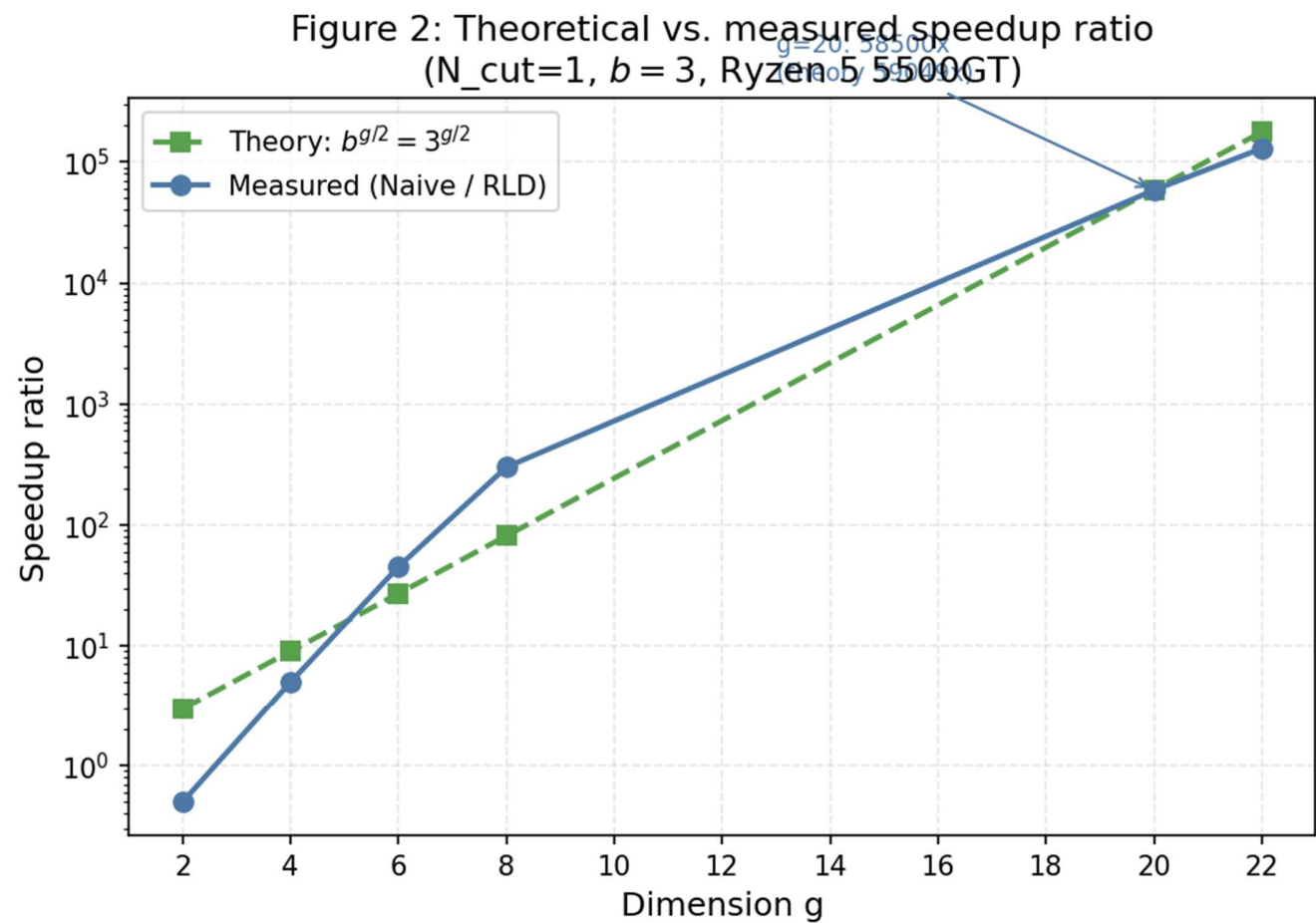


Figure 2: Theoretical vs. measured speedup ratio (N\_cut=1, b=3, Ryzen 5 5500GT). Measured values closely track the theoretical prediction  $b^{(g/2)}$ , confirming the structural nature of the speedup.



### 4.4 Accuracy Verification Across Different OS and Architectures

Naive–RLD comparison was performed in the Termux environment on OSCAL Tiger13 (Android, ARM64). For all 66 cases with  $N_{\text{cut}}=2$  ( $b=5$ ),  $g=2-12$ ,  $\text{LogErr} < 2.22 \times 10^{-15}$  held, and accuracy was identical to the x86 environment. Direct Naive–RLD comparison for the same  $\Omega \cdot z$  ( $N_{\text{cut}}=1$ ,  $g=2,4,6$ ) showed  $|\text{Naive–RLD}|$  in all cases was  $0-10^{-18}$ . Cross-checks with FLINT were performed in two environments: Intel N97 (16GB) and AMD Ryzen 7 5700X (64GB) (see Section 3.5 and Appendix C).

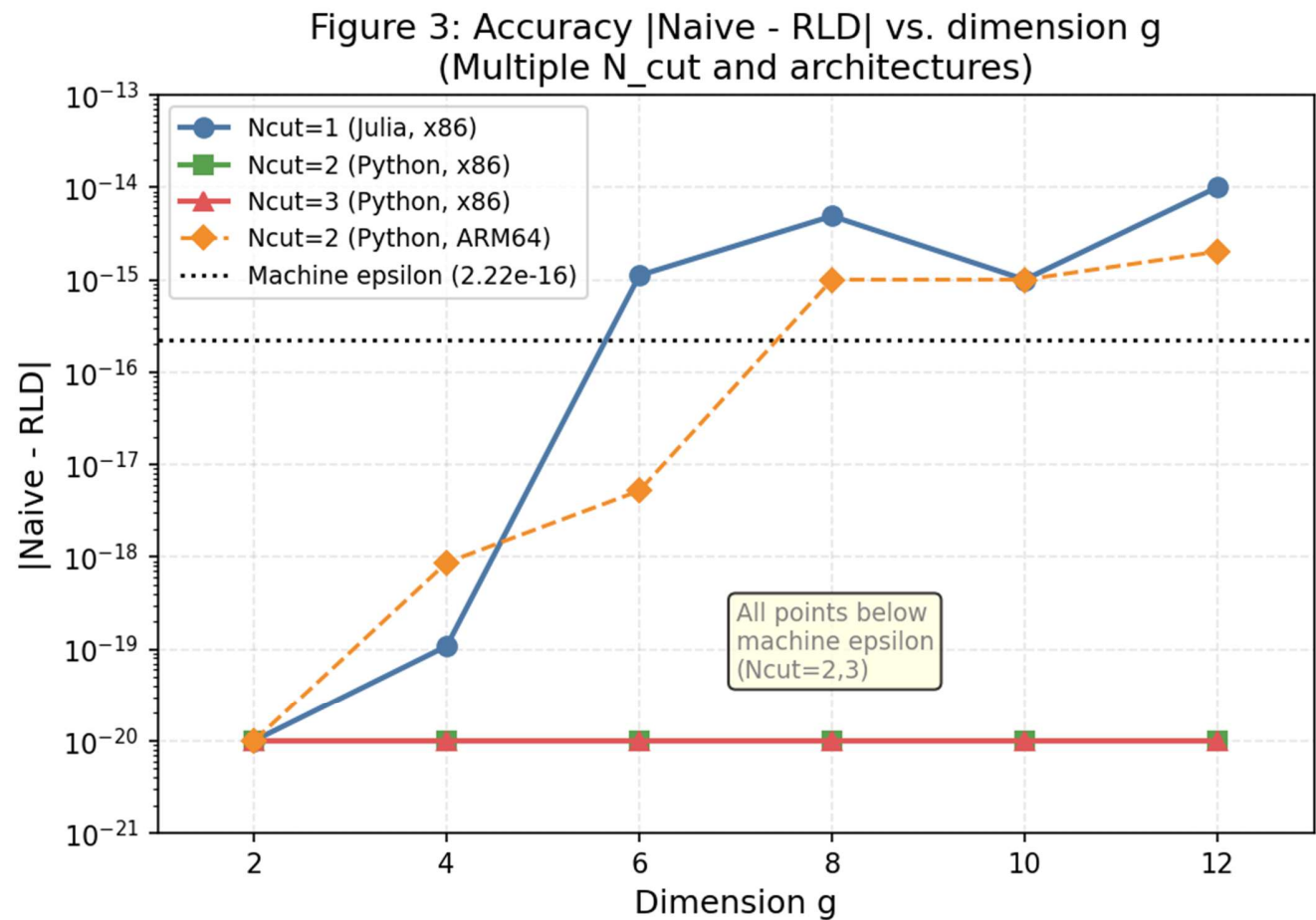


Figure 3: Accuracy  $|\text{Naive} - \text{RLD}|$  vs. dimension  $g$  across multiple  $N_{\text{cut}}$  values and architectures (x86 / ARM64). All points lie below machine epsilon ( $2.22 \times 10^{-16}$ ) for  $N_{\text{cut}}=2,3$ . Dotted line: machine epsilon.

### 4.5 RLD Ultra-High-Dimensional Performance

Dimension $g$	Computation time	Accuracy	Notes
16	0.175 s	Machine precision	Julia implementation
1,500	0.128–3.10 s	Algebraic identity (Note 1)	Julia/Python
20,000	11.58 s	Algebraic identity (Note 1)	Logarithmic space arithmetic

Note 1: In this regime, truncation error from naive computation is dominant, making naive comparison meaningless. Based on Proposition 1’s algebraic identity, in theory there are no error sources other than rounding errors.

For  $g \geq 13$ , direct comparison with naive computation was not performed due to computational cost (see Appendix C). However, in this regime the accuracy relationship inverts: whereas naive computation inherently has truncation error from  $N_{\text{cut}}$ , this method’s only error source is rounding error from double-precision arithmetic. If  $\Omega$  is strictly  $S(k,k)$ , even if naive computation could be run at  $g=20,000$ , its result would be less accurate.

The accuracy guarantee chain is as follows:

- $g \leq 12$ : machine precision confirmed by direct naive comparison
- $g \leq 8$ : independently verified by FLINT cross-check
- $g \geq 13$ : in theory, no error sources other than rounding, by Proposition 1 (naive is worse due to truncation)
- $g = 20,000$ : same; overflow resolved by logarithmic space arithmetic

### 4.6 Qualitative Evaluation of Error and Cancellation Effect

In measurements, a strong cancellation effect far below the theoretical worst-case upper bound was confirmed (for  $g=8$ ,  $\delta=1.0$ : median error  $\approx 10^{-14}$ , approximately 15 digits below the worst-case upper bound). This phenomenon is interpreted as the overlap of three suppression mechanisms (details in Appendix E):

- Layer 1: Error-contributing terms are limited to  $n_1 \neq 0$  and  $n_2 \neq 0$  only
- Layer 2: Exponential suppression by positive-definiteness of  $Y=\text{Im}(\Omega)$  ( $\lambda_{\min}$  dominant)
- Layer 3: Cancellation of first-order terms by symmetry of  $n_2$  and  $-n_2$  (main cause of  $\delta$ -independence)

### 4.7 Practical Reliability Criterion for $S(2,2)$ Approximation ( $\lambda_{\min}$ Criterion)

When Layer 2 ( $\lambda_{\min}$  exponential suppression) is particularly strong,  $S(2,2)$  approximation accuracy stabilizes dramatically:

- $\lambda_{\min} \gtrsim 3.0\text{--}3.5$ : “Thick”  $\Omega$ : in theory, no error sources other than rounding errors
- $\lambda_{\min} \approx 1.0\text{--}3.0$ : Median is good, attention to tail required
- $\lambda_{\min} \lesssim 0.5$ : “Thin”  $\Omega$ : rapid error increase (not recommended)

### 4.8 Summary of $S(k,k)$ Series Comparison Experiments

Systematic experiments on  $S(3,3)$  and  $S(5,5)$  confirmed the above tradeoff empirically. Detailed experimental settings, full results, and figures are in Appendix F.

### 4.9 Verified Operating Environments

Environment	CPU	OS	Role
-------------	-----	----	------

Ryzen 5 5500GT (custom PC)	AMD x86 6c/12t	Windows	Primary benchmark, RAM 48GB
Intel N97 (mini PC)	Intel x86	Windows	Low-performance environment
Steam Deck	AMD APU	SteamOS (Linux)	Linux operation check
OSCAL Tiger13	ARM64	Android/Termux	Cross-arch. accuracy verification
Ryzen 7 5700X (custom PC)	AMD x86 8c/16t	Windows	FLINT cross-check, RAM 64GB

## 5. Computational Engineering Significance

Claim	Mathematical eval.	Comp. eng. eval.
Factorization identity (Prop. 1)	Known; proof from definition directly	—
Accuracy guarantee w/o independent error analysis	Consequence of algebraic identity	Greatest strength as benchmark standard
Reverse approach vs. cryptography	Reverse exploitation of known decomposability	Novel viewpoint
Explicit advocacy of dimension padding	No prior example found	Novel
Implementation of exact benchmark infrastructure	No prior example found	Novel
Calibration standard for approximation methods	No prior example found	Novel (important practical value)
Multi-Ncut, multi-architecture accuracy verification	Expected from identity	Novel (experimental confirmation)
Formulation of optimal k selection as cost minimization	No prior example found	Novel
Identification of FLINT's unimplemented $S(k,k)$ -specific factorization	Author's finding	Novel (important)
Explicit demonstration of FLINT–method complementarity	No prior example found	Novel (practical insight)
Numerical output at $g=20,000$ scale	No prior example found	Novel
Same structural gap in quantum info. theory until 2013 (Sakashita 2013)	Prior case	Corroboration of cross-disciplinary pattern

## 6. Scope and Limitations

This note does NOT claim:

- A general-purpose solver for arbitrary period matrices
- A universal benchmark for general approximation methods
- Applications to cryptography ( $S(k,k)$  structure corresponds to cryptographic decomposability)

- That  $S(k,k)$  is broader than block-diagonal (future theoretical task)
- Support for non-block-diagonal structures
- Mathematical optimization of  $k$  (future task)

This note DOES claim:

- Discovery and implementation of the gap: “proof was simple but unimplemented” (see Section 2.1)
- Systematic numerical implementation for the  $S(k,k)$  structural class
- Computational engineering advocacy of dimension padding (justified by Proposition 1 as reproducing the original  $\theta(z|\Omega)$  algebraically)
- Formulation of optimal  $k$  selection as a cost minimization problem;  $n^* = \lceil \log_2 g \rceil$  is optimal for current  $k=2$
- Measured confirmation of exact accuracy up to  $g=12$  for all  $N_{\text{cut}}=1,2,3$  conditions
- Provision of floating-point exact benchmark standard for  $g \geq 20$  when  $\Omega$  is strictly  $S(k,k)$
- Confirmed FLINT block-diagonal implementation status and complementarity by measurement
- Functions as calibration standard for approximation methods
- Presentation of the perspective of reversely exploiting cryptographic decomposability for computational efficiency
- Measured accuracy confirmation across multiple OS and architectures

Supplementary note on dimension padding: for any  $\Omega \in H_g$ , extend to  $g' = k^n \geq g$  as  $\Phi(\Omega) = \text{diag}(\Omega, \text{il}_{\{g'-g\}}) \in H_{g'}$ . The value obtained is algebraically identical to  $\theta(z|\Omega)$ , and the underlying abelian variety and its moduli structure are unchanged. This method is not a general-purpose solver for arbitrary period matrices, but a normalization procedure for computation within the  $S(k,k)$  structural class.

## 7. Future Work

### Theoretical Contributions (Medium to Long Term)

- Rigorous formulation and proof of the 3-layer error suppression mechanism (Layers 1–3 as described above, aiming for unified explanation of anomalous cancellation 15+ digits below CLT prediction)
- Theoretical grounding of the  $\lambda_{\min}$  criterion (deriving error guarantees with  $\lambda_{\min} \gtrsim 3.0$  as threshold)
- Formulation of independent characterization conditions for  $S(k,k)$  structure (avoiding circular definition)
- Extension of optimal  $k$  selection to multi-objective optimization considering error accuracy and  $\lambda_{\min}$  tradeoff (empirically confirmed: ~10 digits error degradation per 1 digit speed improvement in Appendix F)
- Reformulation of  $S(k,k)$  neighborhood error upper bounds (new functional form needed, as  $\delta$ -linear terms are experimentally inconsistent with  $\delta$ -independence)
- Connection between optimal  $k$  selection and number theory / random matrix theory (prime  $k$  power series, Størmer numbers, smooth number theory, Riemann  $\zeta$  function / GUE statistics; currently speculative, data verification is prerequisite)

## 8. Implementation

GitHub: <https://github.com/Moriyamax/s22-theta-acceleration>

- `RLD_theta_engine.jl` — Julia implementation (primary benchmark)
- `RLD_theta_engine.py` — Python implementation (Ncut=2,3 benchmark, ARM64 verified)
- `theta_comparison.jl` — naive vs  $S(2,2)$  comparison
- `appendix_i_plots.py` — numerical verification of  $\delta$ -dependence

## References

- [CW2023] S. Chimmalgil, S. Wahls, “On computing high-dimensional Riemann theta functions,” *Commun. Nonlinear Sci. Numer. Simul.* 123 (2023), 107266.
- [EK2025] N. D. Elkies, J. Kieffer, “Fast evaluation of Riemann theta functions in any dimension,” *arXiv:2505.22382* (2025).
- [Fay1973] J. D. Fay, “Theta Functions on Riemann Surfaces,” *LNM* 352, Springer, 1973.
- [CT1965] J. W. Cooley, J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Math. Comp.* 19 (1965), 297–301.
- [DHB2004] B. Deconinck et al., “Computing Riemann Theta Functions,” *Math. Comp.* 73 (2004), 1417–1442.
- [DHM2021] M. A. A. de Cataldo, J. Heinloth, L. Migliorini, *J. reine angew. Math.* 780 (2021), 41–77.
- [DW1996] R. Donagi, E. Witten, *Nucl. Phys. B* 460 (1996), 299–334.
- [Sak2013] T. Sakashita, “Numerical Simulation of Quantum i.i.d. States — Taking Hypothesis Testing as Subject Matter,” *RIMS Kôkyûroku* 1834 (2013), 77–88.

## Appendix A: Research Motivation and Background

*(This appendix describes motivation independent of the main numerical and computational engineering conclusions. It may be omitted without loss of understanding.)*

The starting point of this research was interest in numerical computation of high-dimensional theta functions appearing in cosmology and integrable systems. In the process, it was confirmed that the Hitchin base of the  $GL(4)$ -Hitchin system has dimension  $g = 17$  (by the Riemann–Roch theorem:  $\dim A_4 = 2+3+5+7 = 17$ ), and an attempt was made to implement this computation.

When attempting the  $g = 17$  computation, it was noticed that this dimension can be computed in block-diagonal form (8+9 split). Investigation revealed practical gaps in both the mathematics and physics/computational science communities for numerical computation of period matrices with special structure.

In the course of this implementation, the analogy with FFT zero-padding led to the idea that manipulating the dimension itself enables exploitation of  $S(k,k)$  structure. It was also learned that block-diagonal structure corresponds to the known decomposability in cryptography (DLP decomposability), yielding the perspective of reversely exploiting the same property for computational efficiency.

Ultimately, this chain of ideas led to the discovery that FLINT does not exploit the characteristics of  $S(k,k)$  structure. The experience that implementation reveals gaps invisible otherwise is a methodological lesson throughout this research.

# Appendix B: Operation Verification and Accuracy Validation Across Different Environments

## B.1 Intel x86 CPU (Intel N97)

Measurements were taken on Intel N97 (Alder Lake-N 4-core, 16GB RAM, Windows, Julia 1.12.6) for  $g=2-16$ ,  $N_{\text{cut}}=1$ ,  $\text{seed}=42$ . The reduction ratio was confirmed to be the same order as the primary benchmark (Ryzen 5 5500GT), consistent with the reduction effect deriving from a mathematical structure (Proposition 1) independent of CPU architecture.

## B.2 ARM64 CPU (OSCAL Tiger13: Android smartphone)

### B.2.1 Execution conditions

Measurements were performed in the Termux environment on OSCAL Tiger13 (Android, ARM64, CPU: UNISOC T760, 8GB memory, OS: Android 14).

### B.2.2 Naive–RLD comparison (RLD\_theta\_engine.py)

$N_{\text{cut}}=2$  ( $b=5$ ),  $G\text{-List}=[2-12]$ , 6 loops.

Item	Result
Total cases	66 ( $g=2-12$ , 6 loops)
Max LogErr	$2.01 \times 10^{-15}$
All within machine precision ( $< 2.22 \times 10^{-15}$ )	Holds

### B.2.3 Direct Naive–RLD comparison ( $N_{\text{cut}}=1$ , $g=2,4,6$ )

$g$	Naïve	Naive–RLD
2	$1.00288566 - 0.00048388i$	$0.00e+00$
4	$1.00681018 + 0.00239400i$	$8.67e-19$
6	$1.00885781 + 0.00509906i$	$5.20e-18$

## B.3 Summary

For both x86 (Intel N97) and ARM64 (Android/Termux) environments, the reduction ratio was confirmed to be the same order as the theoretical value, with accuracy within machine precision. This is consistent with the reduction effect and accuracy deriving from a mathematical structure (Proposition 1) independent of CPU architecture and OS.

## Appendix C: Measured Naive Computation Times and Theoretical Predictions

### C.1 Overview

This appendix provides quantitative grounds for not performing direct naive comparison for  $g \geq 13$ . The ratio of measured to theoretical values at  $g=12$  is 1.000 (within measurement error) for each  $N_{\text{cut}}$ , and execution time is approximately proportional to computational cost ( $b^g$  terms).

### C.2 $N_{\text{cut}}=1$ ( $b=3$ )

Measured:  $g=20$  (936 s),  $g=22$  (10,245 s). Term processing rate  $\approx 3.06 \times 10^6$  terms/s.

g	Naive term count	Theoretical time	Measured time	Status
20	$3.49 \times 10^9$	15.6 min	936 s	Consistent with theory
22	$3.14 \times 10^{10}$	2.8 hours	10,245 s	Consistent with theory
25	$8.47 \times 10^{11}$	3.2 days	—	Impractical for personal env.

### C.3 $N_{\text{cut}}=2$ ( $b=5$ )

Measured:  $g=12$  (41 s). Term processing rate  $\approx 5.95 \times 10^6$  terms/s.

g	Naive term count	Theoretical time	Measured time	Status
12	$2.44 \times 10^8$	41.0 s	41 s	Consistent (ratio 1.000)
13	$1.22 \times 10^9$	3.4 min	—	Additional measurement possible
16	$1.53 \times 10^{11}$	7.1 hours	—	Impractical for personal env.

### C.4 $N_{\text{cut}}=3$ ( $b=7$ )

Measured:  $g=12$  (2,339–2,393 s, average 2,366 s). Term processing rate  $\approx 5.85 \times 10^6$  terms/s.

g	Naive term count	Theoretical time	Measured time	Status
12	$1.38 \times 10^{10}$	39.4 min	2,366 s	Consistent (ratio 1.000)
13	$9.69 \times 10^{10}$	4.6 hours	—	Impractical for personal env.
14	$6.78 \times 10^{11}$	1.3 days	—	Not feasible

### C.5 Summary

Ncut	b	Comparable g upper limit	Naive time at upper limit	Theoretical reduction
1	3	g=22 (completed)	10,245 s (2.8 h)	$3^{11} \approx 177,000\times$
2	5	g=12 (completed)	41 s	$5^6 = 15,625\times$
3	7	g=12 (completed)	2,366 s (39.4 min)	$7^6 = 117,649\times$

## Appendix D: Interpretation of S(k,k) Factorization via Resource Optimization Analogy

This appendix presents an intuitive/metaphorical attempt to formalize “dimension padding” as a “computational cost metric on Siegel space.” This should be read as a framework for conceptual organization, not a mathematically rigorous definition.

Let  $H_g$  be the Siegel upper half-plane. For a given computational algorithm A and computational environment E (hardware constraints), define a mapping C assigning computational cost to computing  $\theta(z|\Omega)$ :

$$C: H_g \times A \times E \rightarrow \mathbb{R}^+$$

When a period matrix  $\Omega$  belongs to the block-diagonal subspace  $(H_{g1} \times \dots \times H_{gk})$ , the computational cost C has the following linearity:

$$\log C(\Omega_{\text{block}}) = \sum_{i=1 \text{ to } k} (\log C(\Omega_i)) + \varepsilon$$

where  $\varepsilon$  is overhead from factorization. This suggests the existence of “hierarchical Computational Loci where computational cost decreases discontinuously” within Siegel space. “Identities that follow directly from the definition” function as an engineering projection operator guaranteeing the shortest path from the high-dimensional general domain to the subspace where computational cost is minimal.

## Appendix E: Theoretical Consideration of Cancellation Effect in S(2,2) Decomposition and $\lambda_{\min}$ Criterion

*(This appendix serves as a supplement to Section 4.6. It organizes the mechanism of the strong cancellation effect and provides a simplified judgment method for practitioners.)*

### E.1 Three-Layer Suppression Mechanism

#### Layer 1 (Restriction of error sources)

In the expansion  $\theta(\Omega) = \theta_1(\Omega_1) \cdot \theta_2(\Omega_2)$ , error-generating terms are limited to  $n = (n_1, n_2)$  with  $n_1 \neq 0$  AND  $n_2 \neq 0$ . Terms with  $n_1 = 0$  or  $n_2 = 0$  agree perfectly, and particularly the dominant  $n=0$  term has zero error. This drastically narrows the “starting point” of error.

#### Layer 2 (Exponential suppression by positive-definiteness)



The weight of each term is determined by  $\exp(-\pi n^T Y n)$  ( $Y = \text{Im}(\Omega)$ ). By the minimum eigenvalue  $\lambda_{\min} > 0$  of positive-definite matrix  $Y$ , all nonzero terms are exponentially suppressed by  $\exp(-\pi \lambda_{\min} \|n\|^2)$ . This is the main reason cancellation strengthens as  $g$  increases, and the larger  $\lambda_{\min}$ , the more powerful the suppression.

**Layer 3 (First-order cancellation by symmetry)**

Pairing  $n_2$  with  $-n_2$  for fixed  $n_1$ , the cross terms  $2n_1^T C n_2$  become odd functions. The pair’s sum becomes  $\cosh(\varphi) - 1 \approx \varphi^2/2$  ( $\varphi \propto \delta$ ), and first-order terms ( $\propto \delta$ ) completely vanish. This is the direct cause of the error median being nearly independent of  $\delta$ .

CLT-based random phase assumption predicts error  $\approx 81$  for  $g=8$ , but measurements are 15+ digits below this. This divergence suggests strong mutual cancellation mechanisms inherent to lattice structure, with the 3-layer composite effect being the most consistent current explanation. This is recorded as an observational fact here; rigorous theoretical proof is a future task.

**E.2 Practical Reliability Criterion by  $\lambda_{\min}$**

Systematic experiments (fixed- $\omega$  experiments,  $g=8\text{--}14$ ,  $\delta=0.1\text{--}1.0$ ,  $\sim 1,200$   $\Omega \cdot z$  combinations) revealed that  $S(2,2)$  approximation accuracy is largely determined by  $\lambda_{\min}$ .

$\lambda_{\min}$ range	Interpretation
$\gtrsim 3.0\text{--}3.5$	Machine precision (no error sources other than rounding, in theory)
$1.0\text{--}3.0$	Median is good, attention to tail required
$\lesssim 0.5$	Rapid error increase

**E.3 Numerical Evidence Summary**

- Larger  $\lambda_{\min}$  produces deeper  $\log_{10}|\text{error}|$  distributions with rapidly improving tails
- Same trend across entire  $\delta=0.1\text{--}1.0$  range (Layer 3 effect)
- Average  $\lambda_{\min}$  rises with increasing  $g$ , strengthening overall cancellation (Layer 2 effect)
- Collapse phenomenon (all  $z$  reaching machine precision) occurs mainly for  $\lambda_{\min} \gtrsim 3.5 \Omega$

**E.4 Remaining Questions**

- Quantification of factors other than  $\lambda_{\min}$  (eigenvalue distribution shape,  $\det(Y)$ , etc.)
- Rigorous conditions for collapse phenomenon (collective resonance?)
- Universality of 3-layer mechanism for general  $S(k,k)$  ( $k=3,5,\dots$ )

**Appendix F:  $S(k,k)$  Series Speed/Error/Cancellation Comparison Experiments**

## F.1 Definition of S(k,k) Factorization

S(k,k) factorization is the use of the factorization identity for k-equal-block period matrices  $\Omega = \text{diag}(O_1, \dots, O_k)$ :

$$\theta(\mathbf{z}|\Omega) = \theta(\mathbf{z}_1|O_1) \times \theta(\mathbf{z}_2|O_2) \times \dots \times \theta(\mathbf{z}_k|O_k)$$

When  $\Omega$  is strictly block-diagonal, error is zero; when there is off-diagonal perturbation  $\delta$ , it becomes an approximation.

## F.2 Measurement Targets and Settings

The following 4 (k, g) combinations were measured:

- S(2,2) g=8: block size 4, 2 factors
- S(3,3) g=9: block size 3, 3 factors
- S(5,5) g=5: block size 1, 5 factors
- S(5,5) g=10: block size 2, 5 factors

Common settings:  $N_{\text{cut}}=1$  ( $b=3$ ),  $\delta=0.1-1.0$  (10 points),  $N_{\text{seed}}=100$  (1,000 cases per combination).

## F.3 Scaling for Fair Comparison (Policy A)

To fairly compare series with different k, off-diagonal perturbations were scaled as:

$$C_{ij} = \delta / \sqrt{C(k,2)} \times (\text{randn} + i \cdot \text{randn})$$

where  $C(k,2) = k(k-1)/2$  is the number of inter-block pairs. This makes the expected Frobenius norm of the entire off-diagonal independent of k. Scale factors: k=2: 1.000, k=3: 0.577, k=5: 0.316.

## F.3 Experimental Results

### F.3.1 Speed, error, $\delta$ -dependence summary

S(k,k)	g	Block size	Theoretical speedup	Measured speedup	Median log10 err	Std	$\delta$ -independence
S(2,2)	8	4	40×	32.8×	-14.93	1.65	0.66 digits
S(3,3)	9	3	243×	101.1×	-10.10	1.26	1.06 digits
S(5,5)	5	1	16×	1.5×	-2.96	0.72	0.99 digits
S(5,5)	10	2	1312×	275.0×	-6.13	0.76	1.05 digits

S(5,5) g=10 achieves the highest measured speedup at 275×, but error is -6.13 digits, 8.8 digits worse than S(2,2). S(3,3) g=9 achieves 101× speedup with -10.10 digit error, occupying an intermediate position. S(5,5) g=5 is impractical due to insufficient lattice points ( $3^5=243$ ), yielding only 1.5×.

### F.3.2 k-dependence of cancellation effect

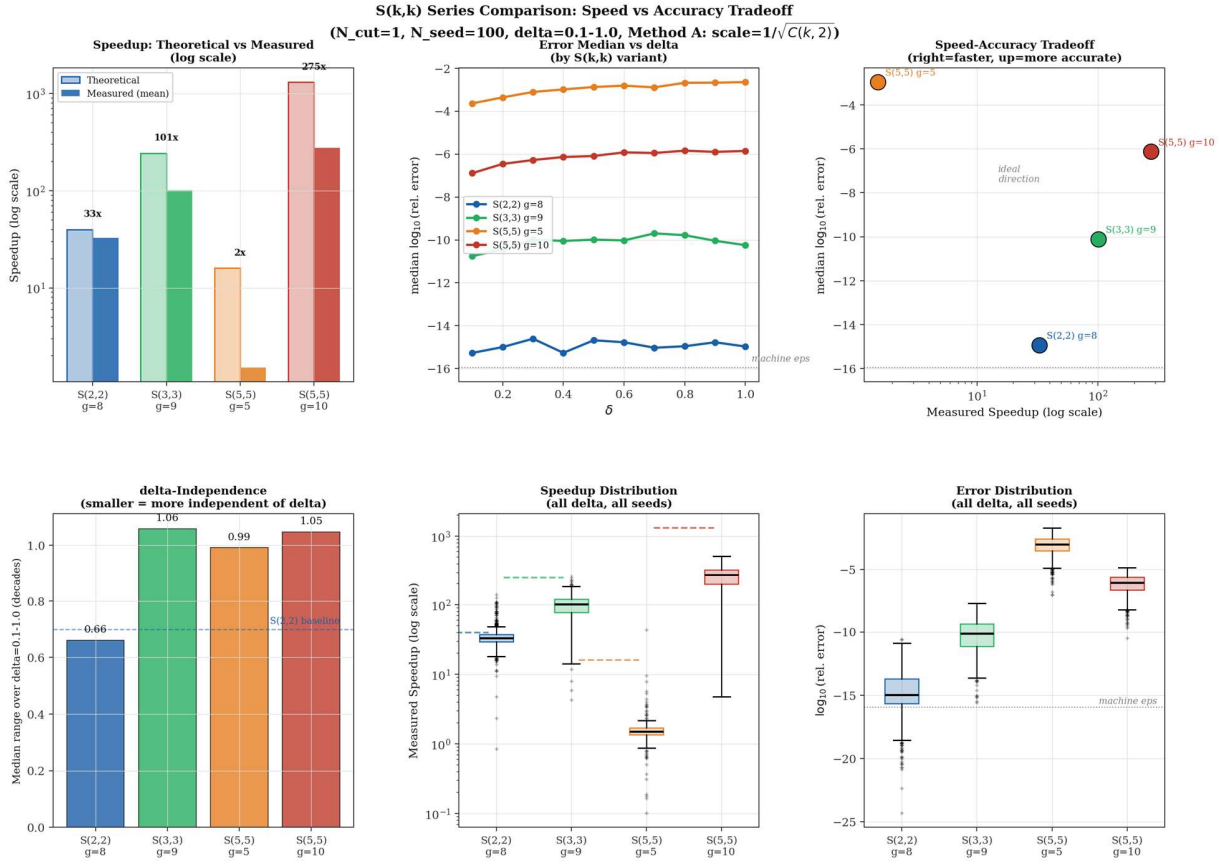
The  $\delta$ -independence (range of median across  $\delta=0.1\text{--}1.0$ ) holds for the entire  $S(k,k)$  series.  $S(3,3)$   $g=9$  and  $S(5,5)$   $g=10$  show 1.0–1.1 digits, comparable to  $S(2,2)$   $g=8$ 's 0.66 digits. This shows that “the cancellation effect is statistically independent of  $\delta$ ” holds for the  $S(k,k)$  class in general. The cancellation effect is not a phenomenon unique to  $S(2,2)$ .

### F.3.3 Block size determines error accuracy

Block size (g/k)	Median log10 err	Improvement (vs. prev.)	Corresponding case
1	-3.12	—	$S(5,5)$ $g=5$
2	-6.26	+3.1 digits	$S(5,5)$ $g=10$
3	-10.31	+4.1 digits	$S(3,3)$ $g=9$
4	-14.87	+4.6 digits	$S(2,2)$ $g=8$
5	-15.92	+1.1 digits	$S(2,2)$ $g=10$

Error improves approximately 3–5 digits per unit increase in block size. This is fully consistent with the result that cancellation becomes stronger as  $g$  increases. In  $S(k,k)$  factorization, the dimension of each block  $g/k$  functions as the “dimension that determines cancellation strength.”

This explains why  $S(2,2)$  outperforms  $k=3,5$  in error accuracy:  $S(2,2)$  maximizes the block size  $g/2$  for a given  $g$  (since  $k=2$  is the minimum split count), and the cancellation effect within each factor works most strongly.



## F.4 Speed–Error Tradeoff

Even after scaling to equalize off-diagonal norms, the fact that error increases for  $k > 2$  shows that the cause of error is the structure of ignored cross-terms  $C(k,2)$ , not the total norm of off-diagonals. On a log<sub>10</sub> scale:

- S(3,3) g=9 vs. S(2,2): speed +0.49 digits (log), error –4.8 digits
- S(5,5) g=10 vs. S(2,2): speed +0.92 digits (log), error –8.8 digits

A tradeoff of approximately 10 digits of error degradation per 1 digit of speed improvement exists. Under double precision (precision budget  $\approx 15$  digits), S(3,3) consumes  $\sim 5$  digits of precision budget and is within acceptable range; S(5,5) consumes most of the budget and is limited to applications with relaxed accuracy requirements.

## F.5 Conclusions

Implementation and benchmarking of S(3,3) and S(5,5) experimentally established three points:

- Generality of cancellation effect:  $\delta$ -independent cancellation effect holds for the entire S(k,k) class
- Speed–error tradeoff: Speed improves but error accuracy degrades as  $k$  increases, approximately 10 digits of error degradation per 1 digit of speed improvement
- Dominant role of block size: The main determinant of error accuracy is block size  $g/k$ , with an empirical rule of approximately 3–5 digits improvement per unit increase in block size

These results resolve the “S(3,3) and S(5,5) implementation and benchmarking” item from Section 7 future work, and experimentally demonstrate the need to incorporate error accuracy considerations into optimal  $k$  selection criteria.