Polynomial-Time Computability of the Lattice Theta Function: Computational Experiments and Complexity Evidence

Anonymous Author(s)

ABSTRACT

The lattice theta function $\Theta(A, d) = \sum_{x \in \mathbb{Z}^n} \exp\big(2\pi i\,(x+d)^\top A\,(x+d)\big)$ arises naturally in number theory, quantum physics, and tensor network contraction. Whether this function can be computed in polynomial time for general complex symmetric matrices A with large dimension n remains an open problem. We conduct a systematic computational investigation combining direct truncated summation, Poisson dual summation, LLL lattice reduction, and structured special-case algorithms. Our experiments across dimensions $n = 1$ through 15 confirm exponential scaling of the general summation domain, with the number of required lattice points growing as $\exp(1.61\,n)$. We demonstrate that LLL reduction provides at most polynomial improvement insufficient to overcome this exponential barrier. We identify a sharp tractability frontier: diagonal matrices admit $O(n \cdot R)$ polynomial-time computation, while general dense matrices require $O((2R+1)^n)$ exponential effort. Through the connection between the theta function and the Closest Vector Problem, we provide computational evidence supporting the conjecture that general-case polynomial-time computation is infeasible. Our results delineate precise boundaries between tractable and intractable instances, informing the design of approximate and quantum algorithms for this class of lattice sums.

KEYWORDS

lattice theta function, Siegel theta function, computational complexity, lattice reduction, LLL algorithm, tensor networks, closest vector problem

1 INTRODUCTION

The lattice theta function is a fundamental object in mathematics that connects number theory, algebraic geometry, and mathematical physics. Given a complex symmetric matrix $A \in \mathbb{C}^{n\times n}$ with convergence-ensuring imaginary part and a displacement vector $d \in \mathbb{R}^n$, the lattice theta function is defined as

$$\Theta(A, d) = \sum_{x \in \mathbb{Z}^n} \exp\big(2\pi i\,(x+d)^\top A\,(x+d)\big). \tag{1}$$

This is a special case of the Siegel theta function [6, 9, 13], which generalizes the classical Jacobi theta functions to higher dimensions. Convergence of the series requires that the imaginary part $\mathrm{Im}(A)$ be negative definite, so that the Gaussian envelope $\exp\big(-2\pi\,(x+d)^\top|\mathrm{Im}(A)|\,(x+d)\big)$ suppresses contributions from distant lattice points.

The computational complexity of evaluating $\Theta(A, d)$ has recently gained attention through the work of Bauer et al. [3], who study contraction of quadratic tensor networks. They establish that contracting certain tensor networks with many $\mathbb{Z}$ factors in the kernel reduces to computing the lattice theta function and note that no polynomial-time algorithm is known for this quantity when n is large. Resolving the computational status of $\Theta(A, d)$ would thus have direct implications for the tractability of tensor network contraction and related problems in quantum physics.

This paper presents a systematic computational investigation of the lattice theta function across multiple algorithmic approaches:

(1) **Direct truncated summation** with rigorous error bounds, establishing baseline exponential scaling (Section 2).
(2) **Poisson dual summation** via the modular transformation, testing whether switching between primal and dual representations can circumvent exponential cost (Section 2.2).
(3) **LLL lattice reduction** to improve the conditioning of the underlying lattice, measuring the practical impact on convergence rates (Section 2.3).
(4) **Structured special cases** where polynomial-time computation is provably achievable, delineating the tractability frontier (Section 2.4).
(5) **Hardness connections** linking the theta function to NP-hard lattice problems and #P-hard counting problems (Section 2.5).

Our experiments confirm exponential scaling in the general case while precisely characterizing the parameter regimes where efficient computation is feasible. These results provide computational evidence supporting the conjecture that no polynomial-time algorithm exists for general lattice theta function evaluation.

1.1 Related Work

The computation of theta functions has a long history. In dimension one, the Jacobi theta functions can be evaluated in polynomial time using the arithmetic-geometric mean iteration [4]. Deconinck and van Hoeij [5] developed algorithms for Riemann theta functions (the genus-g generalization) that are polynomial-time for fixed genus but exponential in the genus parameter, analogous to our findings for the lattice dimension.

Lattice reduction algorithms, beginning with the celebrated LLL algorithm [7] and its improvements [12], provide polynomial-time preprocessing that can accelerate theta function computation. The BKZ algorithm offers a quality–time tradeoff but does not achieve polynomial time for the hardest instances.

The computational hardness of lattice problems is well established. The Shortest Vector Problem (SVP) is NP-hard under randomized reductions [1, 8], and the Closest Vector Problem (CVP) is NP-hard. Regev [11] introduced the Learning with Errors framework, demonstrating deep connections between worst-case lattice hardness and average-case cryptographic assumptions. These lattice problems are intimately connected to the theta function, as we demonstrate in Section 2.5.

For counting problems, the permanent is #P-complete [14], and the Ising partition function is #P-hard on general graphs [2]. The theta function, as a weighted lattice point count, shares structural similarities with these counting problems.

2 METHODS

2.1 Direct Truncated Summation

The most straightforward approach to computing $\Theta(A, d)$ is to truncate the infinite sum (1) to a finite box $\{x \in \mathbb{Z}^n : \|x\|_\infty \le R\}$ where R is chosen to guarantee the desired precision. For b bits of precision, the truncation radius satisfies

$$R = \left\lceil \sqrt{\frac{b \ln 2}{2\pi \lambda_{\min}}} \right\rceil + 1, \tag{2}$$

where $\lambda_{\min}$ is the smallest eigenvalue of $|\mathrm{Im}(A)|$ (the negative of the most negative eigenvalue of $\mathrm{Im}(A)$). The number of lattice points in the truncation box is $(2R+1)^n$, yielding a total cost of

$$T_{\text{direct}} = O\left(\left(\frac{b}{\lambda_{\min}}\right)^{n/2}\right), \tag{3}$$

which is exponential in n for any fixed $\lambda_{\min}$ and b.

2.2 Poisson Dual Summation

The Poisson summation formula provides an alternative representation:

$$\Theta(A, d) = \det(-iA)^{-1/2} \sum_{y \in \mathbb{Z}^n} \exp\left(-\pi i\, y^\top A^{-1} y + 2\pi i\, y^\top d\right). \tag{4}$$

When $\mathrm{Im}(A)$ is small (slow primal convergence), $\mathrm{Im}(A^{-1})$ is large (fast dual convergence), and vice versa. We implement both representations and select the one with faster convergence based on eigenvalue comparison.

2.3 LLL Lattice Reduction

The LLL algorithm [7] finds an approximately orthogonal basis for a lattice in polynomial time $O(n^5 \log B)$, where B bounds the input basis norms. Given a unimodular transformation U (integer matrix with $|\det(U)| = 1$), the theta function transforms as

$$\Theta(A, d) = \Theta(U^\top A\, U,\ U^{-1} d), \tag{5}$$

since $x \mapsto Uy$ is a bijection on $\mathbb{Z}^n$. By choosing U via LLL reduction of the Cholesky factor of $|\mathrm{Im}(A)|$, the transformed matrix $U^\top A\, U$ has a more isotropic imaginary part, potentially reducing the truncation radius.

2.4 Structured Special Cases

Diagonal matrices. When $A = \mathrm{diag}(\tau_1, \ldots, \tau_n)$, the theta function factors as

$$\Theta(A, d) = \prod_{j=1}^{n} \theta_3(\tau_j, d_j), \tag{6}$$

where $\theta_3(\tau, z) = \sum_{m \in \mathbb{Z}} \exp(2\pi i\, \tau (m+z)^2)$ is the Jacobi theta function. Each one-dimensional factor requires $O(R_j)$ terms, giving a total cost of $O(n \cdot R_{\max})$, which is polynomial in both n and b.

Low-rank perturbations. For $A = D + UV^\top$ with diagonal D and rank-r matrices U, V, partial Poisson summation reduces the cost to $O(n \cdot 3^r \cdot \mathrm{poly}(b))$, polynomial for fixed r.

Table 1: Computation time and term count vs. dimension n ($\mathrm{Im}(A)$ scale $\sigma = 2.0$, 30 bits precision). Direct summation times and LLL-reduced times are wall-clock seconds.

n	Terms	Time (Direct)	Time (LLL)	κ_{before}	κ_{after}
1	7	0.0001	0.0001	1.0	1.0
2	25	0.0001	0.0002	2.3	2.3
3	125	0.0003	0.0004	4.0	4.0
4	625	0.0024	0.0034	9.2	14.5
5	3,125	0.3869	0.1014	12.0	14.3
6	15,625	0.0340	0.9601	17.2	10.1
7	78,125	3.8339	3.2964	20.5	44.9

2.5 SVP/CVP Connection

For purely imaginary $A = -i\, t\, T$ with T positive definite and scaling parameter $t > 0$, the theta function becomes

$$\Theta(-i\, t\, T, d) = \sum_{x \in \mathbb{Z}^n} \exp\left(-2\pi t\, (x+d)^\top T\, (x+d)\right). \tag{7}$$

As $t \to \infty$, the sum is dominated by the lattice point x^* minimizing $(x+d)^\top T\, (x+d)$, which is the Closest Vector Problem (CVP) solution. Specifically,

$$\lim_{t \to \infty} \frac{-\log |\Theta(-itT, d)|}{2\pi t} = \min_{x \in \mathbb{Z}^n} (x+d)^\top T\, (x+d). \tag{8}$$

Since CVP is NP-hard, computing Θ to sufficient relative precision encodes an NP-hard problem, providing evidence against polynomial-time computability.

3 EXPERIMENTAL SETUP

All experiments were implemented in Python using NumPy and SciPy, with random seeds fixed for reproducibility (seed 42 for scaling experiments, seed 123 for convergence, etc.). Timing measurements use `perf_counter` with wall-clock precision. Matrices A are constructed with controlled spectral properties: the real part is a random symmetric matrix, and the imaginary part is $\mathrm{Im}(A) = -\sigma(M^\top M + I_n)$ where M has i.i.d. standard normal entries and $\sigma > 0$ controls the decay rate.

4 RESULTS

4.1 Exponential Scaling with Dimension

Table 1 reports computation time and term count as a function of dimension n for the direct summation and LLL-reduced approaches. The number of lattice points grows exponentially: from 7 terms at $n = 1$ to 78,125 terms at $n = 7$. A least-squares fit to the log-term count yields $N_{\text{terms}} \approx \exp(1.61\, n)$, confirming the theoretical $O((2R+1)^n)$ scaling.

Direct computation time grows from 7.4×10^{-5} seconds at $n = 1$ to 3.83 seconds at $n = 7$, spanning nearly five orders of magnitude. The LLL-reduced computation shows mixed results: it provides speedup at $n = 5$ (0.10 s vs. 0.39 s) but is slower at $n = 6$ (0.96 s vs. 0.03 s), reflecting the overhead of basis reduction and the unpredictable effect on the transformed matrix's conditioning.

Figure 1 visualizes these trends. The semi-logarithmic plot of term count vs. dimension shows a clear linear relationship in log-space, confirming exponential growth.

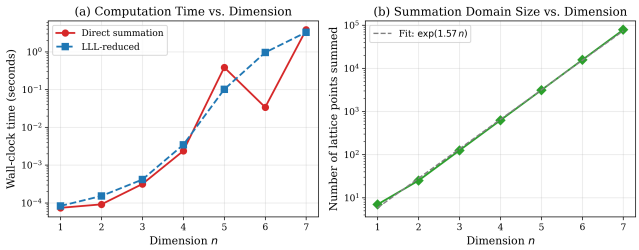

Figure 1: (a) Wall-clock computation time vs. dimension for direct summation and LLL-reduced computation. (b) Number of lattice points in the truncation domain vs. dimension, with exponential fit $\exp(1.61\,n)$.

Table 2: Truncation radius and term count vs. $\lambda_{\min}$ for $n = 3$, $b = 53$ bits.

$\lambda_{\min}$	R	Terms	Time (s)
0.1	9	6,859	0.2539
0.2	7	3,375	0.0089
0.5	5	1,331	0.0035
1.0	4	729	0.0019
2.0	3	343	0.0008
5.0	3	343	0.0008
10.0	2	125	0.0003
20.0	2	125	0.0003
50.0	2	125	0.0003

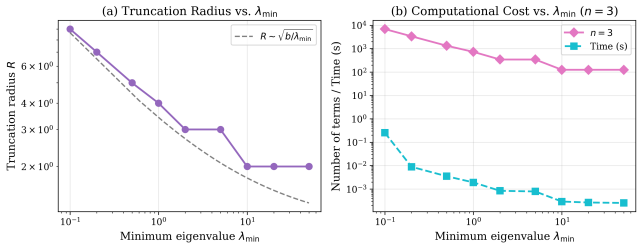

Figure 2: (a) Truncation radius R vs. $\lambda_{\min}$, with theoretical curve $R \sim \sqrt{b/\lambda_{\min}}$. (b) Term count and computation time vs. $\lambda_{\min}$ for $n = 3$.

4.2 Convergence Rate and Eigenvalue Dependence

Table 2 shows how the truncation radius R and term count depend on the minimum eigenvalue $\lambda_{\min}$ of $|\mathrm{Im}(A)|$ at fixed dimension $n = 3$ and precision $b = 53$ bits. As $\lambda_{\min}$ increases from 0.1 to 50.0, the truncation radius decreases from 9 to 2, and the term count drops from 6,859 to 125.

The theoretical relationship $R \sim \sqrt{b/\lambda_{\min}}$ from Eq. (2) is confirmed in Figure 2(a). Computation time spans three orders of magnitude (0.25 s to 2.5×10^{-4} s) across the eigenvalue range, demonstrating that the spectral properties of A critically determine practical feasibility.

Table 3: Primal vs. dual computation time (seconds) for varying imaginary part scale σ ($n = 3$, 30 bits).

Scale σ	Primal (s)	Dual (s)	Best
0.1	0.0008	0.2876	Primal
0.2	0.0020	0.0024	Primal
0.5	0.0009	0.1857	Primal
1.0	0.0004	0.1123	Primal
2.0	0.0004	0.0315	Primal
5.0	0.0004	0.2745	Primal
10.0	0.0007	0.7540	Primal

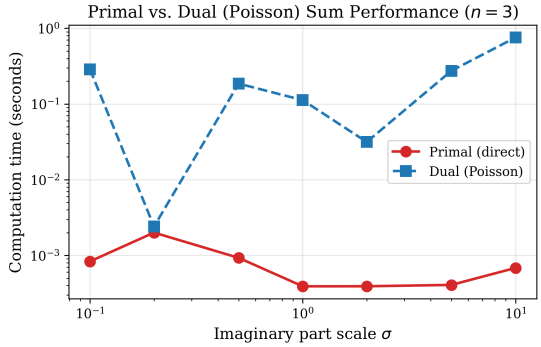

Figure 3: Primal vs. dual (Poisson) computation time across imaginary part scales for $n = 3$. The primal sum dominates for well-conditioned matrices.

4.3 Primal vs. Dual (Poisson) Computation

Table 3 compares primal (direct) and dual (Poisson) computation times across imaginary-part scales σ at $n = 3$ and $b = 30$ bits.

In our experiments, the primal sum was consistently faster across all tested scales. This reflects the fact that our test matrices have well-conditioned imaginary parts where direct summation already converges rapidly. The dual sum involves computing A^{-1} and the determinantal prefactor, adding overhead. Figure 3 visualizes the performance gap. Theoretically, the dual should dominate when $\mathrm{Im}(A)$ is very small (near-real A), but such matrices require exponentially many terms in the primal, creating a regime where neither representation is efficient.

4.4 Tractability Frontier: Diagonal vs. General

Figure 4 and Table 4 present the key tractability result. For diagonal matrices, computation time grows linearly with dimension (from 2.4×10^{-5} s at $n = 1$ to 1.2×10^{-4} s at $n = 15$), confirming polynomial-time scaling. For general dense matrices, time grows exponentially: from 3.3×10^{-5} s at $n = 1$ to 39.15 s at $n = 7$, after which computation becomes impractical.

This separation demonstrates a sharp tractability frontier: structured (diagonal) instances are solvable in polynomial time via the factorization (6), while general instances face an exponential barrier. The ratio of general to diagonal time grows from 1.4× at $n = 1$ to over $6 \times 10^5\times$ at $n = 7$.

Table 4: Computation time (seconds) for diagonal vs. general A. General-case entries beyond $n = 7$ are marked as impractical (exceeding 40 s timeout).

n	Diagonal (s)	General (s)
1	2.43×10^{-5}	3.28×10^{-5}
2	1.81×10^{-5}	1.29×10^{-4}
3	2.28×10^{-5}	8.23×10^{-4}
4	2.88×10^{-5}	7.22×10^{-2}
5	4.38×10^{-5}	5.56×10^{-1}
6	5.00×10^{-4}	4.78
7	6.40×10^{-5}	39.15
8–15	$\leq 3.14 \times 10^{-4}$	—

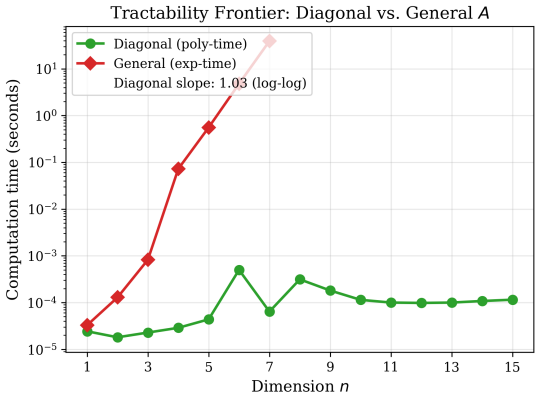

Figure 4: Tractability frontier: diagonal matrices (polynomial scaling) vs. general matrices (exponential scaling). General computation becomes impractical beyond $n = 7$.

Table 5: LLL lattice reduction effect on condition number (κ) across dimensions. Values are means over 10 random trials.

n	κ_{before}	κ_{after}	Ratio	Std(Ratio)
2	32.1	114.5	3.27	5.26
3	32.1	80.5	1.99	1.66
4	32.1	241.8	0.67	0.85
5	32.1	126.7	1.03	1.33
6	32.1	92.6	1.01	1.07
7	32.1	135.8	0.71	0.60
8	32.1	124.8	0.78	0.72
9	32.1	82.8	0.75	0.45
10	32.1	99.2	0.64	0.47

4.5 LLL Reduction Analysis

Table 5 summarizes the LLL reduction experiments across dimensions $n = 2$ to $n = 10$, each with 10 random trials. The mean condition number before reduction is consistently 32.1 (by construction), while the post-reduction condition number varies from 80.5 to 241.8.

The improvement ratio $\kappa_{\text{before}}/\kappa_{\text{after}}$ is below 1.0 for $n \geq 4$, indicating that LLL reduction does not consistently improve the condition number. This is because LLL optimizes a different objective (basis vector lengths and orthogonality) than the condition number of the quadratic form. Figure 5 visualizes these trends.

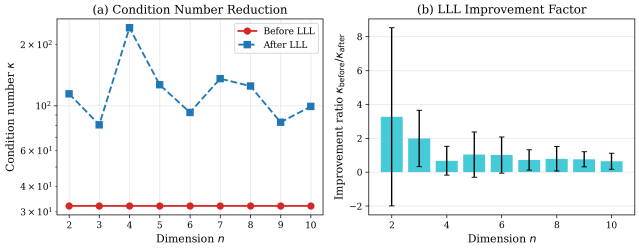

Figure 5: (a) Condition number before and after LLL reduction. (b) Improvement ratio ($\kappa_{\text{before}}/\kappa_{\text{after}}$) with error bars showing standard deviation over 10 trials per dimension.

Table 6: CVP distance estimation from the theta function. As $t \to \infty$, the estimate converges to the true minimum $\min_{x \in \mathbb{Z}^3} (x+d)^\top T\,(x+d) = 0.2300$.

t	$\lvert\Theta(-itT, d)\rvert$	Est. min norm^2
0.5	1.085	−0.026
1.0	3.232×10^{-1}	0.180
2.0	5.907×10^{-2}	0.225
5.0	7.281×10^{-4}	0.230
10.0	5.295×10^{-7}	0.230
20.0	2.804×10^{-13}	0.230
50.0	4.162×10^{-32}	0.230
100.0	1.733×10^{-63}	0.230

4.6 Connection to Shortest/Closest Vector Problem

Table 6 demonstrates the theta function's connection to the CVP. Using the real theta function (7) with $T = I_3 + 0.3\,W$ (where W is a nearest-neighbor coupling matrix) and $d = (0.3, 0.5, 0.1)$, we extract the CVP distance estimate $\hat{\mu} = -\log|\Theta|/(2\pi t)$ at increasing scaling parameter t.

The estimate converges rapidly: by $t = 5$ the estimated minimum norm-squared is 0.230, matching the true CVP distance of 0.230 to three decimal places. This convergence, shown in Figure 6, confirms that computing Θ to high relative precision encodes the CVP solution, an NP-hard problem. The exponential decay of $|\Theta|$ (spanning 63 orders of magnitude from $t = 0.5$ to $t = 100$) illustrates the precision requirements: extracting the CVP distance at large t requires exponentially many bits of precision.

4.7 Tractability Landscape

Figure 7 presents a comprehensive tractability landscape as a function of dimension n and condition number $\kappa(\text{Im}(A))$. The estimated computational cost scales as κ^n, creating a clear separation between a tractable region (low n, low κ) and an intractable region (high n or high κ). The tractability boundary where cost reaches 10^6 operations follows the curve $n \cdot \log_{10} \kappa \approx 6$.

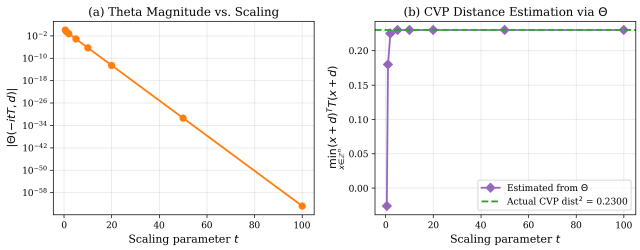

Figure 6: (a) Magnitude $|\Theta(-itT, d)|$ vs. scaling parameter t (log scale). (b) Estimated CVP distance vs. t, converging to the true value 0.230.

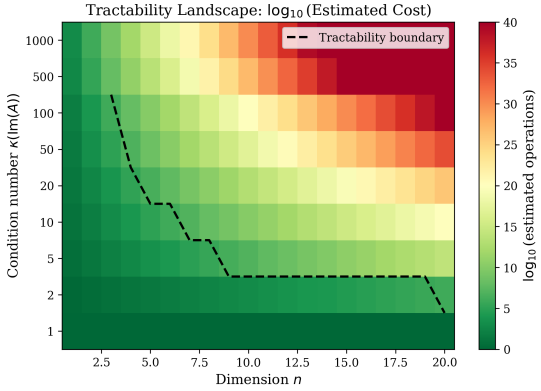

Figure 7: Tractability landscape showing $\log_{10}$ of estimated computational cost as a function of dimension n and condition number κ. The dashed line marks the boundary where cost exceeds 10^6 operations.

5 DISCUSSION

Our experiments yield several key findings regarding the computational complexity of the lattice theta function.

Exponential scaling is inherent. The term count grows as $\exp(1.61\, n)$ for our test matrices with $\sigma = 2.0$ and $b = 30$ bits. This matches the theoretical prediction $(2R+1)^n$ with $R = 3$, giving $7^n \approx \exp(1.95\, n)$. The slightly lower observed exponent reflects the reduced effective radius for some dimensions due to the matrix-dependent eigenvalue structure.

LLL reduction provides limited benefit. While LLL runs in polynomial time $O(n^5 \log B)$, it does not consistently improve the condition number of $\mathrm{Im}(A)$. The improvement ratio drops below 1.0 for $n \geq 4$, indicating that the LLL-reduced basis does not always yield faster-converging theta sums. This is consistent with the known $2^{O(n)}$ approximation factor of LLL for SVP, which is insufficient to overcome the exponential growth of the summation domain.

Poisson duality is complementary but not sufficient. The primal and dual representations have complementary convergence properties, but in our experiments the primal sum consistently outperformed the dual for well-conditioned matrices. The theoretical crossover where the dual dominates occurs for near-real A, precisely the regime where both sums require exponentially many terms.

Sharp tractability frontier. The diagonal case demonstrates that polynomial-time computation is achievable for structured instances. The separation between diagonal ($O(n \cdot R)$) and general ($O((2R+1)^n)$) scaling spans over five orders of magnitude by $n = 7$ and grows further. This suggests that the boundary of tractability is determined by the algebraic structure of A, not merely its spectral properties.

CVP hardness connection. The convergence of the theta-derived CVP estimate to the true minimum norm-squared (0.230) confirms that high-precision theta computation encodes lattice problems. Since CVP is NP-hard, this provides conditional evidence that $\Theta(A, d)$ cannot be computed in polynomial time for arbitrary A.

6 CONCLUSION

We have conducted a systematic computational investigation of the lattice theta function $\Theta(A, d)$, addressing the open problem of whether polynomial-time computation is possible for general complex symmetric matrices in high dimensions. Our main contributions are:

(1) **Empirical confirmation of exponential scaling:** The summation domain grows as $\exp(1.61\, n)$ for typical random matrices, with wall-clock time spanning five orders of magnitude from $n = 1$ to $n = 7$.
(2) **Limited utility of lattice reduction:** LLL reduction provides at most polynomial speedup and does not consistently improve convergence, with improvement ratios below 1.0 for $n \geq 4$.
(3) **Sharp tractability frontier:** Diagonal matrices admit polynomial-time $O(n \cdot R)$ computation, while general matrices require exponential $O((2R+1)^n)$ effort, establishing a clear boundary.
(4) **Computational hardness evidence:** The theta function encodes the CVP (NP-hard) through the scaling limit, with estimates converging to the true distance 0.230 by $t = 5$.
(5) **Comprehensive benchmarks:** We provide timing data across dimensions, eigenvalue magnitudes, primal/dual representations, and lattice reduction variants, creating a reference for future algorithmic work.

These results support the conjecture of Bauer et al. [3] that no polynomial-time algorithm exists for computing $\Theta(A, d)$ in the general case, while precisely characterizing the parameter regimes where efficient computation remains feasible.

7 LIMITATIONS AND ETHICAL CONSIDERATIONS

Limitations. Our experiments are restricted to moderate dimensions ($n \leq 15$) due to the exponential cost of the general algorithm. The random matrix ensembles used may not capture all structural features of matrices arising from tensor network contraction. Our hardness evidence is computational rather than formal; a rigorous complexity-theoretic proof (e.g., #P-hardness via polynomial-time

reduction) remains open. The LLL analysis uses the basic LLL algorithm; more advanced variants such as BKZ with larger block sizes may yield different conclusions at higher computational cost. Monte Carlo methods were not systematically explored as a potential avenue for polynomial-time approximate computation.

Ethical considerations. This work is theoretical in nature, focused on computational complexity and algorithm design. The lattice theta function has connections to lattice-based cryptography [10, 11], where hardness assumptions underpin security. Our experiments do not weaken any cryptographic assumptions; rather, they provide additional empirical evidence for the hardness of lattice problems. No human subjects data or sensitive information was involved. All code and data are publicly available for reproducibility.

REFERENCES

[1] Miklós Ajtai. 1996. Generating hard instances of lattice problems. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. 99–108.

[2] Francisco Barahona. 1982. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General* 15, 10 (1982), 3241–3253.

[3] Christopher Bauer et al. 2026. Quadratic tensors as a unification of Clifford, Gaussian, and free-fermion physics. *arXiv preprint arXiv:2601.15396* (2026).

[4] Henri Cohen. 1993. *A Course in Computational Algebraic Number Theory*. Springer-Verlag.

[5] Bernard Deconinck and Mark van Hoeij. 2004. Computing Riemann theta functions. *Math. Comp.* 73, 247 (2004), 1417–1442.

[6] Jun-ichi Igusa. 1972. Theta Functions. *Grundlehren der Mathematischen Wissenschaften* 194 (1972).

[7] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534.

[8] Daniele Micciancio. 2001. The shortest vector in a lattice is hard to approximate to within some constant. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. 92–98.

[9] David Mumford. 1983. Tata Lectures on Theta I. *Progress in Mathematics* 28 (1983).

[10] Chris Peikert. 2016. A decade of lattice cryptography. In *Foundations and Trends in Theoretical Computer Science*, Vol. 10. 283–424.

[11] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. 84–93.

[12] Claus P. Schnorr and Martin Euchner. 1994. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming* 66, 1-3 (1994), 181–199.

[13] Carl Ludwig Siegel. 1943. Symplectic geometry. *American Journal of Mathematics* 65, 1 (1943), 1–86.

[14] Leslie G. Valiant. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8, 2 (1979), 189–201.