HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

EFFICIENCY AND COMPUTATIONAL COMPLEXITY ANALYSIS OF DETERMINANT CALCULATION ALGORITHMS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS IN ENGINEERING PROBLEMS

T. A. Djalilova
Associate Professor, Andijan State Technical Institute,
Candidate of Physical and Mathematical Sciences

Abstract This paper investigates the efficiency of determinant calculation algorithms for solving systems of linear equations in engineering problems. The time and space complexities of Cramer's rule, Laplace expansion, and Gaussian elimination are analyzed. Based on precise operation counts, a drastic performance degradation of factorial algorithms under increasing matrix dimensions is demonstrated. Practical recommendations for computational optimization in engineering workflows are provided.

Keywords: determinant, Cramer's rule, Gaussian elimination, computational complexity, engineering models.

MUHANDISLIK MASALALARIDAGI CHIZIQLI TENGLAMALAR TIZIMINI YECHISHDA DETERMINANTLARNI HISOBLASH ALGORITMLARINING SAMARADORLIGI VA HISOBLASH MURAKKABLIGI TAHLILI.

Annotatsiya Ushbu maqolada muhandislik masalalarida chiziqli tenglamalar tizimini yechishda determinantlarni hisoblash algoritmlarining samaradorligi tadqiq etilgan. Kramers qoidasi, Laplas kengaytmasi va Gauss usullarining vaqt hamda xotira murakkabligi tahlil qilingan. Aniq hisob-kitoblar asosida, matritsa o‘lchami ortishi bilan faktorial o‘sishga ega algoritmlarning samaradorligi keskin pasayishi isbotlangan. Muhandislik amaliyotida hisoblash resurslarini optimallashtirish bo‘yicha amaliy tavsiyalar ishlab chiqilgan.

Kalit so‘zlar: determinant, Kramers qoidasi, Gauss usuli, hisoblash murakkabligi, muhandislik modellari.

ЭФФЕКТИВНОСТЬ И АНАЛИЗ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ АЛГОРИТМОВ ВЫЧИСЛЕНИЯ ДЕТЕРМИНАНТОВ ПРИ РЕШЕНИИ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ В ИНЖЕНЕРНЫХ ЗАДАЧАХ

Аннотация В статье исследуется эффективность алгоритмов вычисления детерминантов при решении систем линейных уравнений в инженерных задачах. Проведен анализ временной и пространственной сложности правила Крамера, разложения Лапласа и метода Гаусса. На основе точных расчетов количества арифметических операций показано резкое падение производительности факториальных алгоритмов при увеличении размерности матриц. Даны рекомендации по оптимизации вычислений для инженерной практики.

Ключевые слова: детерминант, правило Крамера, исключение Гаусса, вычислительная сложность, инженерные модели.

1. Introduction

The modeling, computation, and optimization of modern engineering problems inevitably converge into the challenge of solving high-dimensional systems of linear algebraic equations (SLAE). In structural mechanics, calculating the deformation of multi-support trusses; in electrical engineering,

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

determining current distribution in complex branched networks based on Kirchhoff's laws; and in aerodynamics or hydraulics, discretizing the Navier-Stokes differential equations using the finite difference method—all these real-world phenomena are structurally represented via linear matrix models of the form $A \cdot x = b$. In practical engineering applications, the dimension (n) of these matrices can range from dozens to several hundred thousand.

In classical linear algebra, Cramer's rule, along with the inverse matrix method, is taught as the most fundamental and theoretically elegant approach to solving SLAEs. Within the academic curriculum, particular emphasis is placed on calculating determinants using the Leibniz formula (permutations) or Laplace's recursive expansion along rows or columns. While these analytical approaches are highly convenient for finding exact solutions for low-dimensional ($n \leq 3$) matrices, they encounter severe algorithmic boundaries when applied to practical engineering computations.

The primary bottleneck stems from the fact that a linear increase in matrix dimension triggers a combinatorial or factorial explosion (O(n!)) in the total number of floating-point operations. Practicing engineers and students in higher technical institutions often rely heavily on the computational power of modern computer algebra systems (such as MATLAB, Wolfram Mathematica, or Python's SciPy library) without critically evaluating the asymptotic complexity of the underlying algorithm. Consequently, even for moderately sized systems (e.g., n = 20), computational processes can suffer from recursive lockups or stack overflow issues due to absolute memory consumption.

Therefore, to optimize algorithmic efficiency in engineering workflows, computing determinants and solving SLAEs must be shifted toward methods with polynomial complexity (O(n^3)), such as Gaussian elimination (triangular transformation) or LU decomposition.

The relevance of this study lies in quantifying the asymptotic limits of deterministic methods widely utilized in engineering simulations based on Floating-Point Operations (FLOPS) metrics. By providing rigorous mathematical comparative analysis of runtime and resource allocation across different algorithms, this paper establishes high-level pedagogical and practical recommendations for computational mathematics in engineering education and industrial design.

2. Methods

To systematically evaluate the efficiency of determinant calculation algorithms for solving systems of linear algebraic equations ($A \cdot x = b$, where $A \in R^{n \times n}$) in engineering contexts, this study employs a rigorous comparative-structural and asymptotic complexity analysis framework. The performance, resource allocation, and scalability of these algorithms are quantified using two primary metrics: asymptotic complexity (Big-O notation) and discrete Floating-Point Operations (FLOPS).

The methodological framework analyzes and evaluates four benchmark algorithmic architectures, ranging from analytical combinatorial structures to advanced numerical linear algebra methods:

2.1. The leibniz permutation formulation.

From a foundational algebraic perspective, the determinant of an n-dimensional matrix is defined as a static combinatorial sum over the symmetric group:

$$\det(A) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

where S_n represents the set of all n! permutations of degree n, and \operatorname{sgn}(\sigma) denotes the signature or parity of the permutation \sigma. This methodology examines the performance of the Leibniz formulation when mapped directly into software data structures, with a specific focus on the extreme random-access memory (RAM) overhead generated by permutation generation algorithms.

2.2. Recursive laplace expansion (method of minors).

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

The structural degradation of a determinant via row or column expansion is modeled as a highly recursive algorithm governed by the following formulation:

$$\det(A) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} M_{ij}$$

where M_{ij} is the $(n-1)\times(n-1)$ minor resulting from the elimination of the i-th row and j-th column. The methodology evaluates the structural impact of this recursive depth on standard modern execution environments (e.g., C++ or Python compilation engines). Particular attention is dedicated to analyzing how deep tree-structured recursions interact with hardware runtime boundaries, specifically focusing on CPU call-stack allocation and stack overflow thresholds.

2.3. Gaussian Elimination and LU Decomposition Architectures

The transformational approach involves utilizing elementary row operations (scalar row multiplication, row swapping, and row addition) to reduce the matrix A to an upper triangular form (U). Under this framework, the determinant is computed directly as the product of the resulting main diagonal elements:

$$\det(A) = \prod_{i=1}^{n} u_{ii}$$

To ensure structural fidelity against numerical instability, this methodological evaluation integrates partial and full pivoting strategies. The role of these pivoting mechanisms is critically assessed in terms of mitigating catastrophic cancellation and round-off error propagation within floating-point arithmetic.

2.4. Computational benchmarking metrics.

The operational viability of each algorithm within industrial and structural engineering simulations is evaluated against standardized benchmarking criteria:

Asymptotic Time Complexity: Modeling the mathematical growth rate of fundamental floating-point arithmetic operations (+, -,×,÷) strictly as a function of the input matrix dimension (n).

Hardware Unumdorlik Emulation (FLOPS Mapping): To bridge theoretical complexity with practical execution, the total required floating-point operations for each method are emulated against a standardized high-performance computing metric: 1 TFLOPS = 10^{12} floating-point operations per second.

The structural evaluation scales across dimension variations of n = 3, 10, 20, and 100, closely mirroring the matrix profiles typically encountered in finite element analysis (FEA) and computational fluid dynamics (CFD) frameworks.

3. Results

The mathematical and algorithmic evaluations conducted across the selected matrix dimensions (n {3, 10, 20, 100}) reveal a stark divergence between theoretical analytical methods and practical numerical algorithms. The empirical and theoretical performance metrics are categorized based on operational counts, asymptotic scaling behavior, and computational runtime emulations.

3.1. Asymptotic and structural operational complexity.

The exact total number of floating-point operations (FLOPS) required to evaluate a determinant varies drastically between the combinatorial/recursive models and the elimination models.

The Laplace Expansion requires a recursive tree structure where the total number of arithmetic operations f(n) asymptotically approaches a factorial growth rate:

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

$$f(n)=\sum_{k=1}^{n-1}\frac{n!}{k!}\approx e\cdot n!$$

where e ≈ 2.71828. This recursive branch multiplication scales independently of hardware optimizations.

Gaussian Elimination requires systematic systematic triangularization. The total operational count for a forward-elimination determinant pass is bounded by polynomial growth:

$$W(n)=\frac{2}{3}n^3+\frac{1}{2}n^2-\frac{7}{6}n\approx\frac{2}{3}n^3$$

Table 1 provides a comprehensive comparative matrix mapping the exact floating-point operations across varying dimensional scales.

Table 1. Quantitative FLOPS requirements across different algorithmic architectures.

Matrix Dimension (n×n)	**Laplace / Leibniz Expansion (e· n!)**	**Gaussian Elimination (32 n3)**	**Algoritm Efficiency Ratio**
n = 3	16 FLOPS	18 FLOPS	1.12 (Laplace marginal edge)
n = 10	9.86 \times 10^6 FLOPS	667 FLOPS	1.47 \times 10^4 (Gauss advantage)
n = 20	6.61 \times 10^{18} FLOPS	5,333 FLOPS	1.23 \times 10^{15} (Gauss advantage)
n = 100	2.53 \times 10^{158} FLOPS	6.67 \times 10^5 FLOPS	\infty (Combinatorial explosion)

3.2. Hardware execution emulation and runtime latency.

To evaluate the operational viability of these algorithms within real-world engineering environments, execution runtimes were emulated assuming a standard high-performance computational baseline operating at 1 TFLOPS = 10^{12} floating-point operations per second).

A) System scaling under laplace expansion and cramer's rule.

When applying Cramer's rule to solve an engineering system of dimension n=20 (e.g., a standard multi-joint truss optimization problem), the solver must evaluate n+1 = 21 separate determinants.

Using the Laplace recursive minor algorithm, the core computation time (T) for a single determinant is:

$$T_{\text{single}}=\frac{6.61\times10^{18}\text{ FLOPS}}{10^{12}\text{ FLOPS/sec}}=6.61\times10^{6}\text{ seconds}\approx76.5\text{ days}$$

To complete the full Cramer's sequence for all 20 unknowns, the total runtime evaluates to:

$$T_{\text{total}}=21\times76.5\text{ days}\approx1{,}606.5\text{ days}\approx4.4\text{ years}$$

As n scales to 100, the calculation timeline reaches 2.53×10^{146} seconds, a duration exceeding cosmological lifespans, rendering the algorithm fundamentally non-executable.

B) System scaling under gaussian elimination.

Conversely, executing the same structural matrix (n=20) via Gaussian triangularization demands a total operational runtime calculated as:

$$T_{\text{Gauss}}=\frac{5{,}333\text{ FLOPS}}{10^{12}\text{ FLOPS/sec}}=5.33\times10^{-9}\text{ seconds}=5.33\text{ nanoseconds}$$

Even when resolving highly complex engineering meshes with an allocation of n=100, the numerical pipeline completes the operation seamlessly:

$$T_{\text{Gauss}}=\frac{6.67\times10^{5}\text{ FLOPS}}{10^{12}\text{ FLOPS/sec}}=6.67\times10^{-7}\text{ seconds}=0.667\text{ microseconds}$$

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

These analytical metrics clearly showcase the structural demarcation line where classical analytical strategies fail completely, and polynomial numerical execution architectures become mandatory for industrial engineering computation.

4. Discussion

The analytical and quantitative metrics derived in this study clearly expose the profound dichotomy between abstract mathematical elegance and pragmatic computational engineering. While Leibniz's permutation formula and Laplace's recursive minor expansion serve as foundational pillars in pedagogical linear algebra, their application within engineering frameworks is strictly bounded by the phenomenon of combinatorial explosion. As demonstrated in the results, transitioning from n=3 to n=10 shifts the efficiency ratio by four orders of magnitude in favor of Gaussian elimination, a divergence that becomes infinite for all practical purposes at n \ge 20.

In modern industrial paradigms—such as modeling the tribological wear kinetics of machinery components, analyzing non-linear structural deformations via Finite Element Analysis (FEA), or resolving fluid dynamics via Computational Fluid Dynamics (CFD)—matrix profiles routinely exceed dimensions of $n = 10^3$ to $n = 10^6$. In such domains, relying on determinant-dependent strategies like Cramer's rule is structurally impossible.

However, substituting recursive models with polynomial elimination architectures like Gaussian elimination ($O(n^3)$) is only the first step in optimization. When transitioning these algorithms into production-grade engineering code, researchers must evaluate several systemic constraints:

4.1. Numerical stability and round-off errors.

Gaussian elimination is highly susceptible to floating-point precision degradation. When dividing by an exceptionally small diagonal element (a near-zero pivot), catastrophic cancellation occurs, leading to massive round-off errors that propagate through subsequent row operations. To mitigate this in precision-critical engineering applications, the implementation of Partial Pivoting (row-swapping to ensure the largest absolute value occupies the pivot position) or Full Pivoting is mandatory. This addition preserves numerical fidelity but introduces a slight sorting overhead.

4.2. Exploiting matrix topology (sparse vs. dense matrices).

In structural mechanics and thermal network analysis, the resulting matrices are rarely dense. Instead, they exhibit specific topologies such as banded, symmetric positive-definite, or highly sparse structures (where more than 95% of the elements are zero). Direct application of standard Gaussian elimination to these structures causes a phenomenon known as fill-in, where previously zero elements become non-zero during elimination, destroying the matrix sparsity and inflating memory consumption. For these specialized engineering matrices, alternative modern paradigms are significantly superior:

Matrix Decompositions: Utilizing LU, QR, or Cholesky decompositions (A = LL^T), which isolate structural components and allow rapid back-substitution.

Iterative Solvers: Transitioning from direct elimination to stationary iterative methods (Gauss-Seidel, SOR) or Krylov subspace methods (Conjugate Gradient, GMRES) for sparse systems. These techniques bypass determinant calculations entirely and can reduce computational complexity from $O(n^3)$ down to approximately O(n) per iteration.

4.3. Implications for engineering pedagogics.

The findings of this paper highlight a critical pedagogical gap in higher technical education. Often, engineering curricula heavily weight the manual calculation of determinants and the application of Cramer's rule. While useful for building initial mathematical intuition, this emphasis can create a false impression regarding computational scalability. Academic frameworks must pivot toward

HSR (LONDON),
HOUGHTON STREET REVIEW JOURNAL
ISSN: 2058-5519|Impact factor-15.9
http://houghtonstreetreview.com/index.php/hsr

teaching computational complexity, algorithmic efficiency, and numerical stability alongside classical theory. This ensures that future independent researchers and engineers design algorithms optimized for contemporary multi-core CPUs and high-performance computing (HPC) clusters.

5. Conclusion

1. Academic and Pedagogical Boundaries: Cramer's rule and its underlying recursive determinant calculation architectures (such as Laplace expansion and Leibniz permutations) possess strictly theoretical and pedagogical value. Their application must be confined to low-dimensional analytical contexts ($n \leq 3$) to build initial mathematical intuition within engineering curricula.

2. Algorithmic Mandate for Engineering Simulations: In the context of industrial computational physics, finite element modeling, and mechanical optimization, numerical pipelines must strictly avoid recursive determinant dependencies. To prevent computational collapse driven by combinatorial complexity (O(n!)), the adoption of polynomial-time architectures ($O(n^3)$) such as Gaussian elimination, LU decomposition, or Cholesky factorization is an absolute technical requirement.

3. Strategic Pivot Toward Sparse Matrix Paradigms: For advanced high-dimensional engineering systems ($n \geq 10^3$), direct elimination methods should be strategically superseded by iterative schemes (e.g., Krylov subspace methods, Conjugate Gradient, and GMRES). Exploiting matrix topologies and sparsity patterns reduces computational and memory footprints toward linear scaling trajectories (O(n)), providing the requisite scalability for real-time cyber-physical systems and multi-physics simulations.

References

1. Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations* (4th ed.). Johns Hopkins University Press.
2. Strang, G. (2016). *Introduction to Linear Algebra* (5th ed.). Wellesley-Cambridge Press.
3. Quarteroni, A., Sacco, R., & Saleri, F. (2007). *Numerical Mathematics* (2nd ed.). Springer Science & Business Media. https://doi.org/10.1007/b138611
4. Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM). https://doi.org/10.1137/1.9781611971446
5. Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems* (2nd ed.). Society for Industrial and Applied Mathematics.
6. Trefethen, L. N., & Bau, D. (1997). *Numerical Linear Algebra*. SIAM.
7. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). Cambridge University Press.
8. Knuth, D. E. (2011). *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3rd ed.). Addison-Wesley Professional.