学位論文　博士（工学）

分子シミュレーションに特化した効率的な乱数生成手法の提案と深層学習を用いた乱数生成の有効性の検討

2023 年度

慶應義塾大学大学院 理工学研究科

岡田清志郎

論文要旨

本研究は分子シミュレーションに利用する効率的な乱数生成手法の開発に焦点を当てたものである．日々洗練されていく数値計算手法と計算機性能の向上の交差点において，分子シミュレーション技術の進化が目覚ましい．乱数を利用した粗視化手法によって，メソスケールにおける現象の取り扱いがより容易になり，また，統計力学に基づく確率分布をサンプリングすることにより，効率的に対象とする系の統計量を得ることが可能となった．乱数は効率的な数値計算手法を確立する上で重要な役割を果たしている．計算機における乱数生成手法は今日まで多くの研究がなされてきたが，乱数の性質と計算コストのバランスを改良する余地は残されており，計算コストの観点から乱数生成手法そのものが問題視されることも少なくない．そこで本研究では，各数値計算手法に特化した乱数生成手法を提案し，従来以上の計算効率を達成する乱数生成手法を提案した．まずメソスケールを取り扱う粗視化手法の代表的な例として，散逸粒子動力学 (DPD) 法に着目した．DPD 法を用いた大規模並列計算をする際には，暗号学的ハッシュ関数を用いた乱数生成手法が利用されていたが，計算コストが大きいことが課題であった．そこで，乱数シード値に，より複雑な bit 列を持つ数値を利用することで，乱数生成に必要なハッシュ関数の軽量化を達成した．更に，DPD 計算の過程で常に生成される数値の乱雑性を利用することで，従来のハッシュ関数の約 62 倍の速度で乱数を生成することに成功した．生成される乱数が一般的に利用される NIST 乱数検定に合格することを示した後，この手法を利用することによる計算精度の変化を複数の観点から評価し，精度に影響なく計算が実行できることを示した．また，この「数値計算過程で副次的に生成される数値の乱雑性」を利用することによる乱数生成手法をモンテカルロ (MC) 法にも適用した．乱数を多用する MC 法において，乱数生成のためだけの計算を都度実行するのではなく，MC 計算そのものの過程で生じる数値の乱雑性を利用し，乱数生成することで，極めて少ない演算で良質な乱数を生成できることを示した．このように計算過程で生じる乱雑性を利用し，各計算手法に特化した乱数生成手法を開発することで，従来以上の計算効率を実現できることを示した．一方で，様々な計算手法に特化した乱数生成手法を開発することは容易ではなく，専門家の高度な知見に基づく開発が必須である．そこで，本研究では深層学習を利用したデータ駆動型の乱数生成手法を実現することで，将来的な新規乱数生成アルゴリズム開発に誰もが取り組み，個別の目的に特化したアルゴリズムを開発できるようになることを目指した．ここでは，特に敵対的生成ネットワークを利用し，乱数性質が低い数値列を，NIST 乱数検定の基準に合格する数値列に変換するモデルの開発に取り組み，深層学習を用いた乱数生成手法の開発に初めて成功した．また同時に，学習において，無限の学習データを取り扱う際における過学習の可能性に言及し，深層学習の利用に貢献する知見を提供した．本研究で示した知見は，計算機シミュレーションにおける効率的な乱数生成手法開発から，将来の深層学習を利用したデータ駆動型の新規乱数生成手法の開発に貢献するものである．

Thesis Abstract

This study focuses on the development of efficient random number generation techniques tailored for molecular simulations. At the nexus of advancing computer science and evolving numerical methods, molecular simulation techniques have seen remarkable progress. Notably, the employment of random numbers in coarse-graining methods has facilitated handling phenomena at the mesoscale. Furthermore, by sampling probability distributions grounded in statistical mechanics, it is now feasible to efficiently derive statistical metrics from target systems. Random numbers, thus, have come to play an indispensable role in crafting efficient numerical methodologies. While a plethora of research has been devoted to random number generation in computational environments, room for refinement persists. From a computational cost perspective, the very methods of random number generation have been frequently challenged. Addressing this, we advocate for random number generation techniques specialized for each numerical computation methodology, aiming to surpass traditional computational efficiencies. Our initial focus was on the dissipative particle dynamics method, a quintessential coarse-graining method for the mesoscale. Historically, cryptographic hash functions were employed for random number generation in large-scale parallel computations using this method. Yet, the significant computational cost posed a challenge. Innovations in random number seed values enabled us to streamline the hash function. Moreover, by leveraging the intrinsic randomness perpetually produced during numerical computations, we achieved a random number generation speed approximately 62 times faster than traditional hash functions. After confirming the generated random numbers passed the NIST standard tests, we evaluated the impact on computational accuracy, ensuring unaffected performance. Additionally, we adapted our random number generation approach, leveraging computational randomness, for Monte Carlo (MC) simulations. In MC, known for its high random number utilization, we showcased the generation of quality random numbers using minimal operations by harnessing computation-induced randomness and updating a stored bit sequence in memory. Developing techniques specialized for various computation methodologies is undeniably intricate, necessitating expert insights. Hence, this study aimed to enable widespread participation in the development of novel random number generation algorithms, using deep learning-driven approaches, allowing individuals to create algorithms tailored to specific objectives. We embarked on developing a model using generative adversarial networks, successfully converting number sequences with low randomness properties to sequences compliant with NIST random number test standards, marking the pioneering success in random number algorithm development using deep learning models. Concurrently, we addressed the potential for overfitting when dealing with infinite training data in learning and furnished insights beneficial for deep learning applications. The insights presented in this research not only contribute to the efficient random number generation for computer simulations but also pave the way for future data-driven random number algorithm development leveraging deep learning.

目次

第 1 章

序論

1.1　本研究の位置付けと目的

1.1.1　本研究の産業的位置づけと意義

現代社会は，技術革新の波が押し寄せる時代となり，その速度は著しく，人類史に類を見ない．その速度は，21 世紀始まりの年に Ray Kurzweil が提案した “The Law of Accelerating Returns” に現状で従っていると言われている [1–3]．この法則の厳密性について議論の余地は多く残されているが，GPT4 モデルを始めとする新しい深層モデル [4–7] の卓越した性能など，対象領域の専門家の予想をも上回る技術が実際に実現している状態である．この技術革新は，個々の分野のみの技術進歩を超え，経済力や社会的影響力の源泉となっている．そして，この全体を牽引する大きな存在の一つが計算機の性能向上である．ソフトウェアの進化，AI の発展，データサイエンスの普及，これら一連の動向はすべて日々性能向上する計算機によって推進されている．計算機は私たちの生活における必需品となり，時には全体の経済や社会を動かす力を持つツールともなっている．その証左として，半導体企業である NVIDIA や TSMC の時価総額が急速に上昇している事実が挙げられる．これらの企業は，強大な計算力を有する製品を提供することにより，彼らが持つ経済的，社会的な力は増大し続けている．この計算力の増大は，さらに広範な分野に適用可能となり，その結果，より高度で効率的な計算手法が求められるようになっている．計算機シミュレーションはその先端に位置しており，物理現象の解析から新素材の設計，さらには薬物開発に至るまで，多様な分野でその利用が広がっている．

このように拍車がかかる情報化社会で計算機を利用する際に，必要不可欠な技術のうちの一つは乱数生成技術である．まず，乱数生成技術は，暗号技術において欠かすことのできない要素である．乱数を用いて生成される暗号鍵は，通信の安全性を保証する上で必須であり，またブロックチェーン技術のような新しい分野でも乱数が重要な役割を果たす．それだけでなく，現代の計算科学技術の多くの分野で重要な役割を果たしている．シミュレーションを用いた確率的な現象のモデル化から未知の統計量の探索など利用範囲は広い．自然現象はしばしば確率的に振る舞う．気象学の予測から量子力学に至るまで，自然の振る舞いを正確に予測するためには確率的な手法が必要である．乱数は，そのような現象を模擬するための基本的なツールとして利用される．たとえば，気象モデルでは，乱数を用いて気候の不確定性を反映させ，より現実に近い予測を行うことが可能である．乱数はまた，未知の可能性を探求するツールとしても使われる．例えば，モ

ンテカルロ (Monte Carlo: MC) 法 [8,9] という手法は，確率的なシミュレーションを行うために乱数を用いており，物理学から金融工学まで，広範な分野で利用されている [10–13]．乱数を用いることで，複雑な問題に対する近似的な解を効率的に求めることができる．また，新しい解の可能性を探求するためにも乱数が活用される．具体的には，新しい薬物の探索や新素材の設計などにおいて，乱数を用いた探索アルゴリズムが使われることが多い．

しかしながら，このように広範な分野で利用される乱数生成には，一つの大きな課題が存在する．それは，乱数生成に高い計算コストがかかるという問題である．乱数を生成するプロセスは，複雑な計算を伴い，大量の乱数を短時間で生成するためには高度な計算能力が求められる [14]．また，乱数が真にランダムであることを保証するためにも，その生成過程には大きな計算資源が必要となる．この問題は，特にエネルギー効率を重視する現代の計算環境において，より深刻な課題となっている．計算リソースは有限であり，それを効率的に利用することは，持続可能な社会を実現するための重要な要素である．それだけに，乱数生成の計算コストの削減は，我々が直面する重要な課題となっている．さらには，現代社会はデータ駆動型の社会であり，大量のデータを効率的に処理するためには，大量の乱数を生成する能力が必要となる．乱数の生成速度が計算のボトルネックとなり，全体の計算効率を下げる可能性もある．これは，計算を利用して新たな価値を生み出す現代社会においては，大きな制約となる．したがって，乱数生成の効率化とその計算コストの削減は，現代社会における技術革新の一環として，重要な課題となっている．この課題を解決し，乱数生成が効率的に行えるようになれば，その結果，計算機によって取り扱える幅はさらに広がる．乱数の生成が一層効率化されれば，その影響は省電力による環境への負担の軽減だけでなく，高速な計算の実現による物理科学から経済科学，さらには暗号技術に至るまで，多くの領域に影響すると考えられる．

以上のことから，乱数生成の効率化とその計算コストの削減は，科学技術の発展を推進し，持続可能な社会を実現する上で，重要な課題であると言える．本研究は，特に分子シミュレーションを実行する際の，効率的な乱数生成手法の開発に焦点をあてたものである．

1.1.2 研究課題の定義と目的

コンピュータシミュレーションは一般的に材料の物性を予測するために利用されている．材料の物性を精度良く近似する理論がない場合，もしくは解くことが困難である場合，シミュレーションによる解析が有効である．原子，分子の挙動が大きな影響力をもつナノスケールの物性解析では，特に分子シミュレーションが利用されている．この手法の登場により，実験的には観測することのできないスケールの現象を精度良く予測することが可能になってきた．上述したように，近年の計算機科学の発達を受けて，分子シミュレーションで扱える現象が増え，薬品設計，材料科学，生物物理学などの分野での応用が拡大している．

分子シミュレーションは分子動力学 (Molecular Dynamics: MD) 法 [15,16] と，MC 法 [8,9] の 2 つに大別される．1 つ目の MD 法では，対象とする原子や分子について，相互作用する粒子系についてニュートンの運動方程式を数値的に解くことにより時間発展を追跡し，古典的な多体系の構造的，動的，及び熱力学的など様々な物性値を予測することができる．系を構成するすべての粒子座標と速度を詳細に追跡することができるため，実験では観測することのできない物理現象の解析が可能である．一方で，シミュレーションで扱える粒子の個数の上限は，常に計算資

源の上限によって制限される．したがって，観測の対象を決定し，得たい物性値が得られるような計算方法を選択することが重要である．たとえば，数十，数百 nm といったメゾスケールでの解析に関心がある場合，個々の原子の詳細な運動ではなく，分子群で構成される集団の挙動に着目することが多い．そこで，それぞれの原子の微視的な挙動の詳細をを集約し，系を取り扱う方法がある．空間的，時間的に比較的大きな現象を取り扱う方法の一つに粗視化手法がある．粗視化手法を用いると，複数の粒子をひとまとめにしてみなすため，計算で取り扱う必要のある計算点を大幅に削減することが可能である．この方法により大きな系のシミュレーションをより小さい計算コストで取り扱うことができる．このような粗視化手法はいくつか提案されているが，本研究では，その中でも特に，熱力学的な挙動を，乱数を利用して疑似的に再現する散逸粒子動力学 (Dissipative Particle Dynamics: DPD) 法 [17–19] を対象とする．この方法では，粒子間に作用する相互作用に乱数を利用して計算するランダム力が利用される．このランダム力は粒子ペアに対して等しい値が与えられ，一連のシミュレーションで大量の乱数生成を必要とする．したがって DPD 法を利用したシミュレーションでは，乱数生成の効率が，シミュレーションの計算コストに大きな影響を及ぼす．また，2 つ目の MC 法は，確率分布のサンプリングを利用して，対象とする物理現象や数学的問題の近似解を得ようとする手法である．この方法は MD 法と異なり，多体系の時間発展を再現しない．つまり，系の輸送特性などの動的な性質の解析は行わない手法である．この方法の利点は，系の時間発展を取り扱わないため，その系の時間発展の速度や仕方に依存せず，比較的効率的に系の取りうる複数の状態をサンプリングすることが可能である．したがって，MC 法は平衡状態の取得のために MD 法よりも有利であると考えられる．このような MC 法の計算にも，大量の乱数が必要となる．これらの手法において，乱数生成に必要な計算コストを削減することは大きな意義を有する．様々な現象を取り扱うこれらの手法に，従来よりも効率的な乱数生成手法を提案することで，これまでにない効率でシミュレーションを実行することができる．本研究では，この乱数生成手法の効率に着目し，DPD 法と MC 法専用の特化型乱数生成手法を提案する．これにより，従来よりも効率的なシミュレーションを実現する．更に，本研究の後半で，深層学習を活用した乱数生成手法を提案する．これにより，DPD 法と MC 法のみにとらわれず，様々な手法に特化した乱数生成手法の開発を支援する，データ駆動型の乱数生成手法の実現に貢献する．これらの研究は今後の効率的な特化型乱数生成手法の開発に貢献するものである．

1.2 提案する 3 つの新しい乱数生成手法

本研究では，大きく分けて 3 つの新しい乱数生成手法を開発した．1 つ目は，DPD 法 [17–19] に用いる乱数生成手法の開発である．DPD 法は数百 nm，数十 μs のスケールでの動的性質の解析が可能であり，相互作用する力の計算に大量の乱数を利用する手法である．大規模なシミュレーションを実行するために並列計算を行う際には，相互作用する粒子ペアに等しい乱数を与えるため，暗号学的ハッシュ関数を利用した乱数生成手法を利用する [20]．この場合，Mersenne Twister (MT) 法 [21] のような一般的な乱数生成手法用いる場合に比べて，乱数生成に必要な計算コストが大きくなる．そこで，暗号学的ハッシュ関数を軽量化し，粒子座標を乱数シードとして高品質な乱数を生成する方法を提案した．具体的には，Tiny Encryption Algorithm (TEA) 法 [22] を用いて，計算コストを低減しつつ，乱数の品質を維持する方法を検討した．計算コスト

を低減しながらも，乱数の品質を維持する方法を検討する過程で，シミュレーションの実行過程で副次的に生じる複雑な数値列を初めて発見した．この発見は，シミュレーションにより出力される物性値を反映した数値列が，予期せぬ高いレベルの乱雑さを有していたことを示唆している．そこで，本研究では，この複雑な数値列を効率的に抽出し，再利用する方法を開発した．具体的には，シミュレーションの各ステップで生成されるデータから，乱数としての要件を満たす数値列を選出するアルゴリズムを設計した．このアルゴリズムにより，従来の乱数生成手法に比べて，計算コストを大幅に削減しながらも，乱数の品質を損なうことなく，高速な乱数生成を実現することが可能となった．さらに，提案手法の乱数品質を評価するため，NIST 乱数検定を使用し，安定して性質の良い乱数を生成し続けられるかどうかを評価した．結果として，提案手法は従来の乱数生成法に比べて，1/62 の計算コストで高品質な乱数を生成できることが明らかとなった．シミュレーションの計算過程で生じる乱雑性を再利用することで，効率的な乱数生成を達成する本手法は，DPD シミュレーションに限らず，様々な数値シミュレーションや大規模な並列計算環境での利用が有望である．また，乱数生成の周期は系を構成する粒子の状態数に依存し，システムサイズに比例してスケールするため，将来的に巨大な系を取り扱った場合でも，十分な周期を持つため，高いスケーラビリティを有する．結論として，本研究により，DPD 法における乱数生成の計算コストを削減することができ，スーパーコンピュータ等の超並列アーキテクチャでの高速なメソスケールの粗視化シミュレーションを実現することが期待される．

2 つ目に，DPD 法のための乱数生成手法を応用し，一般的な MC 法に利用する乱数生成手法を開発した．MC 法は，乱数生成の際に 1 つのステップで多くの乱数を要するため，計算過程で乱雑性のある数値列が生成されていたとしても，再利用できる回数に制限がある．このように，1 つのステップで使用可能な乱雑性のある数値の個数が限られているため，新たな方法を考案した．具体的アルゴリズムは後述するが，xor や shift 等の基本的な bit 演算と，計算過程で生成される乱雑性のある数値列を利用し，複数の乱数を生成する方法を開発した．この結果，従来の MC 法のアルゴリズムに 7 点の変更を適用し，乱数の初期化，乱数生成，粒子選択，試行移動量の生成，エネルギー計算，及び Metropolis-Hastings 法 [8] の各ステップにおいて，乱数を生成または更新する手法を取り入れた．この新しい方法を採用した結果，従来の MT 法に比べて，1/5 の計算コストで乱数を生成できることが明らかとなった．これは計算コストの大幅な削減と MC シミュレーションの精度の維持を可能にする．結論として，本研究で提案する新たな乱数生成手法は，MC シミュレーションの効率化に寄与し，計算速度の向上と精度の維持を実現した．この手法は，今後の MC 法の効率的な計算手法の開発に貢献するものである．

MC 法をはじめとした，特定のシミュレーション手法に対して効率的な乱数生成手法の適用が可能であることが示唆された．この乱数生成手法の開発は，シミュレーションの精度と効率性の向上に重要である．しかし，本研究で示したような各シミュレーションに特化した乱数を都度開発することは，労力と時間がかかるうえ，専門家の知識が必要である．そこで 3 つ目の研究として，深層学習を用いたデータ駆動型の乱数生成手法の開発に取り組んだ．深層学習の技術を利用すると，開発者の指示なしに複雑な統計的特徴を自動的に抽出することができる．これは，画像認識など多くの分野で既に活用されている技術である．本研究では，敵対的生成ネットワークを用いて，半自動的な乱数生成アルゴリズムの開発に着手した．これはエンドツーエンドで半自動化された PRNGs の生成を目指し，学習型乱数生成手法（Learned Pseudo Random Number Generator:

LPRNG）を提案する．これまでにも，深層学習を用いた乱数生成手法の開発に関する研究は行われてきたが，統計的検定を十分に満たす手法の開発は未だに達成されていない [23–32]．LPRNG は，Wasserstein 距離 [33] を用いた敵対的生成ネットワーク GAN [34] を活用しており，MT 法を用いて得られる乱数を教師データとして使用する．このモデルを用いて，NIST 乱数検定を満たす実用的な乱数を生成することに成功した．このことから，一般的な利用に十分な品質を持つ初のデータ駆動型乱数生成手法と言える．

結論として，本研究で開発した乱数生成手法は，それぞれのシミュレーション手法において計算コストを削減し，高速化を実現するとともに，NIST 乱数検定を満たす品質を有することを示した．さらに，データ駆動型の乱数生成手法は，乱数生成手法開発の自動化に寄与し，将来のシミュレーション手法の発展に貢献する可能性を秘めている．

1.3 論文の構成とまとめ

本論文の構成について述べる．本論文は大きく 2 つのパートに分けられる．2-6 章では，2 つの分子シミュレーション手法 : DPD 法と MC 法に特化した高速な乱数生成手法を提案した．7-9 章では，深層学習モデルを用いた乱数生成手法を提案した．10 章では，2 つのパートの研究を総括した．2 章と 7 章では，それぞれの研究の背景と目的について詳細に説明する．本論文で取り扱う研究は次のようにまとめられる．

1. DPD 法特化型の高速な乱数生成手法
   [TEA1 法の開発] : 従来手法 TEA8 法の 8 倍高速な乱数生成手法 (4 章)
   [SHIFT 法の開発] : 従来手法 TEA8 法の 62 倍高速な乱数生成手法 (5 章)
2. MC 法特化型の高速な乱数生成手法
   [3XORs 法の開発] : MT 法の 5 倍高速な乱数生成手法 (6 章)

また上記 2 点の研究を発展させるため，7-9 章では，深層学習モデルを用いたデータ駆動型の乱数生成手法の提案を行った．

3. 敵対的生成ネットワークによる学習型乱数生成手法
   [Learned-PRNG の開発] : NIST 乱数検定を合格する学習型乱数生成手法 (9 章)

1.4 記号表

本論文では断りが無い限り，table. 1.1 中の記号は特定の物理量を意味するものとする．

Table 1.1: 記号表

記号	意味	単位
m	粒子の質量	g/mol
$\mathbf{r}$	座標 ($\mathbf{r} = (r_x, r_y, r_z)^T$)	Å
$\mathbf{p}$	運動量 ($\mathbf{p} = (p_x, p_y, p_z)^T$)	kJ/mol
$\mathbf{v}$	速度 ($p = mv$)	
σ	長さの Lennard-Jones パラメータ	Å
ϵ	エネルギーの Lennard-Jones パラメータ	kJ/mol
k_B	ボルツマン定数	kJ/(mol・K)
N	系の分子数	
K	運動エネルギー	kJ/mol
E	ポテンシャルエネルギー	kJ/mol
V	体積	$Å^3$
T	温度	K
β	逆温度 ($= 1/k_BT$)	$(\text{kJ/mol})^{-1}$
P	圧力	MPa

第 2 章

背景：分子シミュレーションにおける乱数生成について

2.1　散逸粒子動力学法のための乱数生成手法の開発

様々な手法が存在するシミュレーションの中でも，近年特に，計算機の発達を受けてますます利用価値が高まっている手法の一つが，分子シミュレーションである．実験では空間的，時間的に観察することのできない分子スケールの現象解明によって，新しいバイオマテリアルのデザインなどを始めとし，分子生物化学，生態材料科学などに役立てられている．スーパーコンピュータを用いての計算は，例えば，一般的に難しいとされてきた全原子シミュレーションでの，細胞小器官全体の再現や，100μs を超える長時間のタンパク質シミュレーションなどのように非常に大きいスケールでの計算を可能にしており，ますます多岐にわたる応用的な利用が議論されている [4,5]．分子シミュレーションの一つである散逸粒子動力学法 (Dissipative Particle Dynamics: DPD) は，全原子分子動力学法と違い，数個の粒子を 1 つの粗視化粒子としてみなし，少ない計算量で比較的大きな現象を取り扱う手法である [17–19]．一般的な粗視化分子動力学法では，粗視化に伴って粒子の自由度が減少し，分子内摩擦を扱えていないために，正確な動的性質を再現できないという問題が指摘されているが，DPD 法では，粒子間力を算出する際に，保存力項に加え，散逸力項とランダム力項を加えることで，粗視化に伴い再現できなくなる分子の摩擦と揺動の効果を再現し，比較的小さな計算コストで数百 nm，数十 μs ほどのスケールの動的性質の解析が可能となった．それにより，生体膜における両親媒性分子について，垂直方向の移動であるフリップフロップやそれらの自己組織化の観察, ポリマー [35–39] 生体膜 [40–43]，コロイド [44–47] ，そしてその他の複雑な物質 [48–51] に対してなど，長時間スケールのシミュレーションが必要なメソスケール系の動的性質を再現することが可能である．さらに DPD 法はその利用目的に合わせ，自由エネルギーの導入により一般的な状態方程式を与える，Many-body-DPD [52–54]，エネルギー保存を満たした，Energy-conserving-DPD [55, 56]，それらの利点をとりいれ，より一般化した，Smoothed-DPD [57, 58] など様々な改良が重ねられ，現在でも多くの研究者がより洗練された DPD 法の開発に取り組んでおり [59–62]，議論されている. [63]．さらに，並列計算で用いる DPD 法によるシミュレーションコードも複数開発されており，一般的に利用可能である. [64–66] しかし，スーパーコンピュータ等を用いて DPD 法による計算の大規模並列化を行う際，生じる問題がある. 分子動力学法など，力が保存力のみで構成される場合は，粒子間距離から力を計算す

ることができるため，効率的な並列計算が可能であったが，DPD 法のように，力が保存力のみならず，乱数を用いて計算されるランダム項を含む場合は，各計算ノードごとにそれぞれ異なった疑似乱数生成を行うため，各粒子ペアに共通したランダム力を再現することが難しく，高い効率の並列計算を実行することは困難である. [67–71] そこで，各計算ノードが共有する情報を，暗号学的ハッシュ関数で変換することにより，粒子ペアごとに共通した乱数を生成する方法が考案されている [20]．また，その際，使用されている暗号学的ハッシュ関数は TEA(Tiny Encryption Algorithm) [22] である．このアイデアに基づいて，その後より効率的な乱数生成法が開発されている [64, 72]．ここでは，計算速度と生成される乱数の質の調整が容易であることから，TEA 法の使用に焦点を当てる．ここで TEA 法は，疑似乱数生成器としての利用に限るため，計算コストと生成される乱数のランダム性（以降では乱数性質と呼ぶ）のバランスを考慮し，Zafar らの提案により，内部が軽量化されている [71]．しかし，この軽量化は DPD 法での利用を考慮して最適化されたものではないため，さらなる軽量化による計算の高速化が期待できると考えられる. そこで本研究ではまず，ハッシュ関数の見直しを行い，乱数の性質の評価と，DPD シミュレーションへの影響を観察する. 本実験では使用する乱数シードを適切に選択することで，TEA 法の計算量を削減できることを示す.

次の実験では，アルゴリズムを発展させて暗号化を用いない新しい乱数生成法 “SHIFT 法” を提案する．粒子座標を種数として用いた場合に，さらに高速な乱数生成が可能であることを実証する．粒子座標を種数として用いた場合の性質を調べ，望ましい乱数性質を持つことを初めて示した．また，粒子座標のランダムな性質を利用することで，従来の方法のように暗号化に大きな計算資源を割くことなく，高速に乱数を生成できることを示す．この新手法の開発にあたり，以下の手順で検証実験も行った： (i)MT 法による乱数を用いて実行した DPD シミュレーションの粒子座標の乱数性を，NIST 乱数検定を用いて各タイムステップで評価する； (iii)DPD 法では一般に一定の時間ステップ幅が用いられるが，より短い時間ステップ幅をテストし，新乱数生成法を用いた場合に，粒子座標から抽出された連続生成乱数間の相関にどのような影響を与えるかを調べた．SHIFT 法は，ランダムな特徴を抽出することにより，システムに既に存在する数値を効率的に利用する．この考え方は DPD シミュレーションだけに限定されるものではない．ペアポテンシャルに乱数性質の良い成分を含む分子モデルのシミュレーションであれば，特に並列計算環境でシミュレーションを行う場合に，本手法の採用が有効である．また，ブラウン力学，モンテカルロ法など，様々な数値シミュレーションに拡張することができる．さらに，SHIFT 法で生成される乱数の周期は常に十分であり，シミュレーションされるシステムのサイズに比例してスケールする．今回提案する乱数生成法は粒子座標を使用するため，乱数の周期は粒子座標と速度が同一の状態間の間隔として定義される．この周期は粒子数とともに増加するが，これはシステム全体の可能な状態の数が増加するためである．倍精度（64 ビット），3 次元，N 粒子を仮定し，粒子座標と粒子速度の両方を考慮すると，粒子系が取り得る状態の総数は $2^{64\times3\times N\times2}$ となる．系は確率的に遷移するので，各状態がサンプリングされる確率は等しくなる．したがって，粒子数 N が増加するにつれて，粒子座標と粒子速度が 2 回目に一致する確率は減少し，周期が長くなると推定できる．粒子座標をシード数として用いる場合，高度な暗号化アルゴリズムにおいても同様である．したがって，非常に大規模なシステムに対して高いスケーラビリティを持ち，将来的に計算機資源が増大しても本技術は有用であり続けると考えられる．MT 法や Xorshift 法のよう

ないくつかの効率的な PRNG が存在するが，SHIFT 法はより単純で，より少ない演算のみしか必要としない．従って SHIFT 法は，ある条件下では計算効率が良いだけでなく，実装が簡単で革新的であると言える．

開発した本手法は水分子のシミュレーション，また，適用例として，POPC 脂質 (1-Palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine) を用いた生体膜の系にも利用し，従来の手法との物性値の差を比較した. 計算コストの比較では，単純に乱数を生成する時間の比較と，DPD 法によって大規模な系を計算した場合に，力の計算にかかる計算コストを比較した. この様に，DPD 法の精度を落とさない程度まで軽量化を図ることで，広範なメソスケールの粗視化シミュレーションをスーパーコンピュータ等の超並列アーキテクチャでさらに高速に計算することが可能となり，DPD 法による生体分子やポリマーの理解と現象解明を促進すると考えられる.

2.2 モンテカルロ法のための乱数生成手法の開発

本研究では DPD 法の並列計算用に開発した本手法を発展させ，より一般的な分子シミュレーション手法である，モンテカルロ分子シミュレーション：MC 法に適用した．MC 法では，DPD 法と比べて一つのステップで生成するべき乱数の個数が多く，さらに乱数のシードとして利用できる素材が限られていたため，新たなアプローチを導入する必要があった．この課題に対処するため，本研究ではシミュレーション全体を通して再利用可能な乱数シードを新たに用意し，それらを xor 演算や shift 演算などの簡易的な bit 演算のみで，少数のシードから多くの個数の乱数を生成する方法を実現した．この手法は，効率的かつ高速な乱数生成を可能にし，MC 法の精度維持したまま全体の計算コストを削減できることを示した．この新たな乱数生成手法の適用は，MC 法，DPD 法よりも一般的な手法に対する応用の拡大を意味し，数値計算技術の多くの分野に適用できる可能性を示唆する．

MC 法は，MD 法や他の分子シミュレーション手法と同様，物質の分子レベルでの挙動を理解するために広く用いられている手法である．例えば，メタンと水の界面特性の研究において，勾配理論とモンテカルロ分子シミュレーションを同時に適用することで，界面挙動のより詳細な記述が可能になっている [10]．また，単層カーボンナノチューブ内の N2，CH4，CO，CO2 ガスの吸着に関する研究では，実験的手法とモンテカルロ分子シミュレーションを組み合わせることで，吸着過程の微細なメカニズムを明らかにしている [11]．閉じ込め系における水の吸着過程も同様に研究されている [73]．さらに，K-モンモリロナイトと Na-モンモリロナイトの水和に関するモンテカルロ分子シミュレーションは，特定の温度と圧力条件下での粘土鉱物の水和特性を明らかにしている [12]．最後に，モンテカルロ分子シミュレーションプログラムの並列実装に関する研究もなされており，計算効率の向上と大規模シミュレーションの実現を目指している [13].

これらの先行研究は，モンテカルロ分子シミュレーションの多様な応用とその科学的な影響を示しており，本研究の基盤となる重要な文脈を提供している．

特に MC 法では，MD 法による実施が困難なグランドカノニカルアンサンブルを用いた物質の微視的な挙動の解析に多用されている．その主な理由は，モンテカルロ法が確率的な手法を用いることで，動的な発展を追跡する MD 法よりも粒子の生成と消去を取り扱いやすいからである．結果として相転移点を含む複雑な系の解析をすることができる．MC 法のための乱数生成手法を

開発する上で，“SHIFT 法” と同様にシミュレーションの計算過程で二次的に生成される乱雑性を乱数として用いる．MC 法では，DPD 法と比べて一つのステップで生成するべき乱数の個数が多く，乱数のシードとして利用できる乱雑性のある bit 列の個数が限られている．この課題に対処するため，シミュレーション全体を通して再利用可能な乱数シードを新たに用意し，それらを xor 演算や shift 演算などの簡易的な bit 演算のみで，少数のシードから多くの個数の乱数を生成する方法を示す．この MC 法のための乱数生成手法 “3XORs” により，効率的かつ高速な乱数生成を可能にし，MC 法の精度を維持したまま全体の計算コストを削減できることを示す．

MC 法のための，この新たな乱数生成手法の適用は，DPD 法よりも一般的な手法に対する応用の拡大を意味し，数値計算技術の多くの分野に適用できる可能性を示唆するものである．

2.3　分子シミュレーションのための新規乱数生成手法に関する研究目的

本研究では，現代の計算機科学において，密接な関係がある乱数生成手法に着目する．各計算機シミュレーション手法に特化したアルゴリズムを開発し，従来よりも高速なシミュレーションを達成する．そして，深層学習モデルを用いた乱数生成手法の開発方法を示し，新しい乱数生成のあり方とその応用方法を提案する．本研究では具体的に以下の項目を達成する．

1. 並列計算時に用いられる散逸粒子動力学法の暗号学的ハッシュ関数 TEA の内部を軽量化することによる疑似乱数の性質の変化を調べ，水分子系の DPD シミュレーションに与える影響を解析する．
2. TEA の乱数生成法をもとに，分子シミュレーションでの利用により適した新規乱数生成手法を設計する．
3. 新手法を用いて生成した乱数を NIST 検定を用いて検証する．また，乱数を生成する上で，より厳しい条件の場合に対しても同様に NIST 検定をおこない，乱数を評価する．
4. 散逸粒子動力学法のための新規乱数生成手法の計算コストの変化を調べる．
5. より一般的な分子シミュレーションである，モンテカルロ分子シミュレーションに適用した乱数生成手法を開発する．
6. 本手法を用いた場合のモンテカルロ分子シミュレーションの精度を比較する．
7. モンテカルロ法のための新規乱数生成手法の計算コストの変化を調べる．

第 3 章

分子シミュレーションと乱数生成法に関する理論

原子及び分子のような粒子間の相互作用を元に物理現象や物性を解析するシミュレーション手法は分子シミュレーションと呼ばれ，代表的なものに分子動力学法があげられる．この手法により多くの動力学的性質を得ることができる．しかし一方で，全ての原子を考慮し計算する，全原子の計算は高い計算コストのため，数 nm，数百 ns 等の限られたスケールでの解析が一般である．そこで，全原子の分子動力学法では扱うことが難しいスケールでの計算を可能にするために，様々な計算手法が考案されてきた．数百 nm，数 ns 等の大きいスケールの現象を取り扱うため，複数の分子をまとめて一つの粒子として相互作用を計算する粗視化手法，コロイド系などの，運動が異なる時間スケールのもの同士が混在する系の解析を行うブラウン動力学法，さらにその手法を改善し，運動量保存を満たした系の再現ができる散逸粒子動力学法など，様々な手法がこれまで開発されてきた．この章では散逸粒子動力学法を利用するに至る歴史的背景，またそれぞれの手法のメカニズムを解説する．

3.1　分子動力学法

分子動力学法は全原子に対してニュートンの運動方程式を逐次的に解くことで，各計算 step ごとの分子の軌跡を追跡し，物理現象を再現する分子シミュレーションである．具体的な方法として，粒子 i について初期状態の初期座標 $\boldsymbol{r}_i(0)$, 初期速度 $\boldsymbol{v}_i(0)$ を決定する．次にある時刻 t での粒子座標 $\boldsymbol{r}_i(t)$ を用いて，粒子間相互作用の力 $\boldsymbol{F}_i(t)$ の計算を行う．計算した力を元に数値積分法を用い，Δt 秒後の粒子の位置 $\boldsymbol{r}_i(t+\Delta t)$ と速度 $\boldsymbol{v}_i(t+\Delta t)$ の計算を行う．ここで得られた粒子座標から同様に次 step の力 $\boldsymbol{F}_i(t+\Delta t)$ を求める．このように相互作用計算と粒子の座標，速度の更新を繰り返すことで，粒子 1 つの運動を追跡することができる．この計算サイクルを全ての粒子に適用することで，物理的な現象を再現することができる．このシミュレーションの流れを Fig. 3.1 に示す．

上述した方法で，水素，炭素，酸素など，系に存在する全ての原子を考慮し，それらの間に働く相互作用を全て計算する全原子モデルの計算は，扱う系を再現する上で正確なものである．一方，計算コストが大きく，数百万から数千万原子を扱うような巨大な分子集団を扱う場合，現実的ではない計算コストを必要とする．コロイド流体，生体膜，ポリマーなど，巨大な分子集団の巨視的

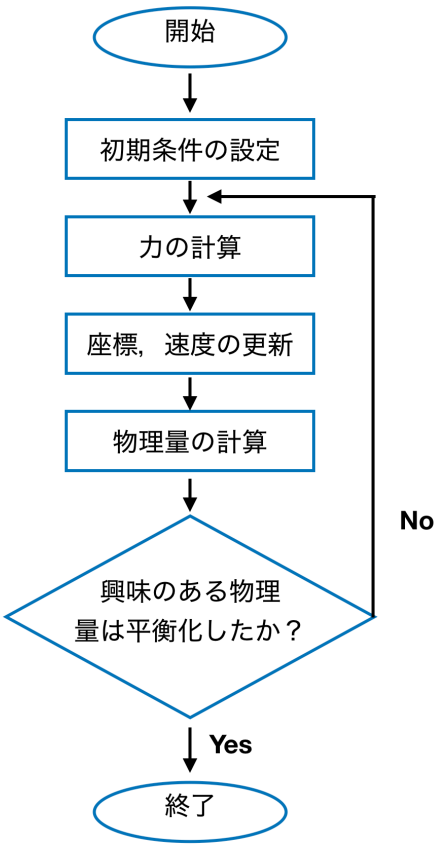


Figure 3.1: 分子動力学法の流れ

な運動を追跡するのに必要な時間スケールはマイクロ秒オーダであり，極めて大きい．水素原子を含むために 1 ～ 2 fs の時間刻みで運動方程式を解く分子動力学法では，これらのような巨大系の長時間の動力学を解析することは現代の計算技術をもってしても，極めて困難である [16]．ここで，例えば，ベシクル変形などの巨大な分子集団の巨視的な運動を再現する際，各分子間結合の分子内伸縮運動，角運動などは全体の挙動に与える影響が小さく，無視しても構わない場合が多い．言い換えると，巨視的な運動を追跡する研究目的からすると，無駄な計算をしている場合が多いことがわかる．そこで，$-CH_3$，$-CH_2-$ や H_2O などの複数の粒子を 1 つの相互作用点として，さらには，複数のメチレン基や複数の水分子を 1 つの相互作用点だけで取り扱う近似を導入する．このような近似を United atom モデルと呼ぶ．この相互作用点の間には，全原子のモデルの場合と同様に距離の関数として相互作用を仮定する．このような粗視化されたモデルを用いることで，取り扱うべき相互作用点が大幅に減る．例えば，H_2O の 4 粒子を 1 つの粒子として扱うと，もとの場合と比べて，相互作用点は 1/12 となる．また，$-(CH_2)_4-$ を 1 つの相互作用点として扱うと，その質量は水素原子の 56 倍であり，Δt は質量の平方根に比例してとって良いと考えれば，結果的に 7.5 倍の時間 step 幅のものを用いることができる．したがって計算時間は粒子数 N に対して $N\log N$ に比例し，時間 step 幅に Δt に反比例するので，計算負荷は約 1/100 に軽減されることがわかる．このことは 100 倍スケールの解析が可能になることを示している．ここで扱う粗視化モデルにおける分子間相互作用のパラメータは経験的に得られたポテンシャルであり，全原子分子動力学や，第一原理計算などを用いたより詳細な計算に基づいて経験的に構築するか，密度などの特定の実験値を再現できるように試行錯誤して決めていく必要がある．ここ

で注意するべきは，通常の粗視化分子動力学シミュレーションにおいて，粗視化に伴い，本来運動方程式中にブラウン運動の原因として現れるべき揺らぎと散逸の影響が考慮されていないことである．これによって粗視化分子動力学の系では，それに対応する全原子分子動力学による系に比べて，早い動的性質が観測されうる．したがって，粗視化分子動力学を用いて，拡散係数，粘性係数等の動的性質を詳しく議論することは困難である．このことから，粗視化手法を用いながら，動的性質を議論するためには，この揺らぎと散逸の影響を考慮しなけれえばならないことがわかる．これらの影響を考慮した分子シミュレーションの手法の一つとしてブラウン動力学があげられる．複数の粒子をひとまとめに見なした粗視化粒子に対し，粒子間相互作用を表す保存力以外に，不規則な運動を表現するランダム力と速度に比例する散逸力が作用している．この手法では，粒子の周りの分子を一様な溶媒として扱い，そこから受ける熱の揺らぎをランダム力として表現する．こうすることで，揺らぎ，散逸の影響を加味した分子シミュレーションを行うことができる．ただし，この手法では各粒子ごとに独立して作用するランダム力と散逸力のために，運動量保存の法則が成り立たないという問題が生じる．そこで，新たに加味されたランダム力と散逸力を各粒子ペアごとに対応した力として作用させることで，質量だけでなく，運動量をも保存しながら，メソスケールなスケールで動的性質を解析する手法が開発された．それは散逸粒子動力学法と呼ばれる．

3.2 散逸粒子動力学法

散逸粒子動力学 (DPD) 法は Hoogerbrugge and Koelman [74] によって考案された手法であり，その後 1995 年に Espanol and Warren [75] は揺動散逸定理を元にし，系がカノニカルアンサンブルを満たすように，ランダム力と散逸力を関係づけた．また，1997 年に Groot and Warren [76] は DPD 法に Flory-Huggins の理論を導入し，分子種ごとに相互作用を区別することを容易にした．

3.2.1 アルゴリズム

DPD 法は先述したように，複数の分子の集合を 1 つの粒子とみなして計算を行う．その粗視化粒子の挙動は次の運動方程式

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i \tag{3.1}$$

$$m_i \frac{d\boldsymbol{v}_i}{dt} = \boldsymbol{f}_i \tag{3.2}$$

に従う．$m_i, \boldsymbol{r}_i, \boldsymbol{v}_i, \boldsymbol{f}_i$ はそれぞれ粒子 i の質量，位置，速度，作用する力を示す．一般的に全ての粒子の質量は等しいとされ，$m_i = m \equiv 1$ と定められる．ここで $\boldsymbol{f}_i$ は次に示す 3 つの 2 体間の力の総和として

$$m\frac{d\boldsymbol{v}_i}{dt} = \boldsymbol{f}_i = \sum_{j\neq i}(\boldsymbol{F}_{ij}^{\mathrm{C}} + \boldsymbol{F}_{ij}^{\mathrm{D}} + \boldsymbol{F}_{ij}^{\mathrm{R}}) \tag{3.3}$$

と定められる．$\boldsymbol{F}_{ij}^{\mathrm{C}}, \boldsymbol{F}_{ij}^{\mathrm{D}}, \boldsymbol{F}_{ij}^{\mathrm{R}}$ はそれぞれ粒子 j から i に作用する保存力，散逸力，ランダム力を表す．保存力は，粗視化粒子の相互作用の力で用いられているものと同様，ソフトポテンシャル

に起因する斥力として，

$$\boldsymbol{F}_{ij}^{\mathrm{C}} = \begin{cases} -a_{ij}\left(1-\frac{r_{ij}}{r_{\mathrm{c}}}\right)\boldsymbol{e}_{ij} & (r_{ij} \le r_{\mathrm{c}}) \\ 0 & (r_{ij} > r_{\mathrm{c}}) \end{cases} \tag{3.4}$$

のように提案されている．r_{c} はカットオフ半径を表しており，DPD 法では長さの単位とし，$r_{\mathrm{c}} \equiv 1$ とする．また，a_{ij} は粒子 i，j 間の最大斥力，そして $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$，$r_{ij} = |\boldsymbol{r}_{ij}|$，$\boldsymbol{e}_{ij} = \boldsymbol{r}_{ij}/r_{ij}$ とする．散逸力と，ランダム力は，γ を摩擦係数，σ を揺らぎの大きさ，$\omega^{\mathrm{D}}(r_{ij})$，$\omega^{\mathrm{R}}(r_{ij})$ を重み関数，また，$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$ として，保存力と同様 $r_{ij} \ge r_{\mathrm{c}} = 1$ で 0 になるという条件のもと，

$$\boldsymbol{F}_{ij}^{\mathrm{D}} = -\gamma\omega^{\mathrm{D}}(r_{ij})(\boldsymbol{e}_{ij}\cdot\boldsymbol{v}_{ij})\boldsymbol{e}_{ij} \tag{3.5}$$

$$\boldsymbol{F}_{ij}^{\mathrm{R}} = \sigma\omega^{\mathrm{R}}(r_{ij})\theta_{ij}\boldsymbol{e}_{ij} \tag{3.6}$$

と表される．ここで θ_{ij} は以下の特性を有するガウス性白色雑音である．

$$\begin{aligned} \theta_{ij}(t) &= \theta_{ji}(t) \\ \langle\theta_{ij}(t)\rangle &= 0 \\ \langle\theta_{ij}(t)\theta_{kl}(t')\rangle &= (\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk})\delta(t-t') \end{aligned} \tag{3.7}$$

ここで δ_{ij} は Kronecker のデルタであり，$i = j$ のとき $\delta_{ij} = 1$ となり，それ以外の場合は $\delta = 0$ となる．これは各粒子ペアごとに $\theta_{ij} = \theta_{ji}$ を満たすので，運動量保存が成立する．さらに，Eq. (3.5)，(3.6) 中の $\omega_{\mathrm{D}}(r_{ij}), \omega_{\mathrm{R}}(r_{ij})$ は

$$\omega^{\mathrm{D}}(r_{ij}) = [\omega^{\mathrm{R}}(r_{ij})]^2 = \begin{cases} \left(1-\frac{r_{ij}}{r_{\mathrm{c}}}\right)^2 & (r_{ij} \le r_{\mathrm{c}}) \\ 0 & (r_{ij} > r_{\mathrm{c}}) \end{cases} \tag{3.8}$$

の関係式を満たす．Eqs. (3.5)，(3.6) を (3.3) に代入して整理すると，粒子の運動方程式は

$$m\frac{d\boldsymbol{v}_i}{dt} = \sum_{j\neq i}\boldsymbol{F}_{ij}^{\mathrm{C}} - \sum_{j\neq i}\gamma\omega^{\mathrm{D}}(r_{ij})(\boldsymbol{e}_{ij}\cdot\boldsymbol{v}_{ij})\boldsymbol{e}_{ij} + \sum_{j\neq i}\sigma\omega^{\mathrm{R}}(r_{ij})\theta_{ij}\boldsymbol{e}_{ij} \tag{3.9}$$

となる．時間 t から時間 $(t+\Delta t)$ までの微小区間で積分し，差分方程式を

$$\Delta\boldsymbol{r}_i = \boldsymbol{v}_i\Delta t \tag{3.10}$$

$$\Delta\boldsymbol{v}_i = \frac{1}{m}\left(\sum_{j\neq i}\boldsymbol{F}_{ij}^{\mathrm{C}} - \sum_{j\neq i}\gamma\omega^{\mathrm{D}}(r_{ij})(\boldsymbol{e}_{ij}\cdot\boldsymbol{v}_{ij})\boldsymbol{e}_{ij}\right)\Delta t + \frac{1}{m}\sum_{j\neq i}\sigma\omega^{\mathrm{R}}(r_{ij})\Delta W_{ij}\boldsymbol{e}_{ij} \tag{3.11}$$

のように得る．ここで ΔW_{ij} は Eq. (3.7) より

$$\langle\Delta W_{ij}\rangle = 0, \langle\Delta W_{ij}\Delta W_{i'j'}\rangle = (\delta_{ii'}\delta_{jj'}+\delta_{ij'}\delta_{ji'})\Delta t \tag{3.12}$$

の確率特性を有する．さらにここで $\Delta W_{ij} = \zeta_{ij}(\Delta t)^{(1/2)}$ を満足する ζ_{ij} を用いて Eq. (3.11) を書き直すとその式の第三項は

$$\frac{1}{m}\sum_{j\neq i}\sigma\omega^{\mathrm{R}}(r_{ij})\boldsymbol{e}_{ij}\zeta_{ij}\sqrt{\Delta t} \tag{3.13}$$

のように表される．ただし，先述した θ_{ij}，ΔW_{ij} と同様に，ζ_{ij} は

$$\langle\Delta\zeta_{ij}\rangle = 0, \langle\Delta\zeta_{ij}\Delta\zeta_{i'j'}\rangle = (\delta_{ii'}\delta_{jj'}+\delta_{ij'}\delta_{ji'}) \tag{3.14}$$

の確率特性を有する．また k_{B} は Boltzmann 定数，T を温度として，散逸揺動定理により

$$\sigma^2 = 2\gamma k_{\mathrm{B}} T \tag{3.15}$$

を満足する．この関係式により系が熱力学的平衡状態の時，正しい平衡分布を与えることが保証される．以上の結果から Eq. (3.11) を書き直すと，DPD 法による運動方程式は

$$\Delta \boldsymbol{r}_i = \boldsymbol{v}_i \Delta t$$

$$\Delta \boldsymbol{v}_i = \frac{1}{m}\left(\sum_{j \neq i} \boldsymbol{a}_{ij} \omega^{\mathrm{R}}(r_{ij}) \boldsymbol{e}_{ij} \Delta t - \frac{\gamma}{m} \sum_{j \neq i} \omega^{\mathrm{D}}(r_{ij})(\boldsymbol{e}_{ij} \cdot \boldsymbol{v}_{ij}) \boldsymbol{e}_{ij} \right) \Delta t + \frac{\sqrt{2\gamma k_{\mathrm{B}} T}}{m} \sum_{j \neq i} \omega^{\mathrm{R}}(r_{ij}) \boldsymbol{e}_{ij} \zeta_{ij} \sqrt{\Delta t} \tag{3.16}$$

のように表される．ここで F_{ij}^{C} には Eq. (3.8) の ω^{R} を用いた．

3.2.2 無次元化

一般的に分子シミュレーションにおいては，諸量を無次元化して計算が行われる．無次元化を行うにあたって，次のような代表値が必要になる．ここでは，速度，距離，時間，数密度の代表値としてそれぞれ $(k_{\mathrm{B}}T/m)^{(1/2)}$，$r_{\mathrm{c}}$，$r_{\mathrm{c}}(m/k_{\mathrm{B}}T)^{(1/2)}$，$(1/r_{\mathrm{c}}^3)$ を用いる．これらを用いると，先述した Eqs. (3.10)，(3.16) は

$$\Delta \boldsymbol{r}_i^* = \boldsymbol{v}_i^* \Delta t^* \tag{3.17}$$

$$\Delta \boldsymbol{v}_i^* = \sum_{j \neq i} \boldsymbol{a}_{ij}^* \omega^{\mathrm{R}}(r_{ij}^*) \boldsymbol{e}_{ij} \Delta t^* - \gamma^* \sum_{j \neq i} \omega^{\mathrm{D}}(r_{ij}^*)(\boldsymbol{e}_{ij} \cdot \boldsymbol{v}_{ij}^*) \boldsymbol{e}_{ij}) \Delta t^* + \sqrt{2\gamma^*} \sum_{j \neq i} \omega^{\mathrm{R}}(r_{ij}^*) \boldsymbol{e}_{ij} \zeta_{ij} \sqrt{\Delta t^*} \tag{3.18}$$

と表される．ただし $\omega^{\mathrm{R}}(r_{ij}^*)$，$\omega^{\mathrm{D}}(r_{ij}^*)$，$a^*$，$\gamma^*$ は

$$\omega^{\mathrm{D}}(r_{ij}^*) = [\omega^{\mathrm{R}}(r_{ij}^*)]^2 = \begin{cases} (1 - r_{ij}^*)^2 & (r_{ij} \leq 1) \\ 0 & (r_{ij}^* > 1) \end{cases} \tag{3.19}$$

$$a^* = a \frac{r_{\mathrm{c}}}{k_{\mathrm{B}} T}, \tag{3.20}$$

$$\gamma^* = \gamma \frac{r_{\mathrm{c}}}{\sqrt{m k_{\mathrm{B}} T}} \tag{3.21}$$

である．なお，無次元化された量は「*」を付して表した．

3.2.3 斥力パラメータの導出

ここでは，DPD 法で用いられる保存力項の斥力パラメータ a_{ij} の導出方法について述べる．任意の液体における熱力学的状態が soft sphere モデルで再現できるならば，液体中の揺動が

正確に再現されている必要がある．これらは対象とする系の圧縮率によって決定されるので，Weeks-Chandler-Anderson の摂動定理により，系の圧縮率 κ は，n を数密度，p を圧力，κ_T を等温圧縮率として

$$\kappa^{-1} = \frac{1}{nk_{\mathrm{B}}T\kappa_T} \tag{3.22}$$

で表される．κ_T は 300 K にいて 15.9835 である．これに対応する状態方程式を導出するために，Virial 定理を用いて圧力を密度の関数として書き出すと，$g(r)$ を動径分布関数，V を体積として，

$$\begin{aligned} p &= \rho k_{\mathrm{B}}T + \frac{1}{3V}\left\langle \sum_{j>i} (\boldsymbol{r}_i - \boldsymbol{r}_j)\cdot \boldsymbol{f}_i \right\rangle \\ &= \rho k_{\mathrm{B}}T + \frac{1}{3V}\left\langle \sum_{j>i} (\boldsymbol{r}_i - \boldsymbol{r}_j)\cdot \boldsymbol{F}_i^{\mathrm{C}} \right\rangle \\ &= \rho k_{\mathrm{B}}T + \frac{2\pi}{3}\rho^2 \int_1^0 r f(r) g(r) r^2 dr \end{aligned} \tag{3.23}$$

として得られる．この式により，任意の a に対して，密度の関数としてシミュレーションから圧力を得る．ここで a を適当に決定し，横軸に密度，縦軸に圧力を ρ^2 で除した図を描くと，密度が十分大きい場合において，a に関わらず，全ての曲線が重なる．さらに密度がある大きさに達すると，ほぼ一定値となり，過剰圧力は ρ^2 に比例していることがわかる．ここから，高密度領域においては近似的に次の状態方程式

$$p = \rho k_{\mathrm{B}}T + \alpha a \rho^2 \qquad (\alpha = 0.101 \pm 0.001) \tag{3.24}$$

が利用できる．この式と Eq. (3.22) から，

$$\kappa^{-1} = 1 + \frac{2\alpha a\rho}{k_{\mathrm{B}}T} \approx 1 + \frac{0.2a\rho}{k_{\mathrm{B}}T} \tag{3.25}$$

が得られる．ここで，水の等温圧縮率は $\kappa_{\mathrm{water}}^{-1} \approx 16$ で与えられるので，

$$\frac{a\rho}{k_{\mathrm{B}}T} \approx 75 \tag{3.26}$$

が得られる．しかし，ρ の増加とともに，各粒子に対する相互作用の数が線形に増加し，計算時間の増大を招くことが知られている．そのため多くの研究では，$\rho = 3$ が用いられている．この場合，DPD 法における水の斥力パラメータは

$$a = 25k_{\mathrm{B}}T \tag{3.27}$$

として得られる．

3.2.4　計算スケール

DPD 法は上述したように複数の粒子をたった一つの作用点として見なし，計算を実行するため，計算負荷が大幅に削減される．また，全原子の分子動力学法による計算と比較して，計算

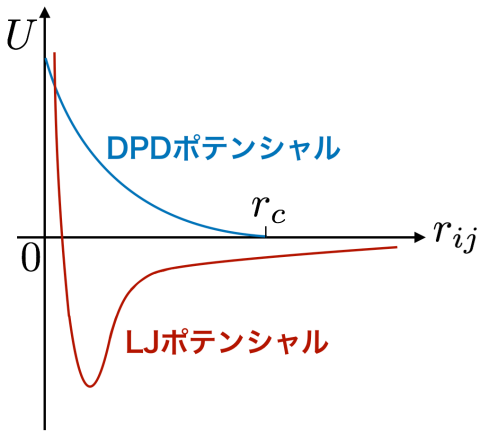


Figure 3.2: LJ ポテンシャル，DPD ポテンシャル模式図

上，利点となる点がもう一つあげられる．一般的に全原子の分子動力学で用いられる保存力は Lennard-Jones (LJ) ポテンシャルを用いて

$$U_{\mathrm{LJ}}(r_{ij}) = 4\epsilon \left[(\frac{\sigma_0}{r_{ij}})^{12} - (\frac{\sigma_0}{r_{ij}})^{6} \right] \tag{3.28}$$

のように定義される．ϵ と σ_0 はそれぞれエネルギー及び長さの次元を持ち，物質により異なる定数である．ここで，LJ ポテンシャルと DPD 法によるポテンシャルを Fig. 3.2 に示す．Fig. 3.2 からわかるように，DPD 法では LJ ポテンシャルに比べ，短いカットオフ半径を用いることができる．さらに，LJ ポテンシャルでは粒子間距離が 0 付近において，発散しているのにたいして，DPD 法によるポテンシャルは発散しないことがわかる．これは，シミュレーションを実行する上で，粒子間の重なりを許すことを意味している．そのため時間 step 幅 Δt を大きくとっても，安定した計算を実行することができる．

DPD 法では上述したような利点のため全原子分子動力学法と比較して，Fig. 3.3 に示すようなスケールのシミュレーションが一般的である．

3.3　モンテカルロ法

モンテカルロ法（Monte Carlo, MC 法）は，乱数を用いて数値計算を行う手法の一つであり，物理，数学，工学，金融など多岐にわたる分野で応用されている．例えば，物理学では量子力学や統計力学の計算に使用され，統計学ではベイジアン統計学における事後分布のサンプリングや，汎用的な数値積分に使用される．また，金融学ではオプションの価格設定やリスク管理のためのシミュレーションに活用され，コンピューターサイエンスでは AI や機械学習の分野で，不確実性のある問題や最適化問題の解決に利用される．分子シミュレーション分野では，乱数を利用して対象系の熱平衡状態を模擬することが可能である．ただし，ランダムサンプリングによる計算効率の低下を避けるため，エネルギー的に安定な状態へのサンプリングを重点的に行う必要がある．この目的で，Metropolis 法などの重点サンプリング技法が一般的に用いられる．

熱平衡状態における詳細釣り合い条件は，状態 x から x' への遷移確率 $P(x'|x)$ とその逆遷移の確率 $P(x|x')$ が

$$P(x'|x)P(x) = P(x|x')P(x') \tag{3.29}$$

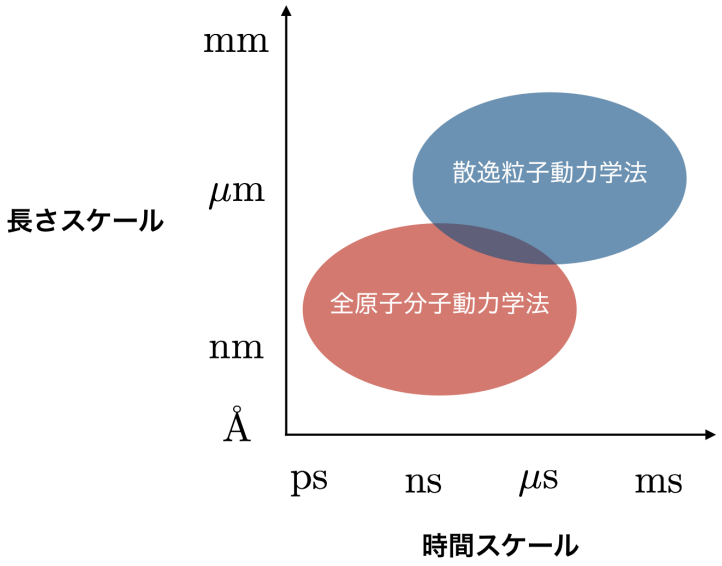


Figure 3.3: DPD 法の計算スケール

の関係を満たすことを意味する．カノニカル分布を仮定すると，確率分布 $P(x)$ は分配関数 $Z(\beta)$ と Boltzmann 因子を用いて，

$$P(x) = \frac{1}{Z(\beta)} e^{-\beta E(x)} \tag{3.30}$$

と表される．ここで，β は逆温度，$E(x)$ は状態 x のエネルギーである．Metropolis 法においては，状態 x から x' への遷移確率は，エネルギー差 $\Delta E = E(x') - E(x)$ に基づき，

$$P(x'|x) = \min\left(1, e^{-\beta \Delta E}\right) \tag{3.31}$$

と定義される．分子シミュレーションにおいて，系がカノニカル分布に従う熱平衡状態を効率的にサンプリングするために，以下の手順が踏まれる．

1. 初期状態のポテンシャルエネルギー E_1 を計算する．
2. 乱数を用いて分子の位置または配向をランダムに変更する．
3. 新しい状態のポテンシャルエネルギー E_2 を計算する．
4. $\Delta E = E_2 - E_1$ に基づき，遷移の受理または拒否を決定する．
5. 上記の手順を繰り返し，系が熱平衡状態に収束するまで実行する．

モンテカルロ法によるサンプリングは，分子の動的挙動を直接シミュレートするものではないが，熱平衡状態における系の統計的性質を評価する上で極めて有効である．低温での秩序構造など，特定の物理現象のシミュレーションにおいても広く用いられている．一方で，モンテカルロ法には計算上の問題が存在する．

問題の次元が高くなると，必要なサンプル数が指数的に増えるため，計算時間が大幅に増加する．これを次元の呪いと呼ぶ．これらの限界を克服するために，モンテカルロ法の改良版や変種が提案されている．例えば，重点サンプリングやマルコフ連鎖モンテカルロ法（MCMC）など，サンプリングの効率を上げる手法を詳細に解説している [9]．これらの手法は，より重要な領域からサンプルを抽出するという戦略を取り入れている．

さらに，モンテカルロ法の改良版として準モンテカルロ法がある．この方法は，より均等にサンプルを分布させるために，一様乱数ではなく超一様分布列 (Low-Descrepancy Sequence) を使用する．こうすることで，高次元の問題でもより効率的な近似が可能となる [77].

3.4 数値積分法

一般的に，考えている全ての粒子に対するそれぞれの運動方程式を解析的に解くことはできない．そのため分子シミュレーションにおいては差分近似法を用いて，時間に沿った数値計算を逐次的に行うことで系の時間発展を再現している．並進運動に関して，ニュートンの第二法則を解くだけの場合は Verlet 法が最もよく用いられる．現在の時刻を t とし，そこから時間刻み Δt 後と前の粒子 i の位置をそれぞれ $\boldsymbol{r}_i(t+\Delta t)$, $\boldsymbol{r}_i(t-\Delta t)$ とし，それらを Taylor 展開すると，

$$\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + \Delta t\dot{\boldsymbol{r}}_i(t) + \frac{\Delta t^2}{2!}\ddot{\boldsymbol{r}}_i(t) + \frac{\Delta t^3}{3!}\dddot{\boldsymbol{r}}_i(t) + O(\Delta t^4) \tag{3.32}$$

$$\boldsymbol{r}_i(t-\Delta t) = \boldsymbol{r}_i(t) - \Delta t\dot{\boldsymbol{r}}_i(t) + \frac{\Delta t^2}{2!}\ddot{\boldsymbol{r}}_i(t) - \frac{\Delta t^3}{3!}\dddot{\boldsymbol{r}}_i(t) + O(\Delta t^4) \tag{3.33}$$

$$\tag{3.34}$$

これらの Eqs. (3.32), (3.33) の両辺を足し合わせると，

$$\boldsymbol{r}_i(t+\Delta t) + \boldsymbol{r}_i(t-\Delta t) = 2\boldsymbol{r}_i(t) + \Delta t^2\ddot{\boldsymbol{r}}_i(t) + O(\Delta t^4) \tag{3.35}$$

となり，Eq. (3.2) より

$$\boldsymbol{r}_i(t+\Delta t) = 2\boldsymbol{r}_i(t) - \boldsymbol{r}_i(t-\Delta t) + \frac{\Delta t^2}{m_i}\ddot{r}_i(t) + O(\Delta t^4) \tag{3.36}$$

となる．これは，現在時刻 t において粒子 i が受ける力と位置，また Δt 前の位置から，Δt 後の位置が導出できることを示す．また，Eq. (3.32) から Eq. (3.33) の両辺を引くと，

$$\boldsymbol{v}_i(t) = \frac{\boldsymbol{r}_i(t+\Delta t) - \boldsymbol{r}_i(t-\Delta t)}{2\Delta t} + O(\Delta t^2) \tag{3.37}$$

が得られ，時刻 t における速度が求まる．以上が Verlet 法と呼ばれる手法であるが，実際のプログラミングでこれらの式を直接用いると数値計算上の桁落ちが生じるという問題がある．そこで以下に示す，Velocity-Verlet 法と呼ばれる Verlet 法を変形した手法がよく用いられる．Eq. (3.36) は

$$\begin{aligned}
\boldsymbol{r}_i(t+\Delta t) &= \boldsymbol{r}_i(t)\frac{1}{2} + \boldsymbol{r}_i(t)\frac{1}{2} + \boldsymbol{r}_i(t) - \boldsymbol{r}_i(t-\Delta t) \\
&\quad + \frac{1}{2}\boldsymbol{r}_i(t-2\Delta t) - \frac{1}{2}\boldsymbol{r}_i(t-2\Delta t) + \frac{\Delta t^2}{m_i}\ddot{r}_i(t) \\
&= \boldsymbol{r}_i(t) + \Delta t\frac{\boldsymbol{r}_i(t) - \boldsymbol{r}_i(t-2\Delta t)}{2\Delta t} \\
&\quad + \frac{1}{2}\left\{2\boldsymbol{r}_i(t-\Delta t) - \boldsymbol{r}_i(t-2\Delta t) + \frac{\Delta t^2}{m_i}\ddot{r}_i(t-\Delta t)\right\} \\
&\quad - \boldsymbol{r}_i(t-\Delta t) + \frac{1}{2}\boldsymbol{r}_i(t-2\Delta t) + \frac{\Delta t^2}{m_i}\ddot{r}_i(t) + \frac{\Delta t^2}{m_i}\ddot{r}_i(t) \\
&= \boldsymbol{r}_i(t) + \Delta t\left\{\dot{r}_i(t-\Delta t) + \frac{\Delta t}{m_i}\frac{\ddot{r}_i(t) + \ddot{r}_i(t-\Delta t)}{2}\right\} + \frac{\Delta t^2}{m_i}\ddot{r}_i(t)
\end{aligned}$$

のように式変形できる．ここで，時刻 t における速度を

$$\boldsymbol{v}_i(t) = \dot{r}_i(t - \Delta t) + \frac{\Delta t}{m_i}\frac{\ddot{r}_i(t) + \ddot{r}_i(t - \Delta t)}{2} \tag{3.38}$$

により導出するものとすれば，時刻 $t + \Delta t$ における位置は，

$$\boldsymbol{r}_i(t + \Delta t) = \boldsymbol{r}_i(t) + \Delta t \boldsymbol{v}_i + \frac{\Delta t^2}{m_i}\ddot{r}_i(t) + O(\Delta t^4) \tag{3.39}$$

から計算できる．Velocity-Verlet 法は Verlet 法と数学的に同等の式であるが，大きな数と小さな数を分離して，先に小さな数で和を取った後に大きな数に加えることができるので，計算上の桁落ちを防ぐことができる．

DPD 法では，力の成分の内，散逸項において速度に依存するため，上記の Velocity-Verlet 法を以下のように修正したものを利用する．

$$\boldsymbol{r}_i(t + \Delta t) = \boldsymbol{r}_i(t) + \Delta t \boldsymbol{v}_i(t) + \frac{1}{2}(\Delta t)^2 \boldsymbol{f}_i(t) \tag{3.40}$$

$$\tilde{\boldsymbol{v}}_i(t + \Delta t) = \boldsymbol{v}_i + \lambda \Delta t \boldsymbol{f}_i(t) \tag{3.41}$$

$$\tilde{\boldsymbol{f}}_i(t + \Delta t) = \boldsymbol{f}_i(\boldsymbol{r}_i(t + \Delta t), \tilde{\boldsymbol{v}}_i(t + \Delta t)) \tag{3.42}$$

$$\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i + \frac{\Delta t}{2}(\tilde{\boldsymbol{f}}_i(t), \tilde{\boldsymbol{f}}_i(t + \Delta t)) \tag{3.43}$$

3.5　周期境界条件

現実の物質系はアボガドロのオーダの数の分子で構成されている．また，我々が通常，研究の対象としている系もこのような巨大な数で構成された分子集団である．ここで，分子シミュレーションの計算を実行するにあたって，これらの数と同等の規模の分子数で行なった場合，計算時間が現実的ではないほど膨大になるとともに，計算機に搭載しているメモリの容量も圧倒的に足りなくなり，計算自体が実行できないという問題が生じる．また，一方で，対象としている系に対して，特に境界条件を用いず，自由境界条件でシミュレーションを実行した場合，表面効果が生じてしまい，バルク系と比べ，大きな差が生じてしまう．そこで，この表面効果を取り除き，疑似的にバルク系を再現するために，周期境界条件が分子シミュレーションでは一般的に用いられている．周期境界条件の 2 次元的な構造を Fig. 3.4 に示す．中央のセルを基本セル，その周辺のセルをイメージセルと呼ぶ．ここで，ある粒子が運動し，基本セルの外にでて，左側のイメージセルに移動したと仮定する．その際に，Fig. 3.4 に示すように，自動的に右側のイメージセルから同一分子が補充され，密度は一定に保たれる．この仕組みはシミュレーション上で，以下のように実装される．Fig. 3.4 における粒子 j が x 方向に基本セルの外側に飛び出した場合，セル幅を L とすると，x_i を $x_i > L$ ならば $x_i - L$ に，$x_i < L$ ならば，$x_i - L$ とする．これを 3 次元に拡張した場合でも y, z 方向で同様に行う．

3.6　動径分布関数

Fig. 3.5 に示すように，ある粒子に着目して，その周辺に位置する他の粒子の存在確率を，距離 r の関数として定義したものは動径分布関数 (Radial distribution function : RDF) と呼ばれ

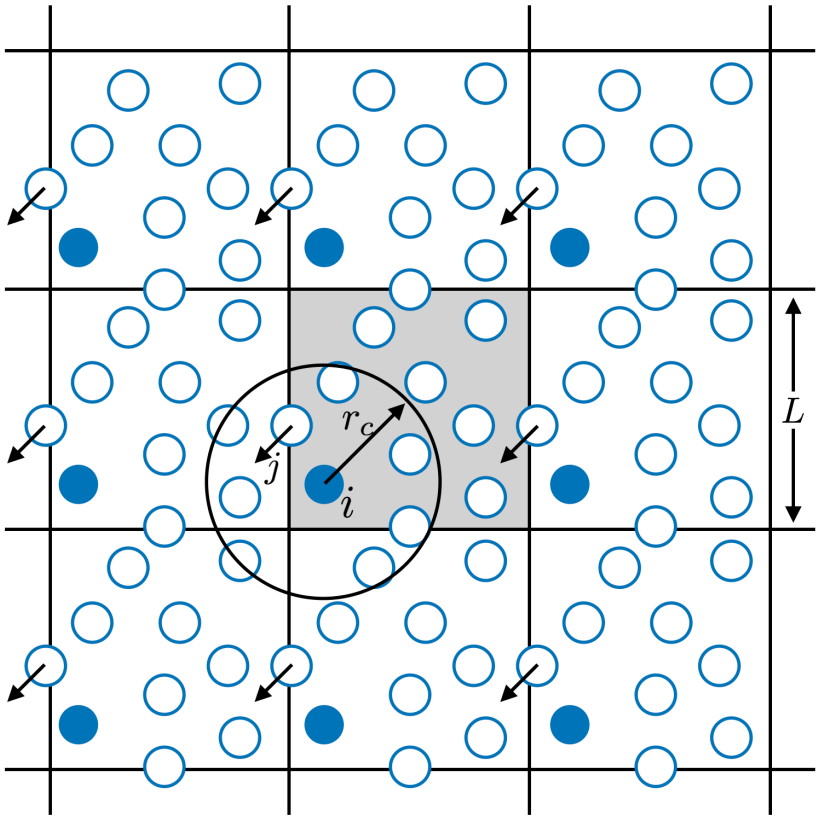


Figure 3.4: 周期境界条件の概念図

ており，液体などの乱れた系の構造等の静的性質を議論する際に最もよく用いられる関数の一つである．ある粒子 i からの距離 r から $r+dr$ の球殻内にある粒子の数を $n(r)$ とすれば，その区間の密度 ρ_i は

$$\rho_i = \frac{n(r)}{4\pi r^2 dr} \tag{3.44}$$

と表される．したがって，ある粒子に対する動径分布関数 $g_i(r)$ は ρ_i を系の平均密度で割ることによって得られ，

$$g_i(r) = \frac{n(r)}{4\pi r^2 dr\rho} \tag{3.45}$$

となる．この $g_i(r)$ について，粒子平均，時間平均をとると，

$$g(r) = \langle g_i(r)\rangle = \frac{\langle n(r)\rangle}{4\pi r^2 dr\rho} \tag{3.46}$$

となり，系の動径分布関数が得られる．

3.7　拡散係数

ある系の動的性質を議論する際によく用いられる自己拡散係数は輸送係数の一つであり，その値 D_S は，分子の平均自乗変位から，

$$D_{\mathrm{s}} = \lim_{t\to\infty}\frac{1}{6t}\langle|r_i(t) - r_i(0)|^2\rangle \tag{3.47}$$

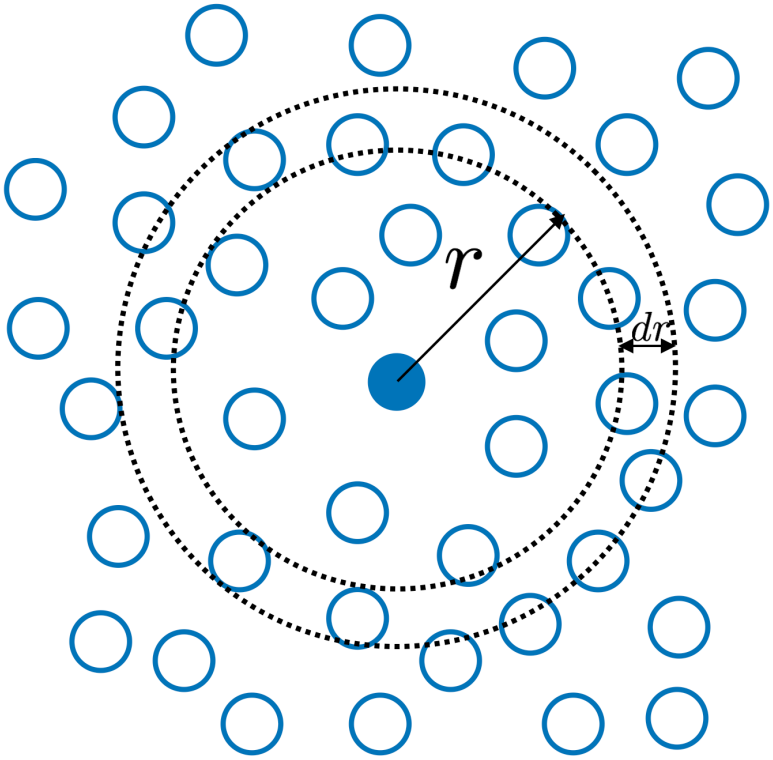


Figure 3.5: 動径分布関数の概念図

のように計算できる．平均自乗変位は一般的に小さな時刻 t_S において，非直線的な増加を示し，大きな時刻 t_L において時間 t に比例した直線的な増加が観測できる．前者は並進の局所的な秤動運動を表し，後者は拡散による長距離の移動を表す．したがって，Eq. (3.47) のように後者の直線的な増加部分の傾きを求めることで自己拡散係数 D_S を得ることができる．

3.8　分子シミュレーションの並列計算

この章では DPD 法を用いて大規模計算を行うための手法について述べる．まずは 2 章で述べた分子シミュレーションで用いられる際のアルゴリズムのうち，特に力の計算について，どのような実装方法が用いられているかを述べる．その後，大規模系に適した実装方法と，DPD 法を用いた時に生じる問題と提案されている解決方法について述べる．

3.8.1　共有メモリシステム

共有メモリシステムは，Fig. 3.6 で示すように，複数のプロセッサが共通のメモリにアクセスできるような環境のことを言う．つまり，全てのプロセッサがメモリ上のデータを参照および更新することが可能である．しかし，このシステム下で，複数のプロセッサが同時に共有されているデータを更新しないようにする必要がある．なぜなら，あるプロセッサがデータの更新を終了しないうちに，他のプロセッサーががデータを参照してしまう恐れがあるからである．これを防ぐために行う操作は排他制御と呼ばれ，複数のプロセッサが同時に同じデータを更新する可能性がある処理において，1 つのプロセッサだけがデータを更新するように制限する [78]．日常的に

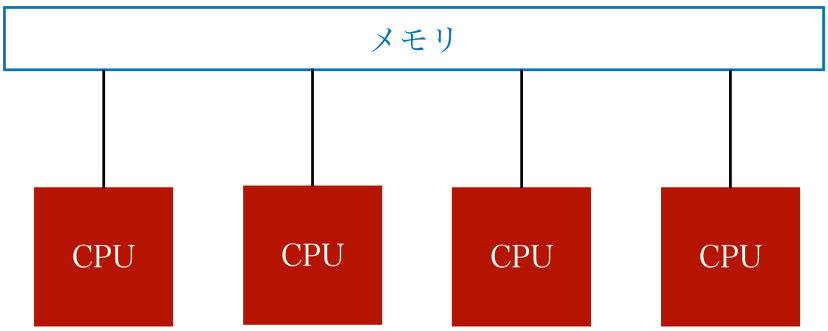


Figure 3.6: 共有メモリシステム

用いられているパソコンなどは，この共有メモリシステム環境で稼働していることが多い．

Open Multi-Processing: OpenMP は共有メモリシステム上での並列化手法であり，並列コンピューティング環境を利用するために用いられる標準化された規格である．スレッドと呼ばれる共有メモリを介して行う処理をノード内に複数立ち上げ並列に処理を実行する．（ノード間は不可）ユーザーが明示的に並列のための指示を与えることで利用できる．具体的にはコード内に指示行の挿入を行うことで並列化が可能となる．これは既存の非並列プログラムに対して，もともとのコードの構造を大きく変えることなく並列化することができ，比較的手軽に利用できる．

3.8.2　分散メモリシステム

分散メモリシステムは，Fig. 3.7 に示すように，共有メモリシステムと異なり，各プロセッサがそれぞれのメモリを保持し，独自にアドレス空間を有している．この場合，他のプロセッサが保持しているメモリのデータを参照および更新する際に，ネットワークを介して，通信を行う必要がある．このような分散メモリシステムを用いることで，共有メモリシステムに比べてプロセッサ数を増やすことが容易になる [78]．そのため，スパコンのような超並列計算機の場合，分散メモリシステムを利用していることが多い．しかし，高い性能を発揮するためには処理時間の大きい通信ができるだけ少なくなるようなプログラムを組む必要がある．

Message Passing Interface: MPI は分散メモリシステム上での並列化手法であり，並列コンピューティング環境を利用するために用いられる標準化された規格である．MPI では，各 CPU のメモリの中身を参照するために，メッセージ・パッシングを必要とする．プロセスと呼ばれる，並列処理の実行単位ごとに別メモリとして実行される処理ごとに，各々が計算を行い，必要に応じて，通信による同期を行う．MPI を用いることで，1 つの CPU で用いるメモリサイズ以上の容量を用いることが可能であり，比較的大規模スケールでの計算が可能である．また，それに伴い，単一 CPU では長時間を必要とする計算が短時間で処理することができる．MPI を用いた並列計算は，並列台数を比較的高くまで上げることが可能であり，これを用いたコードは理論上実行可能な上限の並列数まで動作する．

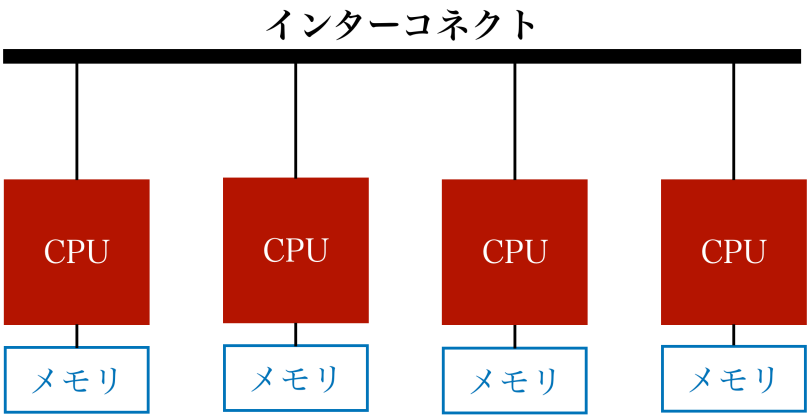


Figure 3.7: 分散メモリシステム

3.8.3　Framework for Developing Particle Simulator(FDPS)

効率的な粒子シミュレーションを行うための様々な手法が提案されている．この大規模並列化を行うためには，ロードバランスのための動的領域分割，領域分割に合わせた粒子交換，ノード間通信の削減と最適化，キャッシュ利用効率の向上，SIMD ユニットの利用効率の向上．GPU 等のアクセラレータへの対応など，多くの方法がある．しかし，これらの処理を行うためには専門的な知識と技術が必要であり，研究者への負担が大きい．そこで，それらの処理の最適化を FDPS フレームワークが担うことで，開発者の計算高速化の技術的なレベルによらず効率的なシミュレーションコードを作成することが可能であり，そういった考えのもと開発されたフレームワークである [79].

コードの動作について

実際に FDPS を用いたコードがどのような流れで動作するのかの概略を記述する．このコードは以下の 4 つのモジュールで構成される．

1. 領域クラス：全プロセスが担当する領域の情報と，領域分割を行う API を有する．
2. 粒子群クラス：全粒子の情報と，プロセスの間での粒子交換を行う API を有する．
3. 相互作用ツリークラス：粒子分布から作られたツリー構造と，相互作用リストを作成する API を有する．
4. ユーザー定義クラス：ある 1 粒子を定義するクラスと，粒子間の相互作用を定義する関数オブジェクトを有する．

また，実際の動作では Fig.3.8 に示すように，これら 4 つのモジュールの間で情報が交換される．動作の流れは，(1) 次の (a)(b) の 2 段階の手順でどのプロセスがどの粒子を運動方程式に従って時間発展させるかを決定する．(a) プロセスの間でロードバランスを取れるようにシミュレーションで扱う空間領域を分割し，各プロセスの担当領域を決定する（領域分割）．(b) 各プロセスが，自分の担当する領域に存在する全粒子の各物理量を保持するように，他のプロセスと各物理量を通信により交換する（粒子交換）．

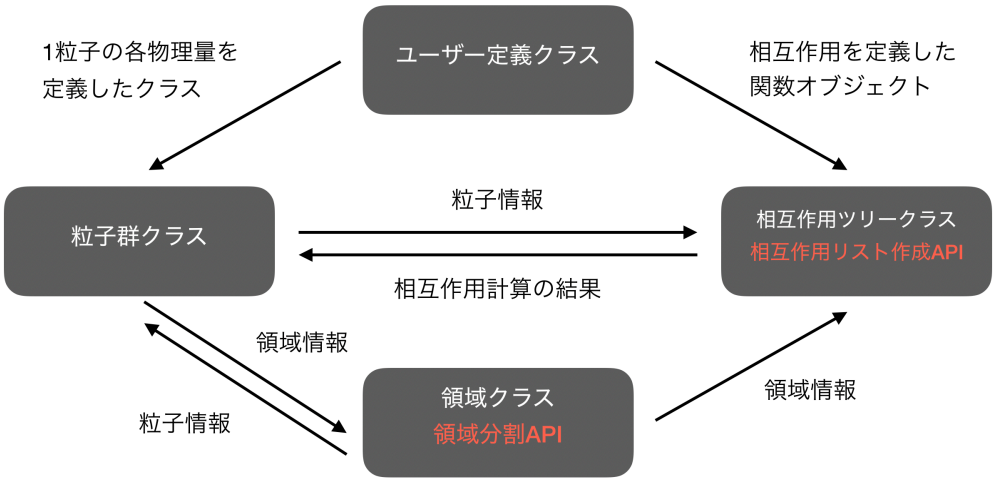


Figure 3.8: 領域分割法

（2）各プロセスは，自分の担当する全粒子の相互作用を計算するのに必要な粒子情報を，他のプロセスとの通信を通して収集し，相互作用リストを作成する（相互作用リストの作成）.

（3）各プロセスは自分の担当する全粒子に対して，相互作用を計算し，粒子速度を更新する（相互作用の計算）.

（4）各プロセスは，自分の担当する全粒子の各物理量とその時間導関数（粒子速度）を利用し，全粒子の時間積分を行う．そして，次の時刻の物理量（位置座標）を求める.

（5）手順（1）に戻る.

FDPS はこれらの流れの処理を OpenMP と MPI のハイブリッド並列化を用いて行う．ノード間はプロセス間通信を用いて MPI 並列化を，ノード内はメモリを共有して OpenMP 並列化を行う．各プロセスは動的領域分割により，ロードバランスが保たれ，それぞれの計算量はほぼ同じなるように自動的に調整される．これによって，空間に疎密が生まれる系に対しても，効率よく並列計算を行うことが可能である．それぞれのプロセスではスレッドを立ち上げ，OpenMP を用いた並列化が行われる．これによって OpenMP を使用しない場合よりも高速に計算を処理することが可能である．立ち上げるプロセス数とスレッド数の組み合わせは複数通りあり，その処理効率は使用する計算機に依存する．同様に計算によって扱う系にも依存するため，予備計算を行い，事前に調べる必要がある.

力の計算について

次に示す 1 は力の計算手法として一般的に分子シミュレーションにおいて広く用いられているものである.

ここで N は粒子数，f_{ij} は粒子 j から粒子 i に作用する力，$F[i]$ は粒子 i が受ける力の合計値を表す．また calculate force from() は () 内の情報に基づき，力を計算する関数である．このアルゴリズムを用いることにより，本来相互作用計算において $O(N^2)$ の計算コストが必要であるとこ

Algorithm 1 作用反作用を利用した力の計算

for $i = 0$ to n -1 **do**
  for $j = i + 1$ to n **do**
    f_{ij} = calculate force from(position[i], position[j])
    $F[i] = F[i] + f_{ij}$
    $F[j] = F[j] - f_{ij}$
  end for
end for

ろを，相互作用の作用反作用の法則を考慮することにより，$O(N^2)/2$ の計算コストで済むため，効率的に力の計算が行えるものである．なお，N は粒子数を，O はオーダを表す．

しかし，系が大規模になり，計算の並列化など，分散メモリシステムでの計算を行う場合，作用反作用の法則を利用した Algorithm 1 は計算 node 間通信を伴うため Algorithm 2 のようになる．

Algorithm 2 作用反作用を利用した力の並列計算

for $i = 0$ to $N - 1$ **do**
  for $j = i + 1$ to N **do**
    f_{ij} = calculate force from(position[i], position[j])
    $F[i] = F[i] + f_{ij}$
    Comunicate $\{F[j] = F[j] - f_{ij}\}$ with other node
  end for
end for

しかし，分散メモリシステムにおいて，計算 node 間通信は非常に大きなコストを要するため，できるだけ少なくすることが望ましい．したがって，次に示すような作用反作用の法則を利用しない Algorithm 3 を用いての計算が一般的である．

Algorithm 3 力の並列計算

for $i = 0$ to N **do**
  for $j = 0$ to N **do**
    if $i \neq j$ **then**
      f_{ij} = calculate force from(position[i], position[j])
      $F[i] = F[i] + f_{ij}$
    end if
  end for
end for

この計算手法を用いると Algorithm 1 の場合と異なり，$O(N^2)$ の計算コストがかかる．しかし，分散メモリシステムでの利用が多い，GPU やスーパーコンピュータ等を用いて計算する場合は，作用反作用の法則を適用する方が，力の計算値を他の node に通信することによるオーバー

ヘッドが大きくなり，計算効率が悪くなってしまうという問題が生じる．そのため，多くの場合では Algorithm 3 を用いた計算が行われる．ここで，Algorithm 3 を用いて，DPD 法の大規模計算を考えてみる．その時，力の計算は次の Algorithm 4 のようになる．

Algorithm 4 DPD 法における力の並列計算

```
for i = 0 to N do
  for j = 0 to N do
    if i ≠ j then
      f_ij = calculate force from(position[i], position[j], RAND[k])
      F[i] = F[i] + f_ij
    end if
  end for
end for
```

ここで，RAND[k] は k 番目の計算 node により生成される疑似乱数を表す．DPD 法において，力の計算の際に疑似乱数を用いることは，2 章に述べた通りである．疑似乱数は疑似乱数生成器の初期値である seed に基づき，生成されるが，各計算 node により，疑似乱数生成器の内部状態が異なるため，生成される疑似乱数は異なる．Algorithm 4 からわかるように，この手法により計算を実行すると，粒子間相互作用が互いに一致せず，作用反作用の法則が成り立たないため，運動量が保存しないという問題が生じる．そのため，DPD 法の並列化による計算の効率化は難しいため，十分に研究されていないのが現状である [80]．そこで Phillips らは，暗号学的ハッシュ関数による疑似乱数生成器を提案した [20]．この手法は次章で述べる暗号化アルゴリズムの性質である拡散性を利用した疑似乱数生成法である．粒子間距離を平文として暗号化を行うことで，各粒子ペア毎に共通した乱数を生成することができる．この手法を用いた計算手法を Algorithm 5 に示す．

Algorithm 5 暗号学的ハッシュ関数を用いた DPD 法における力の並列計算

```
for i = 0 to N do
  for j = 0 to N do
    if i ≠ j then
      f_ij = calculate force from(position[i],position[j],ENCRYPT(position[i],position[j]))
      F[i] = F[i] + f_ij
    end if
  end for
end for
```

ここで ENCRYPT(position[i],position[j]) は () 内の情報を平文として暗号を生成する．暗号は疑似乱数として十分な性質を保持しているため，疑似乱数として取り扱う．Algorithm 5 が示すように，各粒子ペアは共通した疑似乱数を与えられることになる．したがって，作用反作用の

法則を満足したまま，並列化による効率的な計算を行うことができる．本研究では暗号学的ハッシュ関数，The Tiny Encryption Algorithm (TEA) を利用した Algorithm 5 の手法を用いる．

3.9　乱数生成器と暗号学的ハッシュ関数

この章では一般的に DPD 法だけでなくその他の様々な分子シミュレーションで用いられている乱数生成器を紹介する．また，情報の機密性を守る為に広く使われている暗号化の技術について説明する．その後，DPD 法による計算を並列化しようとする際に生じる問題と，暗号学的ハッシュ関数を用いることで，どのようにその問題を解決するかについて述べる．

3.9.1　乱数

乱数とは，予測不可能な数列のことを指す．それは見かけ上のランダムさが求められ，それぞれの数が他の数と無関係であるという特性が求められる．乱数はコンピュータシミュレーション，暗号技術，統計的サンプリングなど，科学技術の様々な分野で広く利用されている．

乱数には 2 つの主要なタイプ，すなわち，真の乱数と疑似乱数が存在する．真の乱数は物理的なランダム現象から生成される一方，疑似乱数は決定論的なアルゴリズムによって生成される．真の乱数は予測不能であるが，大量に迅速に生成するのが困難である．一方，疑似乱数は決定論的アルゴリズムによって生成されるため，同じ初期値（シード）から始めれば，同じ数列が再現される．しかし，良い疑似乱数生成器は生成される数列が見かけ上ランダムであるように設計され，統計的な特性が真の乱数に近い．乱数の生成にはさまざまな方法が存在する．疑似乱数の生成には，線形合同法，MT 法，Xorshift 法などのアルゴリズムが一般的である．乱数の品質は，DIEHARD tests や TestU01 などの統計的テストを用いて評価される．乱数生成器の品質を評価するためには，統計的な均等性，無相関性，特定のパターンの欠如など，さまざまな性質が考慮される．これらのテストは乱数列が求められる品質基準を満たしているかどうかを判断するための重要なツールである．

以上が乱数の基本的な概要である．乱数は多くの科学的，工学的アプリケーションで重要な役割を果たし，その生成と利用は現代の計算と分析の重要な部分を形成している．

3.9.2　乱数生成器

疑似乱数

壺の中に 0 から 9 までの数字が書かれたカードが入っているとする．そこにある人が無作為に壺の中からカードを取り出してカードに書かれた数字を記録し，また壺の中に戻す．その時に得られる数字の列はたとえば以下のようなものである．

$$8, 4, 6, 1, 2, 0, 6, 5, 3, 0, 1, 9, 8, 3, 7 \cdots$$

このような操作を無限に繰り返していく時，記録される数字の列は次のような性質を持っている．

(1) 当確率性

上のような操作により，得られた数列のうち，最初の n 回を観察してこの中で数字 $i(i = 0, 1, 2, \cdots, 9)$ の現れた個数を k_i とすると，相対頻度 k_i/n は $1/10$ に近づく．これを数式で表すと以下のようになる．

$$\lim_{n\to\infty} \frac{k_i}{n} = \frac{1}{10} \tag{3.48}$$

これは，n が十分に大きい時，上の操作によって得られる数字の列の中には，0 から 9 までの 10 個の数字はどれも同じ割合で現れることを示している．このような性質を等確率性と呼ぶ [81].

(2) 無相関性

上のような操作により，例えば，1 番目に取り出した数字は 8 であり，2 番目に取り出した数字は 4 であった．この時，2 番目に取り出された数字は，1 番目に取り出された数字に無関係に取り出された．つまり，ある回数において取り出された数字はその前の回数において取り出された数に無関係である．さらには，1 番目と 3 番目，4 番目についても同様のことがいえ，一般的には i 番目に取り出される数字は，j 番目に取り出される数字によらない．このような性質を無相関性と呼ぶ [81].

一般的にコンピュータシミュレーションや，ゲーム等で用いられる疑似乱数生成器は以上のような性質を持っており，そのような利用目的からするとそれだけで十分と言える．しかし，さらに厳密に分類すると，以下のような性質に分類できる．

(3) 予測不可能性

後述する暗号化の過程においても，乱数は利用されており，暗号化の観点からすると，先述した 2 つの性質だけでは十分とは言えない．暗号を解析する者を仮に「攻撃者」と呼ぶことにすると，過去に生成した疑似乱数列を攻撃者に知られても，次に出力する疑似乱数を攻撃者は言い当てることができない．このように，過去の数列から次の数を言い当てることができない性質を，予測不可能性と呼ぶ [82].

(4) 再現不可能性

過去に生成した数列と同じものを再現するためには，その乱数列そのものを保存しておく以外に方法がないというように，ある乱数列と同じ数列を再現することができないと言う性質を，再現不可能性と呼ぶ．現代において，ソフトウェアだけでは再現不可能性を持つ乱数列を生成することは不可能であり，一般的に疑似乱数生成器と呼ばれる所以はそこである．それは，ソフトウェアを機能せしめているコンピュータそのものが，有限の内部状態しか持たないからである．同じ内部状態からは，必ず同じ出力を生成するので，ソフトウェアにより，生成される数列はいつか必ず繰り返しになる．疑似乱数生成器において，生成する数列が，繰り返しになるまでの長さを周期と呼ぶが，どれ程長くなることがあるとしても，周期は常に有限の値である．そのため，疑似乱数生成器による乱数列は，再現不可能性を持たないと言える [82].

以上の 4 つの性質を振り分けると，Fig. 3.9 のようになる．

Figure 3.9: 乱数性質の分類

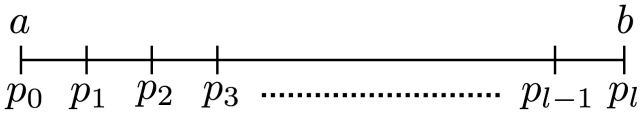


Figure 3.10: 区間 $[a, b]$ の分割

乱数の検定

本研究では後述する暗号学的ハッシュ関数を用いて生成するが，それらの数の集合が疑似乱数として (1)，(2) の性質を満たしているかどうか確認する必要がある．ここでは，それぞれの性質の検定方法について解説する．一般的に分子シミュレーションで用いられている乱数は (1), (2) の性質を満たしたものである．そのため，(3), (4) の性質については考慮しない．

■等確率性の検定　いま，区間 $[a, b]$ に現れる疑似乱数が n 個得られたとする．ここで，Fig. 3.10 のように区間 $[a, b]$ を l 個の部分区間に分割する．この時，i 番目 $(i = 1, 2, 3, \cdots, l)$ の部分区間に現れる個数を $f_i (i = 1, 2, 3, \cdots, l)$ とする．ここで，次のような仮説を立てる．

“生成された N 個の疑似乱数は区間 $[a, b]$ に等確率で現れる”

この仮説は正しいかどうかを検証するために立てられるものであり，帰無仮説と呼ばれている．このような仮説のもとでは，i 番目の部分区間に入る個数は理論的に

$$F_i = n \times \frac{p_i - p_{i-1}}{b - a}, \quad (i = 1, 2, 3, \cdots, l) \tag{3.49}$$

でなくてはならない．ここで F_i を理論度数と呼ぶ．また，実際に各区間に現れた疑似乱数の個数を実現度数と呼ぶ．次の Table 3.1 に各部分区間における実現度数と理論度数をまとめた．

Table 3.1: 各部分区間における実現度数と理論度数

区間 $[a, b]$ の分割	0 p_1	p_1 p_2	p_2 p_3	·····	p_{l-2} p_{l-1}	p_{l-1} p_l	計
実現度数	f_1	f_2	f_3	·····	f_{l-1}	f_{l-2}	n
理論度数	F_1	F_2	F_3	·····	F_{l-1}	F_{l-2}	n

なお，部分区間数は Sturges の公式に基づき，以下の公式によって求める [83].

$$k = \log_2 n + 1 \tag{3.50}$$

なお，n をサンプルサイズ，k を部分区間数とした．この時，各部分区間における理論度数と実現度数の差 $f_i - F_i$ の二乗値 $(f_i - F_i)^2$ の理論度数 F_i に対する比の和は

$$\chi^2 = \sum_{i=1}^{l} \frac{(f_i - F_i)^2}{F_i} \tag{3.51}$$

で得られる．これは理論度数と実現度数の差が大きければ χ^2 の値も大きくなることが分かる．したがって χ^2 の実現値の値によって上述した帰無仮説を捨てるか受け入れるかを χ^2 の確率分布により判断する [81].

χ^2 の確率分布は理論度数が十分大きい時,

$$f_{l-1}(x) = \frac{1}{2^{\frac{l-1}{2}}\Gamma(\frac{l-1}{2})} x^{\frac{l-1}{2}-1} e^{-\frac{x}{2}} \quad (x > 0) \tag{3.52}$$

に示すような自由度 $l-1$ の χ^2 分布に従う．ここで Γ はガンマ関数であり,

$$\Gamma(\frac{l-1}{2}) = \int_0^\infty x^{\frac{l-1}{2}-1} e^{-x} dx \tag{3.53}$$

で与えられる．Eq. (3.52) が示すように，χ^2 分布は自由度 $l-1$ が決まるとその形も決定する．この χ^2 分布において，P を確率関数，$0 < \alpha < 1$ として

$$P\{\chi^2 > \chi_0^2\} = \alpha \tag{3.54}$$

を満たすような定数 χ_0^2 を自由度 $l-1$ の χ^2 分布の α 点と呼び，$\chi_{l-1}^2(\alpha)$ と表す．Eq. (3.52) を用いると Eq. (3.54) は

$$\int_{\chi_0^2}^\infty f_{l-1}(x) dx = \alpha \tag{3.55}$$

とかける．ここで，上述した帰無仮説のもとで，Eq. (3.51) による χ^2 の実現値を求め，それが棄却域に入るかどうかを確認する．ここで，$\chi_{l-1}^2(\alpha)$ より大きい範囲が有意水準 $100\alpha\%$ の棄却域になり，実現値 $\chi_{l-1}^2(\alpha)$ より大きい場合，帰無仮説を棄却し，等確率性は保証されないと言う判断を下す．そして，帰無仮説が棄却されない場合，これを採択し，対象としている疑似乱数に対し，等確率性が満たされていることに矛盾はないとみなす [81]．一般的に，帰無仮説は立証したい事柄の否定とし，その仮説を棄却することで，本来立証したかった仮説である対立仮説を正しいと認めるが，乱数検定においては，帰無仮説を“疑似乱数列は一様である”と仮説し，その帰無仮説を採択することにより等確率性を認めるという手法が多く用いられる．

■無相関性の検定　いま，以下のように n 個の疑似乱数列が生成されたとする．

$$x_1, \quad x_2, \quad x_3, \quad \cdots, \quad x_n$$

k 個ずらしたものと 2 つを 1 組として以下のように考える．

$$(x_1,\, x_{1+k}), \quad (x_2,\, x_{2+k}), \quad (x_3,\, x_{3+k}), \quad \cdots (x_{n-1},\, x_{n-1+k}), \quad (x_n,\, x_{n+k})$$

これら各組みにおける相関係数 $r_{xy}^{(k)}$ は，

$$r_{xx} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i x_{i+k} - \bar{x}^2}{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.56}$$

で与えられる．ただし，

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad x_{n+j} = x_j, \quad j = 1, 2, \cdots, k$$

とする．ここで Eq. (3.56) は一般的な形式である

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.57}$$
$$\left(\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i\right)$$

によって与えられており，$r_{xy}^{(k)}$ は $-1 < r_{xy}^{(k)} < 1$ の値をとり，-1 に近ければ，負の相関，1 に近ければ，正の相関，0 に近ければ相関がないとみなす．また，Eq. (3.56) の $r_{xx}^{(k)}$ は系列相関係数と呼ばれ，この値により，疑似乱数列の無規則性の検定を以下のように行う．等確率性の検定方法と同様に，区間 $[a, b]$ に現れる疑似乱数 n 個に対し，帰無仮説を“生成された n 個の疑似乱数は無規則に並んでいる”とすると，この仮説のもとで，$Z^{(k)}$ を次のように定義する．

$$Z^{(k)} = \frac{\sqrt{n} r_{xy}^{(k)}}{\sqrt{13}} \tag{3.58}$$

この式において，n が十分大きい時，任意の K を持ってきても，平均値が 0，分散が 1 の正規分布に従うことが知られている．ここで有意水準を定めると，棄却域を求めることができ，$Z^{(k)}$ が棄却域に入るかどうかを確認する．等確率性の検定と同様，帰無仮説が採択されれば，生成された n 個の疑似乱数は，無規則に並んでいることに矛盾はないとみなす [81].

乱数の統計的検定：NIST SP 800-22rev1a

NIST 乱数検定では，各検定ごとに p-value が算出される．p-value とは，先述したとおり，検定で出力される統計量の正規分布もしくは，カイ二乗分布において，それよりも偏った統計量位が発生する確率を表現したものである．NIST 乱数検定では優位水準 1 %が用いられており，p-value< 0.01 の時に良い乱数ではないと判定する．

各検定では，複数の標本系列（NIST 乱数検定では 1000 を推奨）に対して検定を行い，(i)p-value の一様性，(ii)p-value が 0.01 より大きくなる割合の 2 通りについて評価する．(i) について，算出された-p-value が区間 $[0, 1)$ で一様に分布しているかどうかを調べるために，$[0, 1)$ を 10 の区間に分割し，分割した区間毎の頻度が一様になっているかどうかをカイ二乗検定によって得られた p-value が 0.0001 以上ならば，良い乱数列であると判定する．また，(ii) では，標本の数を m とした時，0.01 以上となる p-value の数の割合が

$$0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{m}} \tag{3.59}$$

の範囲に入っている場合は，良い乱数として判定する．なお，個々のテストの具体的な詳細は書籍 [84, 85] を参照されたい．

■Frequency Test（一様性の検定） この検定は，生成された乱数列内で 0 と 1 が等しい割合で出現するかどうかを評価する．検定統計量は，

$$T_n = \sum_{i=1}^{n} \frac{2Z_i - 1}{\sqrt{n}} \tag{3.60}$$

で定義される．ここで，帰無仮説の下で n が十分に大きい場合，T_n は標準正規分布 $N(0,1)$ で近似できる．$2Z_i - 1$ は，平均が 0，分散が 1 の確率変数への変換を行う．標本から得られた T_n の値 t_n について，n が大きい場合に $N(0,1)$ で近似し，p 値を求める．$p < 0.01$ で帰無仮説を棄却する．

■Block Frequency Test（ブロック内一様性の検定） この検定は，一様性の検定をブロックに適用した形で，乱数列を M ビットのブロックに分割し，各ブロック内の 1 の出現割合が $1/2$ であるかを検証する．検定統計量は，

$$Q_{FB} = 4M \sum_{l=1}^{[n/M]} \left(\sum_{i=1}^{M} \frac{Z_{(l-1)M+i}}{M} - \frac{1}{2} \right)^2 \tag{3.61}$$

で定義される．ここで，Q_{FB} は帰無仮説のもとで，M および n（ただし $M < n$）が十分大きい場合の確率分布を使用して，p 値が 0.01 未満の場合に帰無仮説を棄却する片側検定を行う．

■Cumulative Sums Test（累積和検定） Cumulative Sums（累積和）検定は，累積和 $W_k = \sum_{i=1}^{k}(2Z_i - 1)$ の振る舞いを，帰無仮説下で期待される振る舞いと比較して乱数列のランダムネスを検定する．検定統計量は $\max_{1 \le k \le n} \frac{|W_k|}{\sqrt{n}}$ を使用し，n が十分大きい場合にその統計量の確率分布が既知であることを利用する．また，反転した累積和 $W'_k = \sum_{i=1}^{k}(2Z_{n-i+1} - 1)$ に基づく検定も実施され，これにより乱数列の前半と後半のランダムネスの偏りを検出する．W_k と W'_k を用いることで，乱数列の始端と終端の両方でランダムネスの異常を検出することが目的である．

■Runs Test（連の検定） 連とは，同一記号（0 または 1）の連続する部分列を指す．乱数列における 0 と 1 の出現に偏りがないかを検定する．検定統計量は，隣接するビットが同一であれば $U_i = 0$，異なれば $U_i = 1$ として，

$$T_n = \frac{Q_n - 2n\hat{p}_n(1 - \hat{p}_n)}{2\sqrt{2n}\hat{p}_n(1 - \hat{p}_n)} \quad \left(\hat{p}_n = \frac{\sum_{i=1}^{n} Z_i}{n} \right) \tag{3.62}$$

と定義する．ここで $n \to \infty$ のとき，検定統計量 T_n は標準正規分布 $N(0,1)$ に収束する．有意水準 α に対し，両側検定で棄却域 d を求め，$T_n \ge d$ の場合に帰無仮説を棄却する．

■Longest Run Test（ブロック内最長連検定） 乱数列内の 0 または 1 の最長連の長さを検定する．ここでは 1 の最長連に焦点を当てる．n ビットの乱数列を M ビットずつのブロックに分割し，各ブロック内の 1 の最長連の長さを求める．これを $N = \left[\frac{n}{M}\right]$ 回繰り返し，1 の最長連の分布を得て，カイ二乗適合度検定を行う．帰無仮説のもとで期待される確率分布との乖離を検定する．

■Binary Matrix Rank Test 乱数列の各部分が特定の線形従属性を持たないかを検定する．$J \times K$ の行列を $\left[\frac{n}{JK}\right]$ 個作成し，各行列の階数を求める．行列の階数分布をカイ二乗適合度検定により検定する．

■Discrete Fourier Transform (Spectral) Test　乱数列における値の出現に周期性がないかを離散フーリエ変換を用いて検定する．周波数領域 $[0,1]$ を n 等分し，各周波数点での離散フーリエ変換を行い，その絶対値を求める．$n \to \infty$ のとき，帰無仮説のもとでの確率分布が $N(0,1)$ に収束することを利用して検定を行う．

■Overlapping Template Matching Test　特定のビットパターンの出現回数をカウントし，その分布が帰無仮説のもとで期待される分布と異なるかを検定する．M ビットのブロックに分割し，各ブロック内でのパターンの出現回数をカイ二乗適合度検定で評価する．

■Non-Overlapping Template Matching Test　Overlapping Template Matching Test と同様だが，パターンが一致した後はそのパターンの終わりから再検索を開始することで重複を避ける．カイ二乗適合度検定により検定を行う．

■Maurer's "Universal Statistical" Test　Maurer のユニバーサル統計検定は，乱数列 $\{Z_i; 1 \leq i \leq n\}$ 内の同じパターンの出現間隔の自然さを評価する．パターンの長さを M_B とし，n ビットを M_B ビットごとに $\frac{n}{M_B}$ 個のブロックに分割する．最初の N_1 ブロックを初期値として使用し，残り $N_2 = \frac{n}{M_B} - N_1$ ブロックを検定に利用する．各 2^{M_B} パターンが初めて出現したブロック番号を記録し，同じパターンが再び出現するたびにそのブロック番号を更新する．N_2 ブロックに対し，それぞれのブロックでパターンが前に出現した最後のブロック番号との差に基づいて対数値を計算し，これらの和を検定統計量とする．理論値との比較により，出現間隔に自然なばらつきがあるかどうかを検定する．

■Approximate Entropy Test　乱数列 $\{Z_i; 1 \leq i \leq n\}$ の各ビットパターンの出現頻度に基づくエントロピーを計算し，m ビットおよび $(m+1)$ ビットのパターンでのエントロピーの差を用いて乱数列の予測可能性を検定する．パターンの出現頻度に基づくエントロピーの差が帰無仮説の下で期待される値から逸脱しているかどうかを評価する．

■Random Excursions Test and Random Excursions Variant Test　これらの検定は乱数列のランダムウォーク特性を評価する．乱数列を -1 と 1 の数列に変換し，途中までの和 $\{W_k = \sum_{i=1}^{k}(2Z_i - 1); k = 1, 2, \ldots, n\}$ を計算する．Random Excursions Test では，W_k の 0 に戻るサイクルごとに特定の値（例えば $-4, -3, \ldots, 4$）を取る回数の分布を検定し，Random Excursions Variant Test では，全サイクルを通じて特定の値を取る回数を検定する．各テストは，帰無仮説の下での期待される分布と実際の分布を比較し，逸脱があれば帰無仮説を棄却する．

■Serial Test　Serial 検定では，保持された順序での $\{Z_i; 1 \leq i \leq n\}$ 内の m ビットの全ての 2^m 個のパターンが等しい割合で出現するかを評価する．この検定では，Z_n の後に最初の $(m-1)$ 個のビットを追加して

$$Z_1, Z_2, \ldots, Z_n, Z_1, Z_2, \ldots, Z_{m-1} \quad (\{Z_i; 1 \leq i \leq n+m-1\} \text{ とする}) \tag{3.63}$$

と拡張し，重複を許して m ビットの各パターンの出現回数をカウントする．カイ二乗適合度検定を用いて各パターンの一様性を検定する統計量 Q_{SER}^m を定義し，$m-1$ および $m-2$ ビットのパターンについても同様に Q_{SER}^{m-1}，Q_{SER}^{m-2} を計算する．これらの統計量の差 $\nabla Q_{SER}^m = Q_{SER}^m - Q_{SER}^{m-1}$，

$\nabla^2 Q_{SER}^m = Q_{SER}^m - 2Q_{SER}^{m-1} + Q_{SER}^{m-2}$ を用いて帰無仮説のもとでの変動を評価し，$n \to \infty$ でカイ二乗分布に従うことを利用して検定を行う．

■Linear Complexity Test　線形複雑性検定は，$\{Z_i; 1 \leq i \leq n\}$ が線形関係から見て十分に複雑であるかを検定する．n ビットを M ビットごとに $N = \frac{n}{M}$ 個のブロックに分割し，各ブロック内で

$$Z_i = c_1 Z_{i-1} + c_2 Z_{i-2} + \cdots + c_{L_j} Z_{i-L_j}, \quad i \geq L_j \tag{3.64}$$

という線形関係が成立するかどうかを調べる．ここで L_j はブロック j における最小の線形スパンであり，c_l は 0 または 1 を取る．各ブロックに対して得られた L_j の値を用いて統計量 T_j を計算し，T_j を $(K+1)$ 個のクラスに分類する．クラスごとの度数 ν_k を用いて

$$Q_{LC} = \sum_{k=0}^{K} \frac{(\nu_k - N p_k^{LC})^2}{N p_k^{LC}} \tag{3.65}$$

という統計量を求め，これが自由度 K のカイ二乗分布に従うことを利用して検定を行う．帰無仮説を棄却する基準は $p < 0.01$ であり，この条件を満たす場合，乱数列は線形複雑性の観点から不自然であると結論付ける．推奨されるパラメータは $n \geq 10^6$，$500 \leq M \leq 5000$，$N \geq 200$ である．

真の乱数

真の乱数とは，その生成に関与する物理的プロセスが本質的にランダムであるため，予測不可能な数値のことを指す．これらの数値は一見すると無規則であり，それぞれの数は他の数と無関係である．真の乱数の生成は物理的なランダム現象から生じる．一部の方法は電子的ノイズ（例えば，熱雑音や放射雑音），無秩序な物理的運動（例えば，気体分子の衝突やブラウン運動），あるいは量子力学的現象（例えば，放射性崩壊）を利用する [86]．これらの方法は，本質的にランダムで予測不可能な結果を生み出す．生成された乱数が均等に分布しているかどうかを保証するためには，しばしばポストプロセシングステップが必要となる．これには，適切なアルゴリズムを用いてデータを処理し，結果の数値が期待される確率分布に従うようにするという作業が含まれる．

真の乱数の主な利点はその予測不能性である．物理的なプロセスから生成される乱数は，理論的にはその後の数値を予測することは不可能である．後述する暗号論においてはこの予測不能性が極めて重要となる．しかし，真の乱数には欠点も存在する．真の乱数生成器は通常，特殊なハードウェアが必要であり，大量の乱数を迅速に生成するのは困難であることが多い [87]．

3.9.3 暗号学的ハッシュ関数

暗号とは

暗号は長い歴史を持ち，戦時下では軍事技術の一つとして発達してきた．インターネットが普及し，多くの情報が溢れている現代においても，その存在意義は大きく，身の回りの様々なところで多く利用されている．基本的な考え方としては，第三者が他者による通信情報を得る機会があっても，特別な知識なしでは内容を得ることができないように機能している．ここで説明のために，Fig. 3.11 に示すように，送信者，受信者，盗聴者を考える．送信者が「ABC」という内容のメッセージを受信者に向けて送信したとする．このメッセージはたくさんのコンピュータや

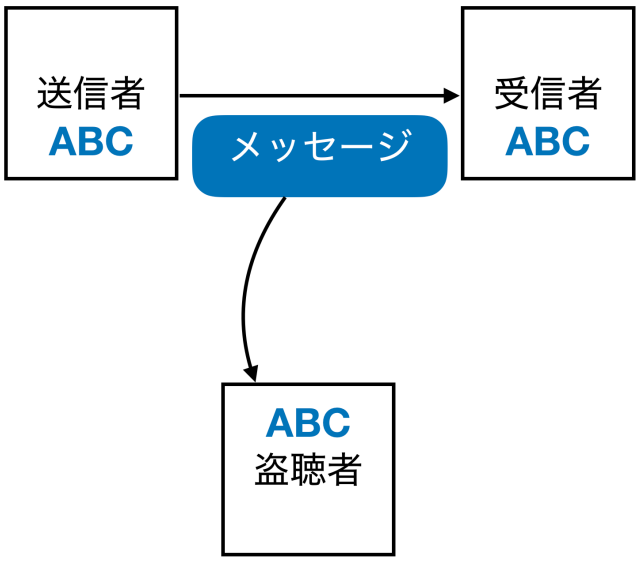


Figure 3.11: 盗聴者が情報を得る例

Figure 3.12: 暗号化の概念図

Figure 3.13: 複合の概念図

通信機器を経由しながら受信者に送られることになるが，その途中で悪意のある盗聴者が許可なくメッセージの中身を確認する可能性がある．このように何の対策もせずに二者間でやりとりが行われると，意図しない第三者に情報が漏れてしまうという危険がある．そこで，送信者は Fig. 3.12 のように元々の文章である平文「ABC」を暗号化し，この暗号文を受信者に向けて送信する．その後，受信者は Fig. 3.13 のように暗号文を複合し，元の平文に戻す．こうすることで，Fig. 3.14 のように第三者が何らかの理由でメッセージを入手したとしても，その内容そのものを入手することはできない．このように暗号はメッセージに含まれる情報の機密性を守る上で非常に重要な役割を果たす．

ここで第三者が暗号文を得た時に元の平文が推測できないように十分暗号化されているかどうかが重要であり，その暗号化の度合いを表すものを，暗号強度と呼ぶ．暗号強度の要因として，拡散性と予測困難性が挙げられる．拡散性とはあらゆる平文に対しても，暗号化により，もとの偏りのある記号の分布を一様に拡散し，分布させる性質のことである．また，予測困難性とは，平文と暗号文の 2 つから，暗号化アルゴリズムを予測することがどれほど困難かをいうものとする．暗号強度は上述したように疑似乱数生成器の性能にも大きく影響され，疑似乱数の生成は暗号文作成の上で重要な役割を果たしている．しかしながら，どれほど優れた暗号化の処理を行なっても，完全に解読されない暗号文を作ることは現代の技術では極めて困難である．急速に発達する計算技術の成果により，数年前の暗号文が解読され，相対的に暗号強度が下がる場合も少なくない．したがって，暗号強度が大きいものとは，多数の平文と暗号文のサンプルがあっても，現代最高の計算技術で解読するのに非常に長い時間がかかるような強い暗号アルゴリズムのことを言う．

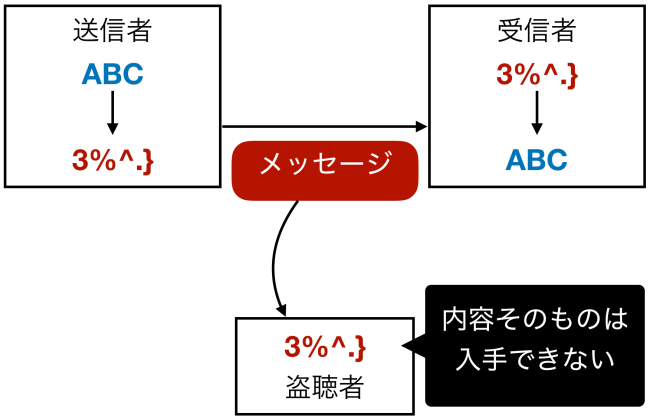


Figure 3.14: 盗聴者が情報を得ても内容は入手できない例

The Tiny Encryption Algorithm

これまでに様々な暗号が開発され，暗号化による演算負荷が大きいが，暗号強度の大きいもの，または，その演算負荷が小さい代わりに，暗号強度があまり大きくないもの，そのほかに，復号を考慮せず，平文に対して不可逆変換を施すものまで，多種多様である．

本研究では現存する暗号アルゴリズムの中でも，最も演算負荷が小さく，高速かつ効率的に暗号化が行えるアルゴリズムの一つである，The Tiny Encryption Algorithm: TEA を用いる．

TEA は David Wheeler と Roger Needham により開発され，1994 年に発表された暗号アルゴリズムで，複数回の繰り返しビット演算を使用する Feistel 構造を有する暗号である [88]．Feistel 構造とはブロック暗号の代表的な構造のことで，実装コストを抑えるため，ラウンドと呼ばれる暗号化の 1 ステップを複数回繰り返すような構造になっている．Fig. 3.15 に 3 ラウンドの Feistel 構造を示す．Feistel 構造では入力（平文）64bit を 2 つの 32bit に分割し，それぞれ別々に処理される．上の 2 つの四角は 32bit に分割された入力（平文）を表す．下の 2 つの四角はこの暗号アルゴリズムを通して得られた出力（暗号文）を表す．また図中に 3 つあるサブ鍵は各ラウンドにおける暗号化の鍵である．Feistel 構造ではラウンドごとにサブ鍵が必要となる．またラウンド関数とは，右側にある 32bit の情報とサブ鍵を用いて左の 32bit を暗号化するためのビット列を生成する関数である．そして，図の左側では，各ラウンドごとにラウンド関数の出力と左の 32bit の情報との XOR がとられ，その結果が暗号化された左の 32bit の情報となる．一方，右の 32bit の情報はそのまま出力になるため，1 ラウンド目では変換されない．その次のラウンドを開始するにあたって右の 32bit の情報と左の 32bit の情報をいれかえて，再び同じような処理を行う．なお，右の 32bit の情報，左の 32bit の情報はそれぞれ簡易的に「右」「左」と表す [82].

1. ラウンドへの入力を左と右に分ける．
2. 右をそのまま，そのラウンドにおける右の出力とする．
3. 右をラウンド関数に送る．
4. ラウンド関数は右とサブ鍵を用いて，ランダムに見えるビット列を計算する．
5. 得られたビット列と，左との XOR を計算した結果を，そのラウンドでの暗号化された左とする．
6. 右と左を入れ替えて，定めたラウンド回数になるまで，同様の処理を繰り返す．

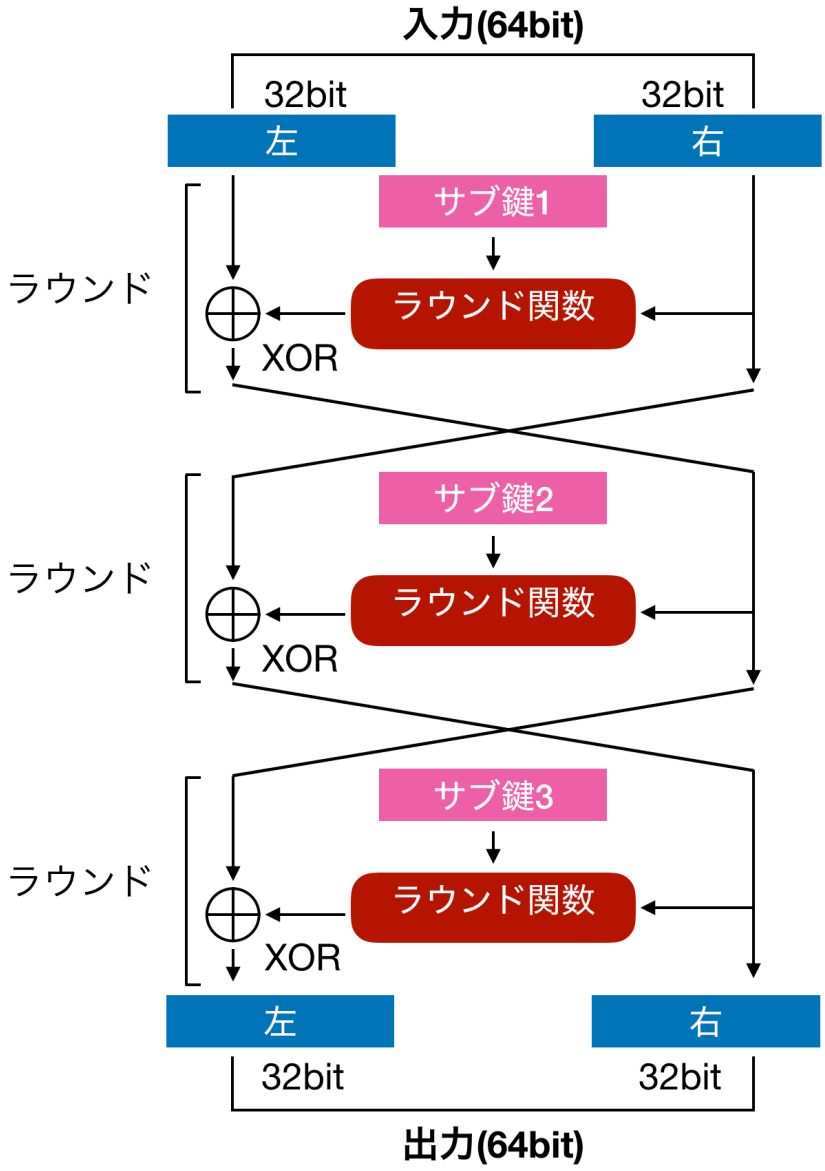


Figure 3.15: Feistel 構造 (3 ラウンド)

Figure 3.16 に TEA の暗号化の仕組みを示す．図中の A は定数であり，各ラウンドごとに SUM に足され，ラウンド関数において用いられる．上述したように，内部は Feistel 構造を有しており，TEA の場合は 1 ラウンド内で，右，左の 32bit の情報をラウンド関数に通す．また，ラウンド関数では右の 32bit の情報に二つの異なる方法でビットシフトが施されたものと，ビットシフトが施されていないものを，それぞれサブ鍵 1，サブ鍵 2，SUM の和をとる．次に，それらの XOR 演算が施され，左の 32bit の情報との和が取られる．さらに，右，左の 32bit の情報が入れ替えられ，同様の処理を行う．この一通りの流れを 1 ラウンドとして計算する．

暗号化で用いられる TEA では計 32 回のラウンドを繰り返すことが推奨されているが，わずか 6 回のラウンドで Avalanche 効果が得られることが報告されている [89]．Avalanche 効果とは入力のわずかな変化を出力において大きな変化に変換する効果のことをいい，十分な拡散性を持つことをいう．一方で TEA の予測困難性は現代において十分とは言えない．ラウンドごとに用いられるサブ鍵の大きさが十分でなく，暗号強度が低いため，暗号学的ハッシュ関数としては実用

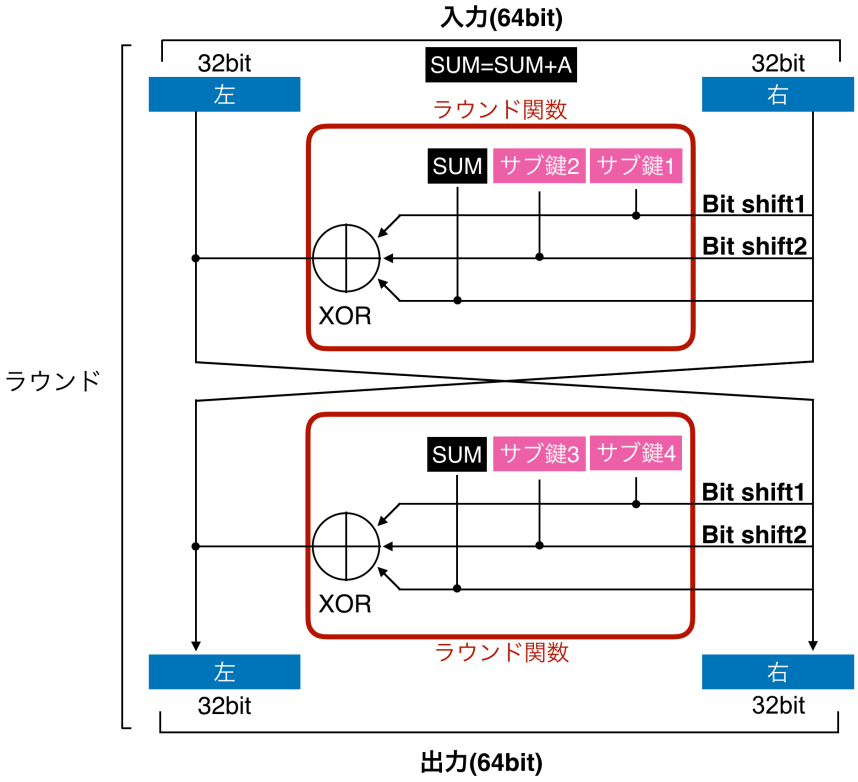


Figure 3.16: TEA の構造（1 ラウンド）

的ではない．しかし，本研究においてはこのアルゴリズムを一種の疑似乱数生成器として利用するため，予測不可能性については議論しない．また，その他の暗号アルゴリズムに比べて，非常に演算負荷の小さいアルゴリズムであるため，分子シミュレーションのような演算負荷の大きい計算に実装するには最適なアルゴリズムであると言える．

疑似乱数生成器としての TEA

一般的にシングルコアにおける DPD 法の計算では乱数生成器が用いられている．しかし，3 章で述べたように，ただ単純に DPD 法を並列化し，大規模な系の計算を行おうとした時，各粒子間ペアに対応した力のランダム項における不一致が原因となり，運動量が保存しなくなるという問題が生じる．そこで，暗号学的ハッシュ関数を乱数生成器として用いることで，この問題を解決する．Phillips らが提案した暗号学的ハッシュ関数を用いた疑似乱数生成器は TEA を用いており，これは数ある暗号学的ハッシュ関数の中でも，高い効率性を示している [20]．また，Zafar らの研究により，DIEHARD の乱数テストおよび，NIST 乱数検定 [85] の合格数と計算効率の双方を考慮し，TEA のラウンド回数は 8 回が推奨されている [71]．そのため，一般的にラウンド 8 回の TEA が広く用いられている．以後，ラウンド回数 n 回の TEA を TEA(n) と呼ぶ

DPD 法による計算のうち TEA(8) は DPD 法による計算全体の 40% を占めており，暗号化の計算コストは依然として高い．そこで，本研究ではこのラウンド回数をさらに減らすことにより，DPD 法により再現する系がどのように変化するのかを調査する．ラウンド回数を $1/n$ にすることで，暗号化に要する計算コストもまた $1/n$ になることが報告されている [71] ので，DPD 法で用いるのに必要最低限の回数までラウンド回数を減らすことで，計算の効率化が期待できる．

さらに，Fig. 3.16 で示す入力の 64bit を代わりに 128bit にし，2 つの 64bit の情報として暗号

化する場合にどうなるのかということも，同様に検証する．この場合，出力される 64bit の暗号文（疑似乱数）も 2 つ与えられる．そのため，一度の TEA の利用で 32bit の場合に比べ，倍の量の情報を暗号化することが可能である．以降では，これらを区別するために，32bit ごとに暗号化する本来の TEA を 32TEA(n) と，64bit ごとに暗号化する TEA を 64TEA(n) と記す．DPD 法で用いられる確率変数は互いに独立であることと，一様に分布していることが要求されていることがわかっているので [76]，その 2 つの性質に関して，32TEA(n) 及び，64TEA(n) の各ラウンド回数で得られた疑似乱数を上述した 2 つの検定により評価する．

第 4 章

散逸粒子動力学法へ与える暗号学的ハッシュ関数型乱数生成手法によるの影響の調査と改善

本章では従来よりも高速に乱数を生成するための手法を提案する．ここでは以下の流れに沿って，順に説明する．1．暗号学的ハッシュ関数を用いた従来の手法を軽量化し，より高速に乱数を生成できるか実験を行う．また，その手法により生成した乱数を用いた場合，DPD シミュレーションによる系はどの様に再現されるのかを詳細に解析する．2．実験 1 で行った解析結果をもとに，更に効率的な乱数生成器を設計，検討する．ここでは，主に乱数シード値に現れる乱雑性の評価とその乱雑性をどの様に用いて，乱数を生成するのかを検討する．また実験 1 と同様，DPD 法に適用する実験も行う．3．実験 2 で設計した乱数生成手法に対して，より厳しい条件のもとでの検証を行う．どのような場合にこの手法は適用できなくなるのか，また一般的に DPD 法で再現されるどのような系に対しても利用可能であるかどうかを検証する．4．本章で提案する新規乱数生成手法を従来の手法と比較し，計算コストの観点でどれだけの利点があるのかを検証する．5．ここまでで紹介した新規乱数生成手法の適用例として，先行研究と同様の系を再現し，問題なくシミュレーションが実行できるかどうかを確認する．

4.1 暗号学的ハッシュ関数を利用した乱数生成手法を評価するための計算方法

暗号学的ハッシュ関数を用いた従来の手法を軽量化し，より高速に乱数を生成できるか実験を行う．また，その手法により生成した乱数を用いた場合，DPD シミュレーションによる系はどの様に再現されるのかを詳細に解析する．

Groot と Warren の研究 [76] を参考にし，密度を $\rho^* = 3$，力の散逸項に関連するパラメータ γ^* を $\gamma^* = 6.75$ と設定した．ランダム項に関連するパラメータ σ^* は Eq. (3.15) より $\sigma^* \approx 3.67$ である．疑似乱数の精度は実際に計算で生成される個数において，上述した等確率性と無規則性の 2 つを満たしておく必要があると考えられる．例えば，領域分割法において，分割する領域サイズを一辺の大きさを 1.5 程度とすると，密度 $\rho^* = 3$ において一つのボックスに約 256 個の DPD 粒

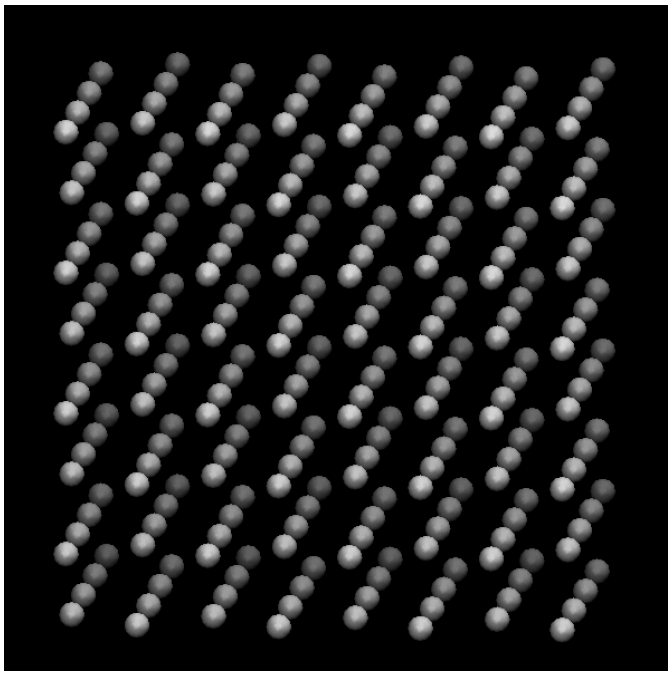


Figure 4.1: DPD 法により再現した水の系の初期配置

子が入ると考えられるため，その場合，疑似乱数生成の回数は各 step 毎に 32640 個であり，これは一般的に用いられる大きさである．そのため，本研究ではこの大きさを採用し，粒子数は 256 個とした．また，疑似乱数生成器として，32TEA(32) ，32TEA(16) ，32TEA(8) ，32TEA(6) ，32TEA(4) ，32TEA(3) ，32TEA(2) ，32TEA(1) ，64TEA(32)，64TEA(16)，64TEA(8)，64TEA(6)，64TEA(5)，64TEA(4)，64TEA(3)，64TEA(2)，64TEA(1) をそれぞれ用いた．ここで，32TEA(n)，64TEA(n) は 4 章で示したように n 回のラウンド回数を持ち，2 つの 32bit または 64bit の平文を使用する TEA を表す．

次にどのように暗号化を行うのかについて説明する．ここで，以下では，一様分布に従う疑似乱数を一様乱数，正規分布に従う疑似乱数を正規乱数と呼ぶ．まず DPD 法では正規乱数を必要とし，それを得るために Box-Muller 法 [90] を用いる．Box-Muller 法は一様乱数を，正規乱数に変換するものであり，

$$G = \sqrt{-2\ln(R_1)}\cos 2\pi R_2 \tag{4.1}$$

によって変換される．ここで，R_1，R_2 はそれぞれ独立で区間 $(0, 1)$ に生成される一様乱数であり，G は正規乱数のことを示す．Eq. (4.1) からわかるように，正規乱数を得るためには 2 つの独立した一様乱数が必要である．そのため，32TEA(n) を用いる場合は 2 度，64TEA(n) を用いる場合は 1 度用いればよいことがわかる．

平文は 64bit，double 型の粒子間距離，x, y 成分をそれぞれ使用し，32TEA(n) を用いる場合は x, y 成分それぞれに 32TEA(n) を 1 度づつの計 2 回暗号化を行う．また，64TEA(n) を用いる場合は粒子間距離，x, y 成分の両方 128bit を用いて 64TEA(n) を 1 度暗号化を行う．それぞれを疑似コードで表すと，Algorithm8, 7 のようになる．

Algorithm 6 32TEA(n) を用いた暗号化のアルゴリズム

```
union Uni
  double HIRA
  unsigned int hira[2]
  unsigned long long ANGO
end union
uni.HIRA = distance
32TEAn(uni.hira[0],  uni.hira[1])
rand = (double)uni.ANGO
```

Algorithm 7 64TEA(n) を用いた暗号化のアルゴリズム

```
union Uni
  double HIRA[2]
  unsigned long long ANGO[2]
end union
uni.HIRA[0] = distanceX
uni.HIRA[1] = distanceY
64TEAn(uni.HIRA[0],uni.HIRA[1])
rand0 = (double)uni.   ANGO[0]
rand1 = (double)uni.   ANGO[1]
```

distance, distanceX, distanceY は平文となる粒子間距離を，rand, rand0, rand1 は TEA(n) による暗号化後に得られる疑似乱数を表す．**union** は共用体の使用を表しており，変数を同時に 2 つ以上の型で保持することができる．ここでは，double 型, unsigned int 型, unsigned long long 型で保持し，double 型 64bit を unsigned int 型 32bit の 2 つで表せるようにした．また，TEA(n) による暗号化後に共用体の double 型で直接 rand に返した場合，double 型の指数部と仮数部の組み合わせの関係上，同じ数字が 2 通り以上で表現できてしまうため，ここでは unsigned long long 型を double 型にキャストして疑似乱数として rand に返すようにした．Algorithm8, 7 の最後の行の “rand = (double)uni.ANGO” はそのことを表す．

4.2 乱数性質の評価

乱数は各粒子ペアごとに生成され，粒子数を 256 とすると各計算 step ごとに 32640 個の疑似乱数が生成されることになる．計算は 10000 step 行われ，合計で 32640 個の乱数を生成した．以下では step = 30 における疑似乱数個についての結果を示す．また，その後，得た疑似乱数に対して行った等確率性の検定，無規則性の検定について述べる．

4.2.1 Mersenne Twister 法を用いた疑似乱数の性質

疑似乱数生成器の一つである MT は，乱数の性質が良く周期も非常に長いため，最もよく用いられるうちの一つである．MT を用いて区間 $(0, 1)$ に 32640 個生成した．区間 $(0, 1)$ を 16 等分

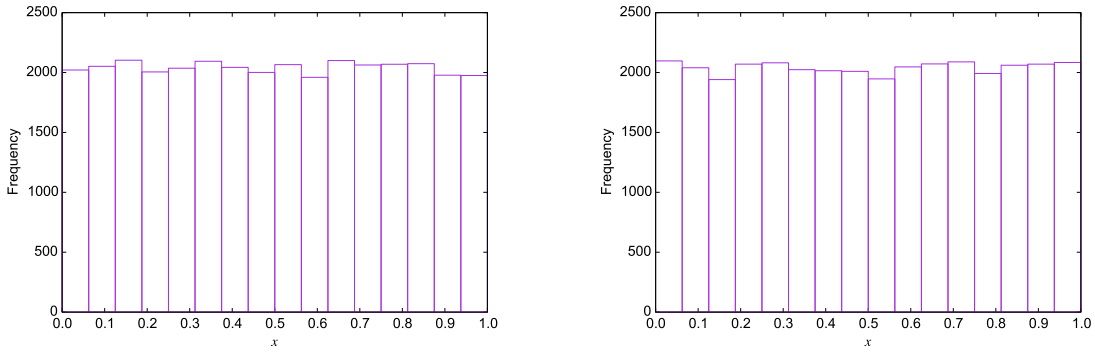


Figure 4.2: MT 法により生成された一様乱数のヒストグラム

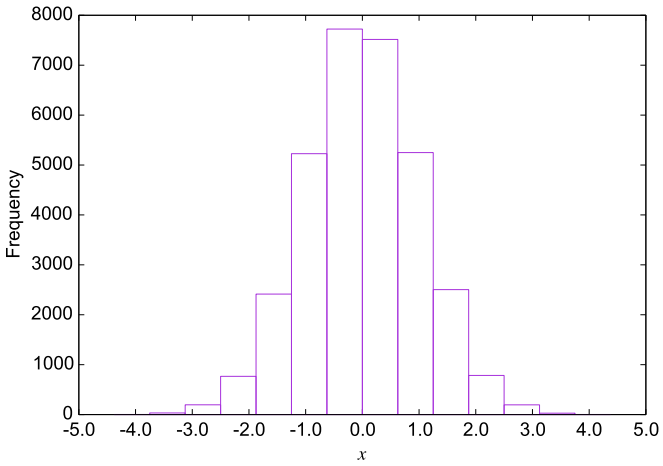


Figure 4.3: MT 法により生成された正規乱数のヒストグラム

し，それぞれの区間に現れた個数を Frequency としてヒストグラムに表した．

Fig. 4.2 に示すように，MT 法を用いて，一様分布に従う疑似乱数が生成できた．また Fig. 4.3 に示すように，２つの一様乱数と Box-Muller 法を用いて正規乱数が得られることが確認できた．

4.2.2　暗号学的ハッシュ関数を用いた疑似乱数の性質

暗号学的ハッシュ関数に与えた平文は粒子間距離の x 成分，y 成分であり，それぞれの分布は Fig. 4.4 に示すようなものとなる．

Figs. 4.5，4.7 に示すように，本来のラウンド回数で用いられた 32TEA(32)，また，先行研究で推奨されている回数である 8 回のラウンド回数で用いられた 32TEA(8) の両方において，偏りのある分布である粒子間距離から一様乱数に変換することが確認できた．また Fig. 4.6，4.8 に示すように，32TEA(32), 32TEA(8) によるそれぞれの一様乱数と Box-Muller 法を用いて，正規乱数が得られることが確認できた．

Fig. 4.9 に示すように，推奨されているラウンド回数 8 回を下回り，さらに最低回数である 32TEA(1) を用いても，一様乱数に変換されていることが確認できた．これは計算コストが，推奨されている 32TEA(8) の 1/8 で，同様に一様乱数を生成できたことがわかる．また Fig. 4.10 に示すように，32TEA(1) による一様乱数を用いて，正規乱数が得られることが確認できた．

(a) (b)

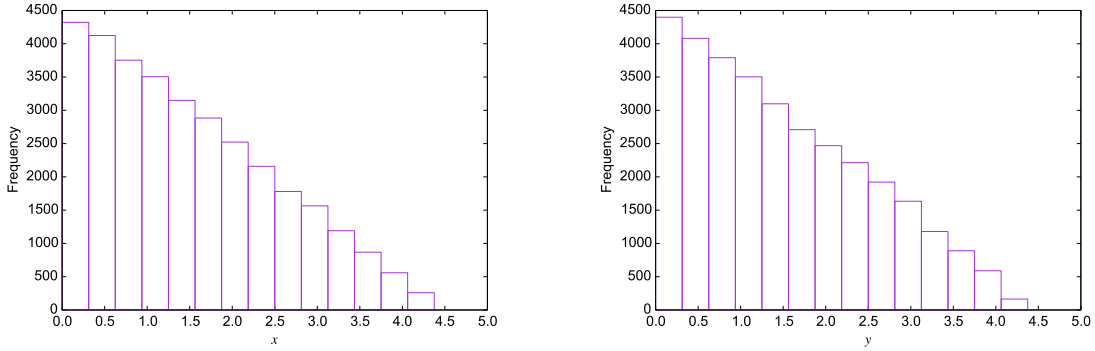


Figure 4.4: 水粒子相対距離のヒストグラム (a)x 成分, (b)y 成分

(a) (b)

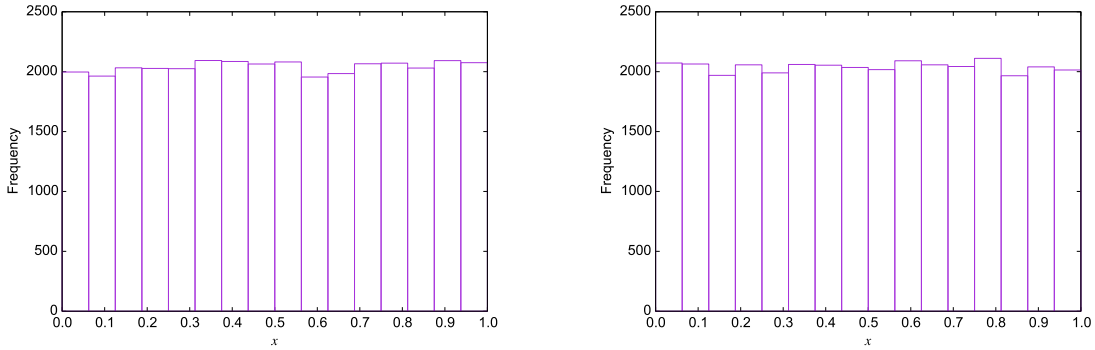


Figure 4.5: 32TEA(32) により生成された一様乱数のヒストグラム

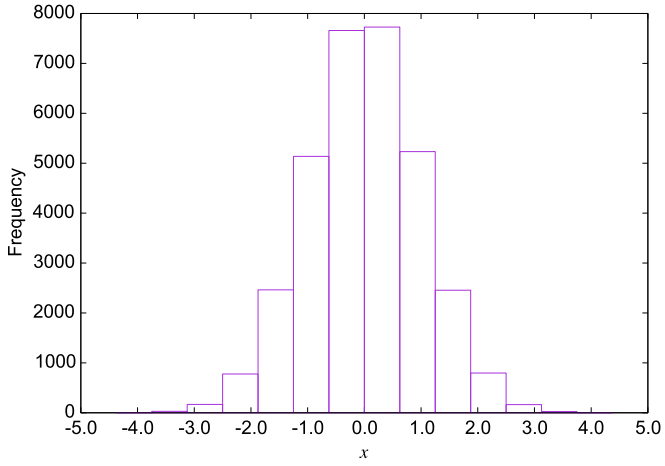


Figure 4.12: 64TEA(32) により生成された正規乱数のヒストグラム

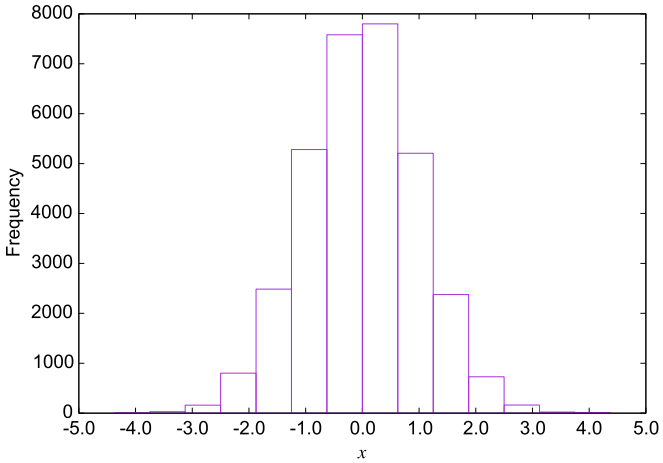


Figure 4.6: 32TEA(32) により生成された正規乱数のヒストグラム

(a)　　(b)

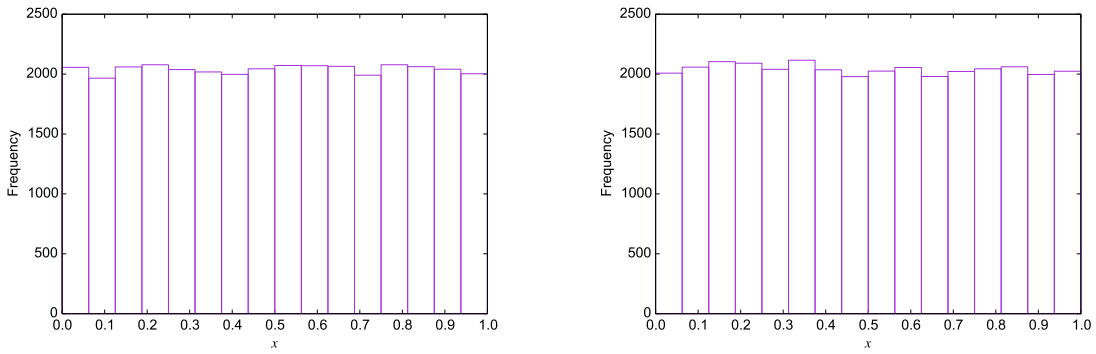


Figure 4.7: 32TEA(8) により生成された一様乱数のヒストグラム

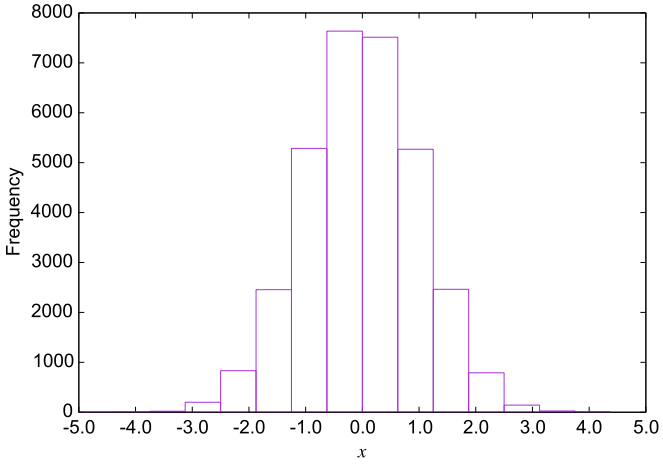


Figure 4.14: 64TEA(3) により生成された正規乱数のヒストグラム

Figs. 4.11 4.13 に示すように，64TEA(32), 64TEA(3) を用いても，一様乱数が得られた．また Figs. 4.12 4.13 に示すように，64TEA(32), 64TEA(3) による一様乱数を用いて正規乱数を生成することが確認できた．

Figs. 4.15 4.17 に示すように，ラウンド回数が 2 回以下で乱数の等確率性が著しく悪くなるこ

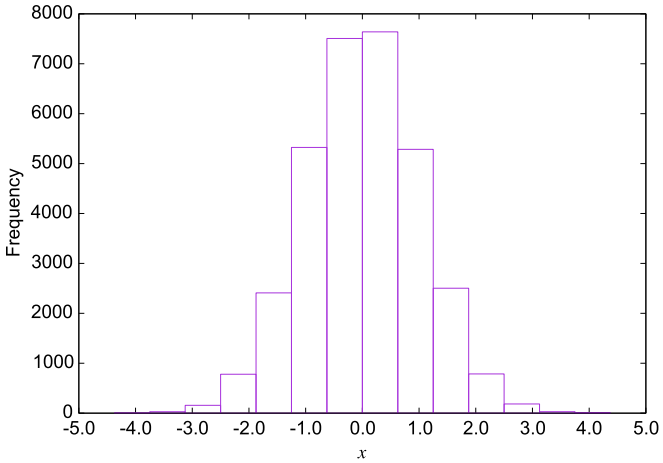


Figure 4.8: 32TEA(8) により生成された正規乱数のヒストグラム

(a)　　(b)

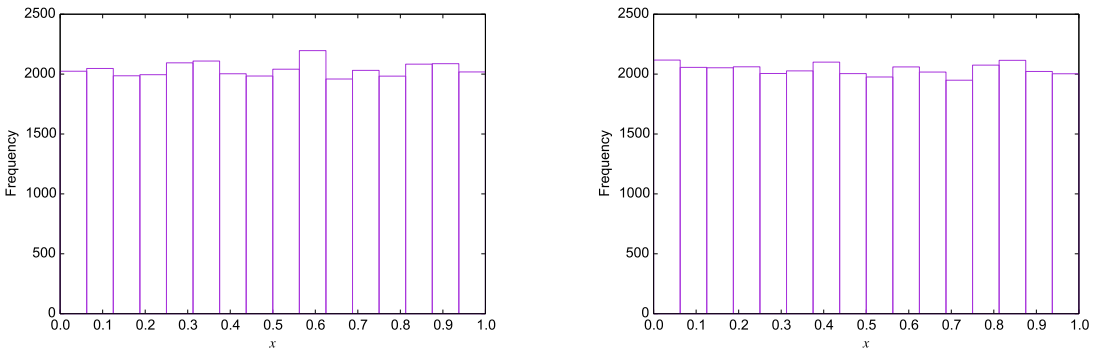


Figure 4.9: 32TEA(1) により生成された一様乱数のヒストグラム

とがわかった．ラウンド回数 2 回において，生成された一様乱数の両方において中央値である 0.5 付近の数の生成が低下した．Fig. 4.17 から，ラウンド回数が 1 回において，区間 (0,1) の中に生成されない区間があることが確認した．また，その代わりに，ある一部の区間に強く偏って生成されていることがわかった．正規分布に従って得られる数の分布に関しては，ラウンド回数 2 回のものからは，正規分布に従った数の分布が得られたが，ラウンド回数 1 回の数の分布から得られたものは正規分布には従わないことがわかった．

4.2.3　各手法により得られた疑似乱数の性質検定

等確率性の検定

各暗号化手法を用いて生成された一様乱数に対して，等確率性の検定を 300 回行った．その結果を Fig. 4.19 4.20 に示す．横軸にラウンド回数を，縦軸に検定により得られる値を Value として示した．Criterion で表す図の赤い線は検定基準であり 4 章の各種検定手法において述べたとおり，帰無仮説を採択するかを判断する基準である．等確率性の検定では，有意水準 5% として棄却域は (24.996,∞) である．この場合において帰無仮説は先述したものと同様に“疑似乱数列は一様である”とする．この値を上回ると帰無仮説は棄却され一様乱数であることを保証しないことを示す．32TEA(n)1，32TEA(n)2 は 32TEA(n) で生成した一様乱数の両方の 300 回の検定結果

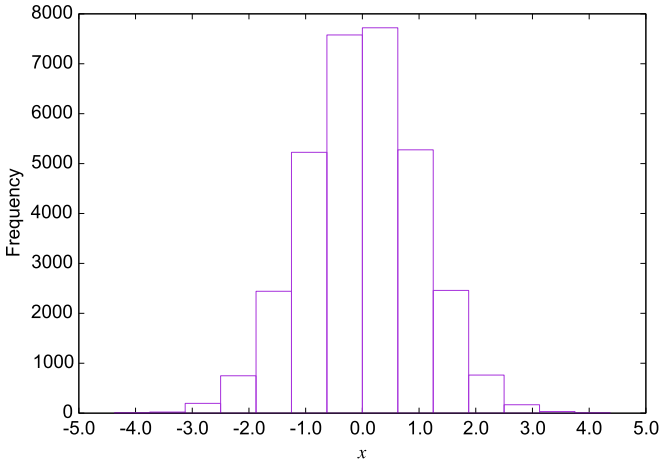


Figure 4.10: 32TEA(1) により生成された正規乱数のヒストグラム

(a)　(b)

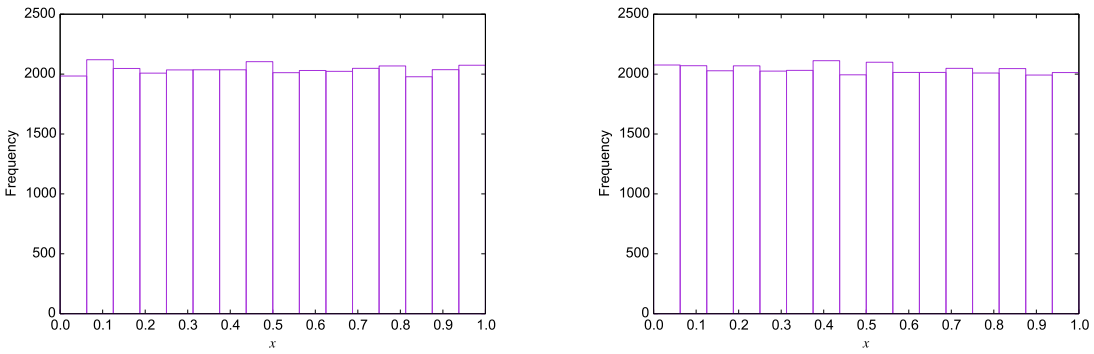


Figure 4.11: 64TEA(32) により生成された一様乱数のヒストグラム

の平均を示し，各ラウンド回数ごとにプロットで示した．

32TEA(n) について，上記のヒストグラムで検証した一様乱数は，すべてのラウンド回数で等確率性が保証されていることが分かった．なお，300 回の内，棄却域に入らなかった一様乱数の回数は 32TEA(32)1 で 289 回，32TEA(32)2 で 288 回，32TEA(16)1 で 281 回，32TEA(16)2 で 285 回，32TEA(8)1 で 286 回，32TEA(8)289 回，32TEA(6)1 で 285 回，32TEA(6)2 で 286 回，32TEA(4)1 で 288 回，32TEA(4)2 で 285 回，32TEA(3)1 で 284 回，32TEA(3)2 で 292 回，32TEA(2)1 で 278 回，32TEA(2)2 で 281 回，32TEA(1)1 で 284 回，32TEA(1)2 で 279 回であった．

64TEA(n) について，上記のヒストグラムで検証した一様乱数は，3 回以上のラウンド回数で等確率性が保証されていることが分かった．なお，300 回の内，棄却域に入らなかったものは 64TEA(32)1 で 289 回，64TEA(32)2 で 286 回，64TEA(16)1 で 286 回，64TEA(16)2 で 288 回，64TEA(8)1 で 279 回，64TEA(8)286 回，64TEA(6)1 で 283 回，64TEA(6)2 で 289 回，64TEA(5)1 で 283 回，64TEA(5)2 で 277 回，64TEA(4)1 で 284 回，64TEA(4)2 で 284 回，64TEA(3)1 で 272 回，64TEA(3)2 で 277 回，64TEA(2)1 で 0 回，64TEA(2)2 で 39 回，64TEA(1)1 で 0 回，64TEA(1)2 で 0 回であった．なお，参考に MT 法による乱数の検定結果を示す．棄却域に入らなかった一様乱数の回数は，一様乱数 1 で 286 回，一様乱数 2 で 287 回

(a)　(b)

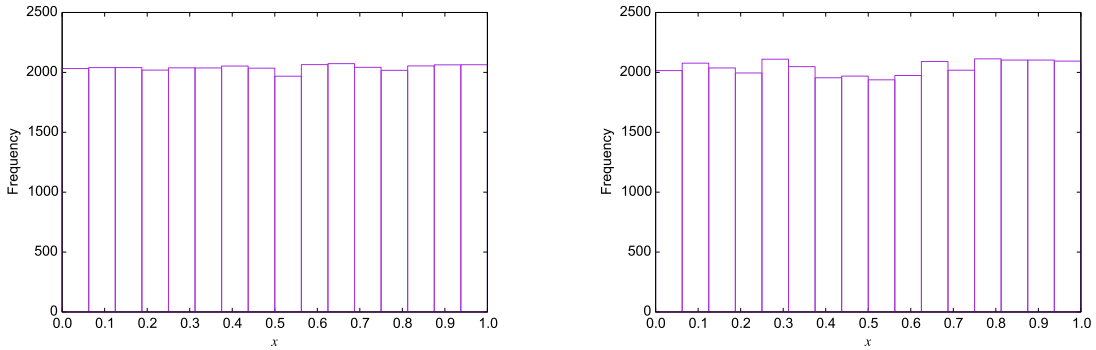


Figure 4.13: 64TEA(3) により生成された一様乱数のヒストグラム

(a)　(b)

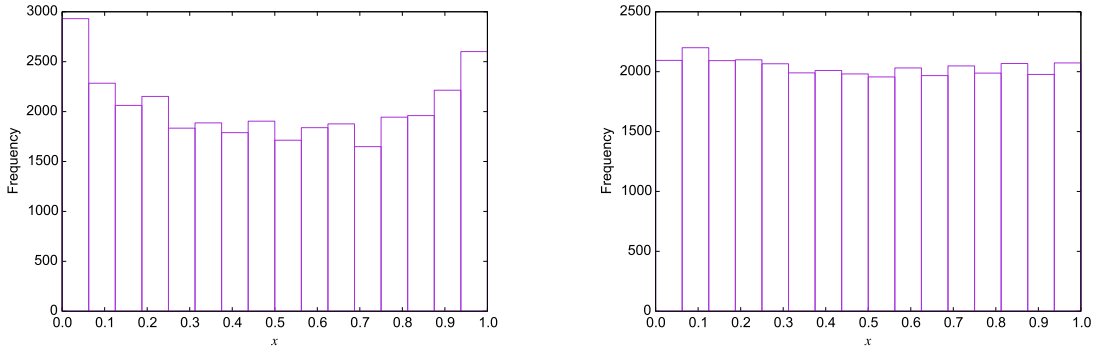


Figure 4.15: 64TEA(2) により生成された一様乱数のヒストグラム

であった．ラウンド回数が 2 回以下のもの，つまり，64TEA(2)1，64TEA(2)2，64TEA(1)1，64TEA(1)2 においては帰無仮説“疑似乱数列は一様である”が棄却され，等確率性が保証されないことがわかった．

4.2.4　無規則性の検定

ここでは，特定のある同じ粒子のペアに対して，生成される一様乱数が互いに時間相関があるかどうかについて検定した．次に示す Table 4.1 では step $= t$ における粒子ペア $(i, j) = (a, b)$ で生成される粒子間距離 $|x_{ij}|$，$|y_{ij}|$ 及びそれらを元に生成される一様乱数 r$and0_{ij}$,r$and1_{ij}$ と，step $= t + 1$，step $= t + 2$，step $= t + 3$，step $= t + 4$，step $= t + 9$，step $= t + 99$ における粒子ペア $(i, j) = (a, b)$ で生成される粒子間距離 $|x_{ij}|$,$|y_{ij}|$ 及びそれらを元に生成される一様乱数 r$and0_{ij}$,r$and1_{ij}$ が相関を持つかどうかを調べた．4 章で述べた通り，0 に近いほど互いに相関が無く，1 に近いほど正の相関を持つことを示している．なお，ここで用いた暗号学的ハッシュ関数は 32TEA(32) であり，4 章で述べた，k 個ずらしの組み 9500 個によって計算されたものである．

Table 4.1 より，1 番目と 10 番目においても $|x_{ij}|$，$|y_{ij}|$ は正の相関があると言える．しかし，最も正の相関が見られる，1 番目と 2 番目の $|x_{ij}|$，$|y_{ij}|$ を用いて生成された暗号文，つまり，生成された一様乱数は相関係数が 0 に近く，時間方向に相関がないことがわかる．次の Figs. 4.21, 4.22 では，元の平文に時間相関が最も大きい，1 番目と 2 番目の平文を元に生成した暗号文，つまり，

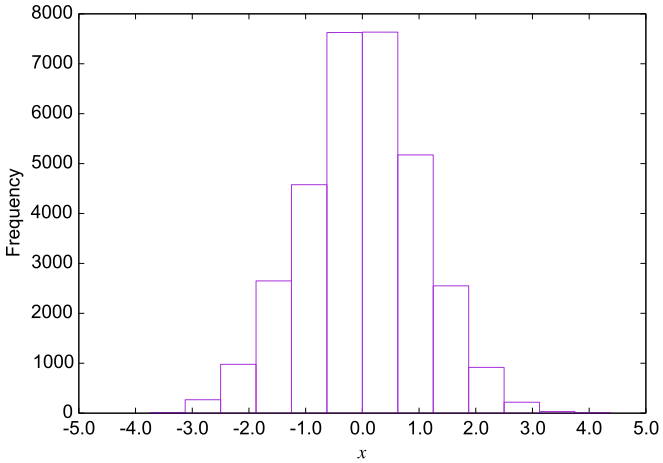


Figure 4.16: 64TEA(2) により生成された正規乱数のヒストグラム

(a)　(b)

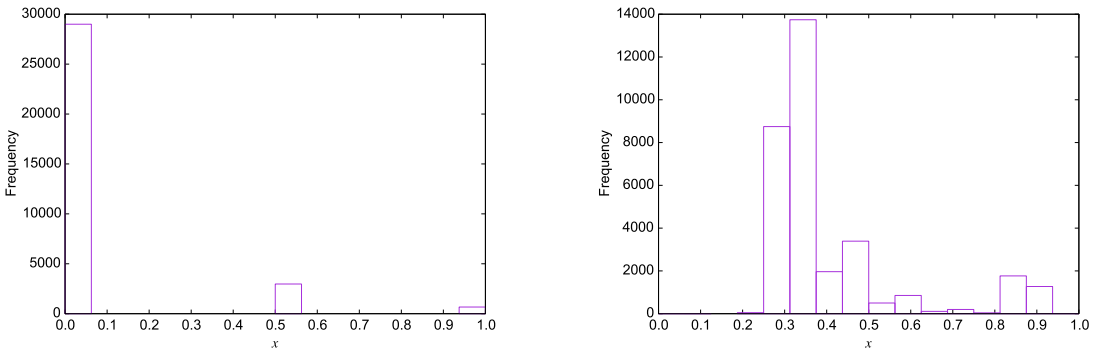


Figure 4.17: 64TEA(1) により生成された一様乱数のヒストグラム

Table 4.1: 相関係数一覧

	1, 2 番目	1, 3 番目	1, 4 番目	1, 5 番目	1, 10 番目	1, 100 番目
x 成分	0.894	0.853	0.816	0.781	0.642	0.031
x 暗号文	0.009	-0.004	0.008	0.012	0.012	0.014
y 成分	0.925	0.890	0.859	0.831	0.714	0.056
y 暗号文	-0.003	0.006	-0.002	0.003	-0.001	-0.009

一様乱数の時間相関について無規則性の検定を行った．Criterion で表す図の赤い線は検定基準であり 4 章の各種検定手法において述べたとおり，帰無仮説を採択するかを判断する基準である．無規則性の検定では，有意水準 5% として棄却域は (-∞,-1.96), (1.96,∞) である．この場合において帰無仮説は先述したものと同様に“生成された n 個の疑似乱数は無規則に並んでいる”とする．棄却域に入ると帰無仮説は棄却され疑似乱数は無規則性を保証しないことを示す．等確率性の検定と同様，横軸に 32TEA(n) 及び 64TEA(n) のラウンド回数，縦軸に検定基準を示した．

32TEA(n) の場合，すべてのラウンド回数で 1 番目と 2 番目の一様乱数に対して，無規則性が保証されていることが確認できた．

64TEA(n) の場合，ラウンド回数が 1 回と 2 回の時を除くその他のラウンド回数で，1 番目と

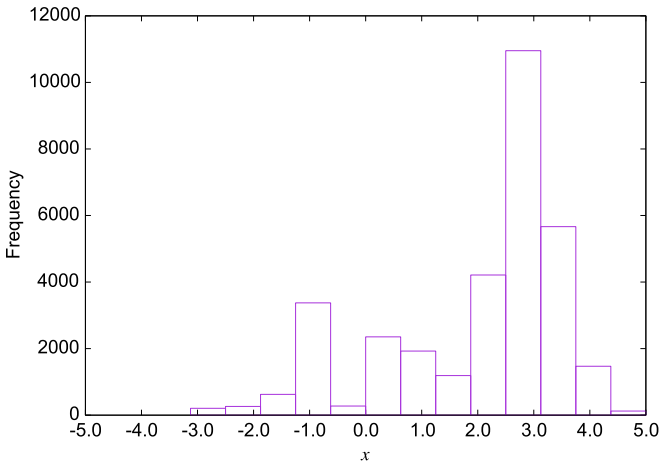


Figure 4.18: 64TEA(1) により生成された正規乱数のヒストグラム

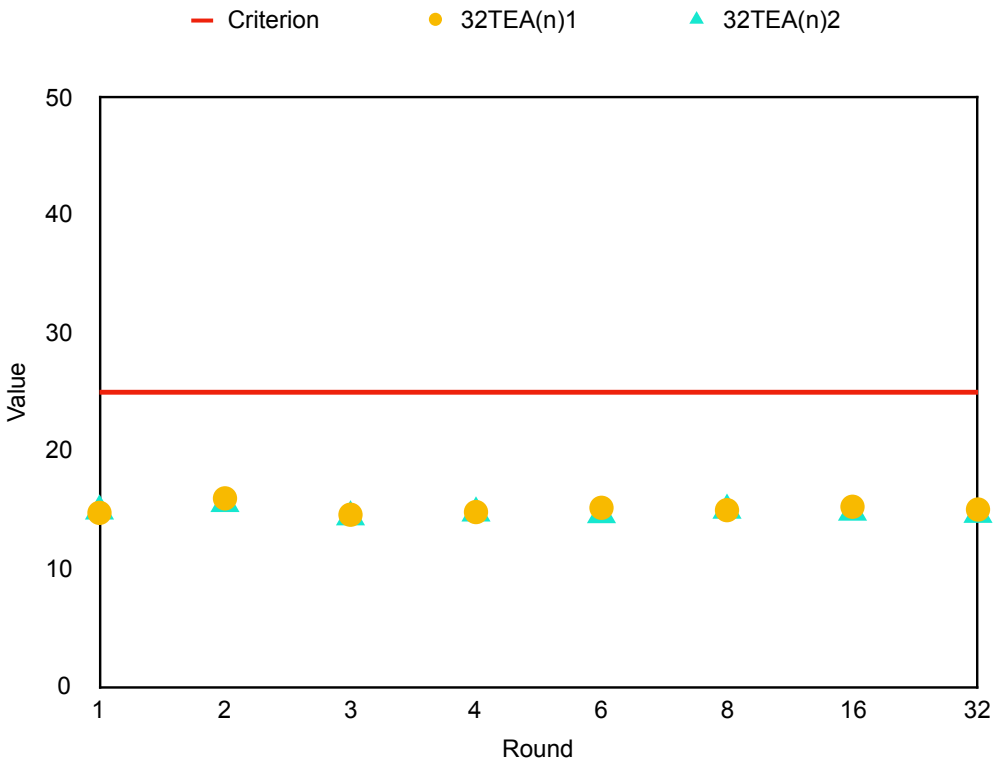


Figure 4.19: 32TEA(n) の等確率性検定

2 番目の一様乱数に対して，無規則性が保証されていることが確認できた．ラウンド回数が 1 回と 2 回の場合，64TEA(n)1，64TEA(n)2 はそれぞれ，64TEA(1)1 の時，10.139, 64TEA(1)2 の時，16.629, 64TEA(2)1 の時，0.279 と，そのすべてが検定基準を上回り，帰無仮説“生成された n 個の疑似乱数は無規則に並んでいる”は棄却され，無規則性が保証されないことがわかった．

4.2.5　正規分布の性質

ここで，各暗号化手法による正規乱数の平均値，分散を 32TEA(n), 64TEA(n) のそれぞれにおいてラウンド回数ごとにまとめた．32TEA(n) について，Figs. 4.23，4.24 に, 32TEA(n) について，Figs. 4.25, 4.26 に示す．なお，MT 法で得られた正規分布の平均値は-0.0003，分散は 0.999

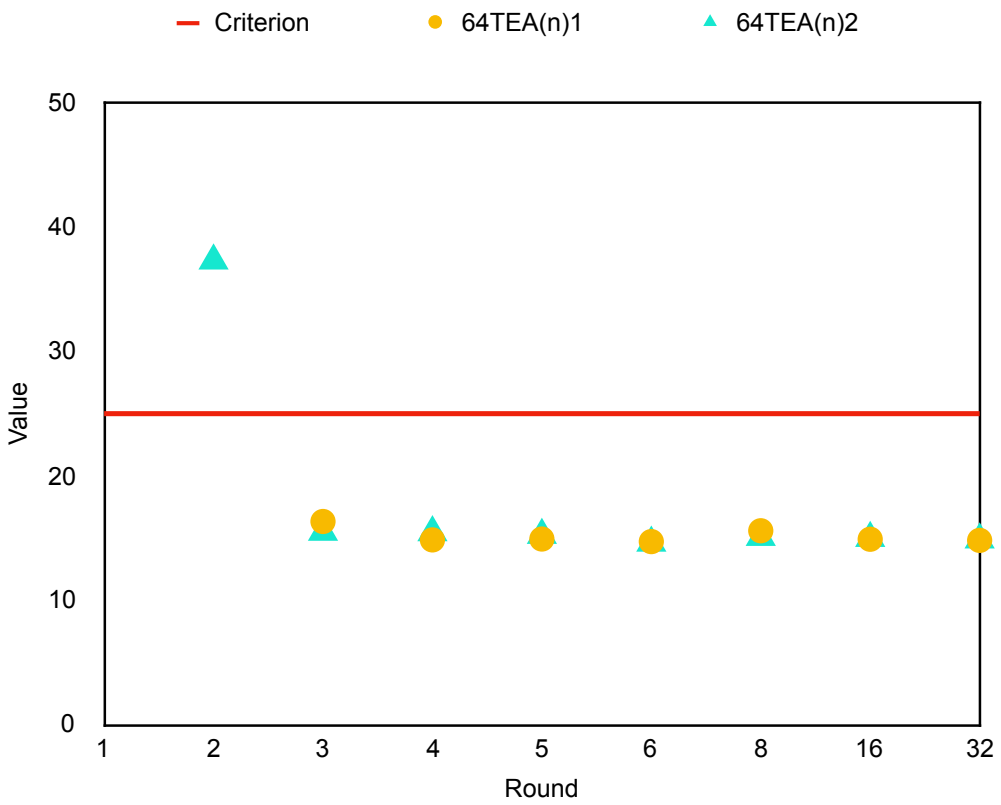


Figure 4.20: 64TEA(n) の等確率性検定

であった．

Figs. 4.23, 4.24 に示すように，等確率性が保証されている一様乱数を用いて生成された正規乱数の平均値，分散は共に，MT 法を用いて生成された正規乱数と良い一致を示すことが確認できた．

Figs. 4.25, 4.26 に示すように，等確率性が保証されている一様乱数を生成する，ラウンド回数が 3 回以上のものにおいて，生成された正規乱数の平均値，分散は共に，MT 法を用いて生成された正規乱数と良い一致を示すことが確認できたが，その性質が保証されなかった．一様乱数を用いて生成した正規乱数は，平均値が 64TEA(1) の時，2.00, 64TEA(2) の時，0.01，分散が，64TEA(1) の時，2.50, 64TEA(2) の時，1.07 と，他のラウンド回数のものに比べ，差異が大きくなったことを確認した．

4.3　解析した物性値の評価

各種法により生成した乱数を用いた時の，DPD 法による水の各物性値についての結果を以下に示す．また，以下で用いる r, t, T はすべて無次元化された変数である．

4.3.1　動径分布関数

Fig. 4.27(a) に，MT 法，32TEA(n), 64TEA(n) を用いて再現した水の系の動径分布関数を示す． また (b) にその誤差率を示す．

Fig. 4.27 に示すように，等確率性，無規則性が保証されている一様乱数を用いて再現された水

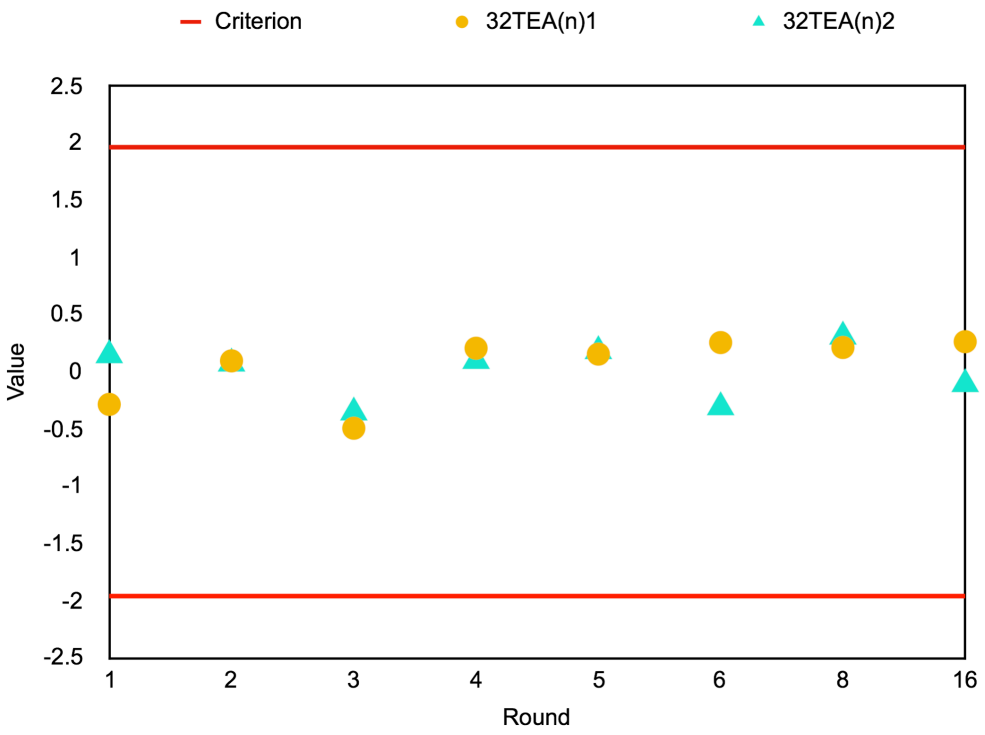


Figure 4.21: 32TEA(n) の無規則性検定

の系の動径分布関数に関しては，MT 法を用いて再現されたものと良い一致を示すことが確認できたが，64TEA(2) では動径分布関数に差異が生じ始め，64TEA(1) では，全く異なるものが得られた．特に，粒子間距離が 0 に近いところにおいて，RDF の値が約 4.0 となり，大きな値を示し，4 つの粒子が集合したものが多数存在していることが確認できた．これは，ランダム項が引力として働いていることを示唆しているが，これは 64TEA(1) に無規則性が保証されていないために，距離が 0 付近の平文の多くが，Fig. 4.18 に示したもののうち，0 以下の値に変換されていると考えられる．

4.3.2 拡散係数

Fig. 4.28(a) に，MT 法，32TEA(n), 64TEA(n) を用いて再現した水の系の平均二乗変位を示す．(b) に拡散係数の誤差率を示す．また，32TEA(n), 64TEA(n) を用いて再現した水の系の拡散係数と MT 法を用いて再現した水の系の拡散係数の比を Figs. 4.29, 4.30 に示す．

Fig. 4.28 に示すように，動径分布関数のものと同様に，64TEA(2)，64TEA(1) に見られるような当確率性及び，無規則性を保証されていない一様乱数を用いて再現された水の系の平均二乗変位は，それらの性質が保証されている一様乱数を用いて再現された水の系のものと比べて，差異が生じていることが確認できた．Fig. 4.30 からわかるように，平均二乗変位より求めた拡散係数の値は，MT 法によって再現した水の系のものと比較して，64TEA(2) の場合約 1.15 倍に，64TEA(1) の場合約 2.20 倍になった．

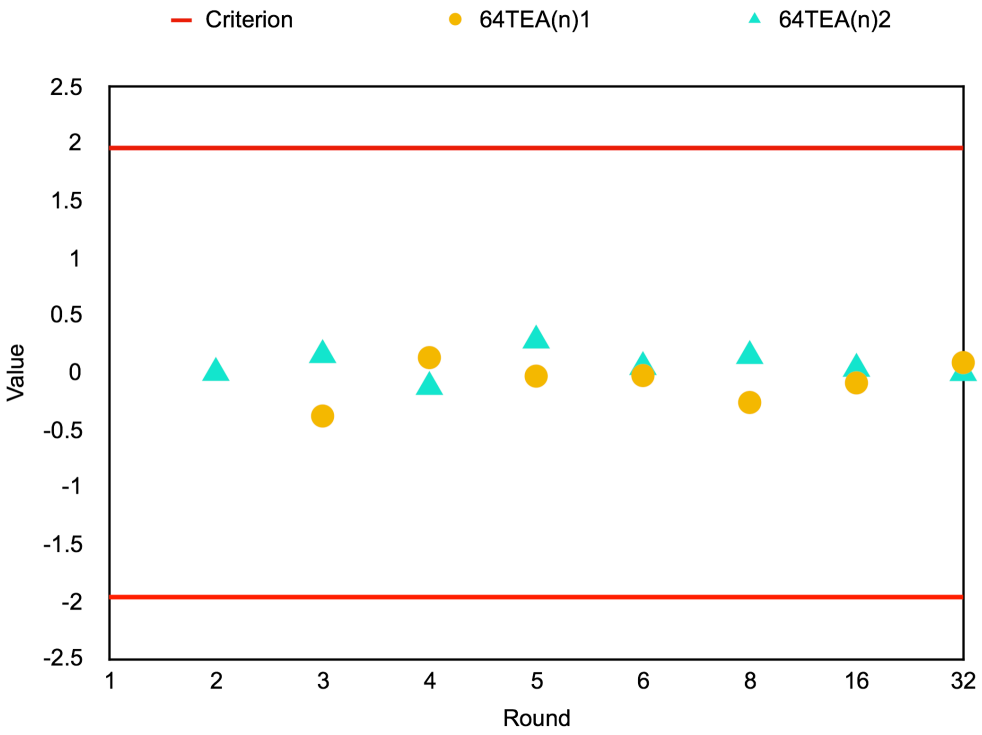


Figure 4.22: 64TEA(n) の無規則性検定

4.3.3　温度

Fig. 4.31(a) に，MT 法，32TEA(n), 64TEA(n) を用いて再現した水の系の温度を示す．(b) にその平均温度，分散を示す．また 32TEA(n), 64TEA(n) を用いて再現した水の系の温度平均及び，分散と MT 法を用いて再現した水の系の温度平均，分散の比を Figs. 4.32, 4.33, 4.34, 4.35 に示す．

Fig. 4.31 に示すように，64TEA(2)，64TEA(1) に見られるような当確率性及び，無規則性を保証されていない一様乱数を用いて再現された水の系の温度は，それらの性質が保証されている一様乱数を用いて再現された水の系のものと比べて，差異が生じていることが確認できた．また，64TEA(1) に関して，温度の揺らぎが大きいことがわかる．

Fig. 4.33 からわかるように，温度の平均値は，MT 法によって再現した水の系のものと比較して，64TEA(2) の場合約 1.15 倍に，64TEA(1) の場合約 2.64 倍になった．また，Fig. 4.35 からわかるように，温度の分散は，MT 法によって再現した水の系のものと比較して，64TEA(2) の場合約 1.09 倍に，64TEA(1) の場合約 6.33 倍になった．

Fig. 4.33 に，32TEA(n), 64TEA(n) からの正規分布に従う乱数を用いた場合に，DPD 法で再現した水分子系の (a)32TEA(1) の各方向の速度分布，(c) 理論値からの誤差率，(b)64TEA(2) の各方向の速度分布，(d) 理論値からの誤差率を示す．この図から，乱数が一様性と無相関性を示す場合は，誤差率が 0.02 であるのに対して，64TEA(2) に用に性質を満たさない乱数を用いた場合にはその誤差率が大きくなる上に，各方向の速度分布に差が生じる．

Fig. 4.37 に 32TEA(n), 64TEA(n) を用いて，乱数を 1.0×10^11 個生成した場合の計算コストを示す．計算コストが最も小さく，かつ，MT 法を用いて再現する水の系と同様のものを再現す

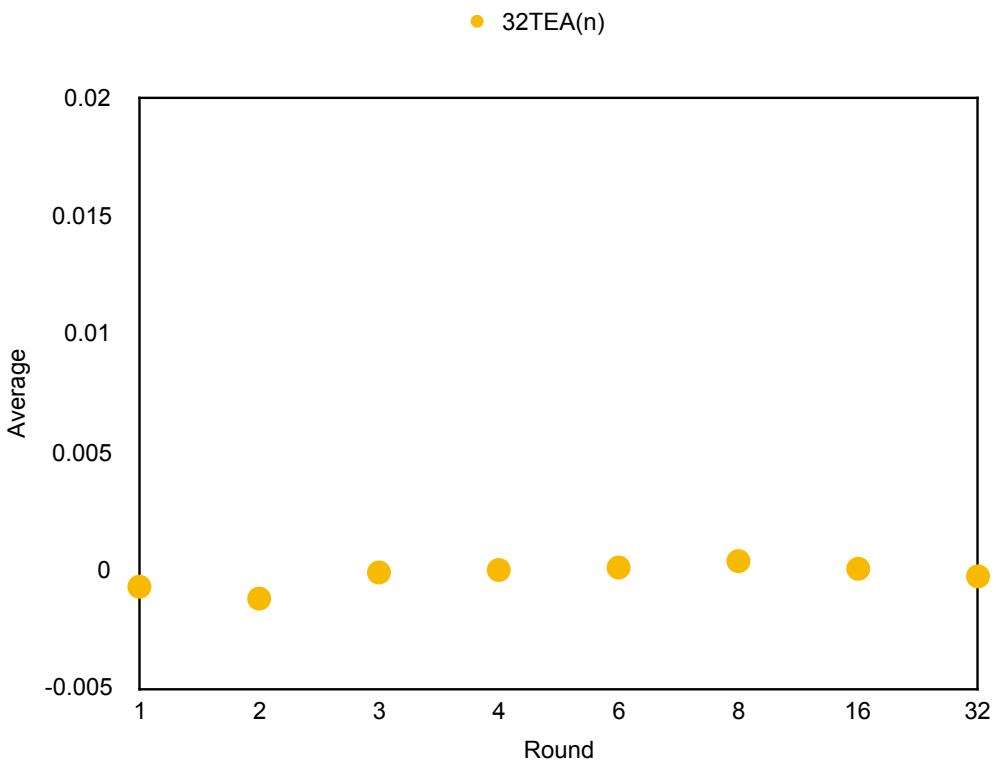


Figure 4.23: 32TEA(n) による正規分布の平均値

Table 4.2: 総計算時間

	MT	32tea(8)	32tea(1)	64tea(3)
総計算時間	6.839	10.733	6.100	6.232

るためには 32TEA(n) の場合ラウンド回数が 1 回，64TEA(n) の場合はラウンド回数が 3 回であることが示された．MT 法，32TEA(1)，64TEA(3) それぞれを用いた場合に，本研究の水分子系 256 個を 1000 step を再現するための総計算時間を Table 4.2 に，一様乱数生成に要した計算コストの全体に対する割合を Fig. 4.38 に示す．なお，DPD 法では保存力，散逸力の計算コストが小さいために，カットオフ半径を判定するのに，粒子間距離の計算に要するコストが比較的大きい．そのため，力の計算コストに要する時間は，カットオフを判定するために用いる効率的なアルゴリズムに依存する．そこで，そのような影響を無視するために，カットオフ判定による効率化を無視し，粒子間距離を計算するすべての粒子ペアに対して乱数を生成し，計算コストを比較した．計算本研究で作成したコードはカットオフ外に存在する粒子に対しても乱数を生成している．

Table 4.2 および Fig 4.38 より，32TEA(1) や，64TEA(3) は提案されている 32TEA(8) に比べ計算コストが 1/8 になり，MT 法を用いての乱数生成に要する計算コストとほぼ同等の計算コストで水の系を再現することができた．なお，32TEA(8) は全体の 41.57%, 32TEA(1) は全体の 8.01% の計算コストを占めるため，従来の約 1.57 倍の計算速度で水の系を再現することができた．

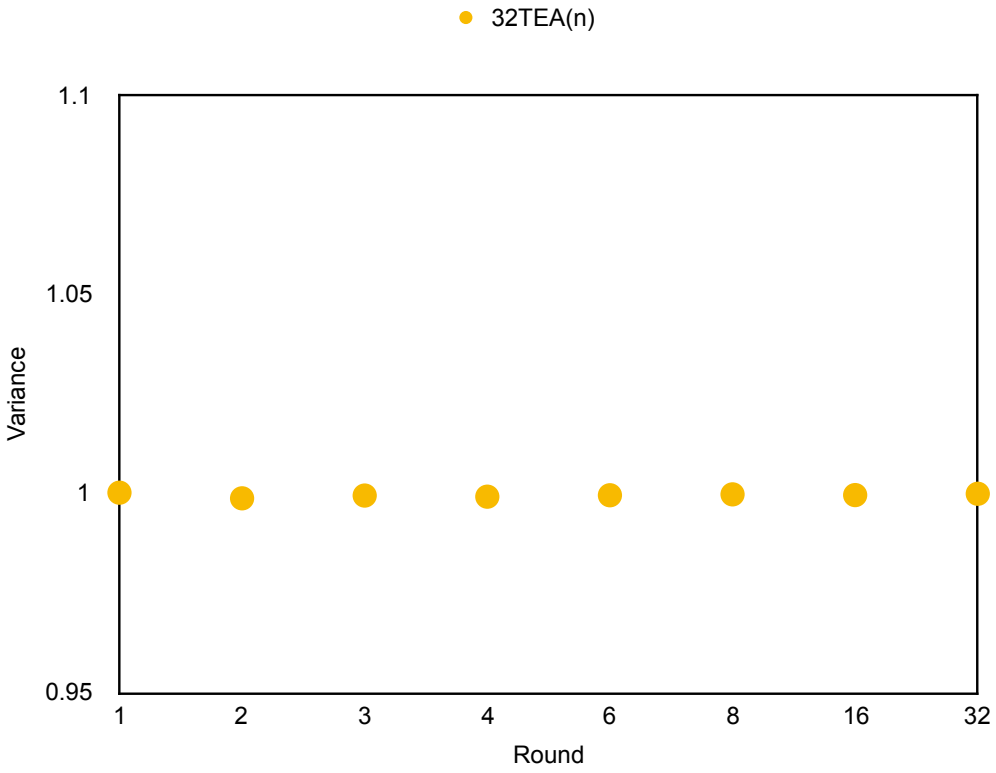


Figure 4.24: 32TEA(n) による正規分布の分散

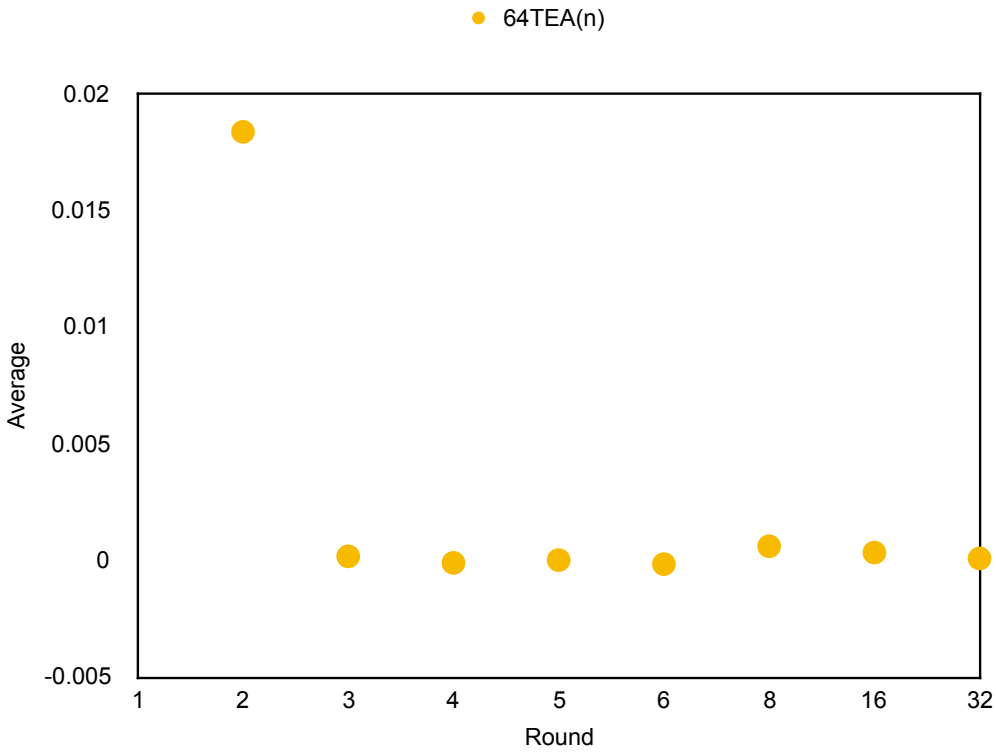


Figure 4.25: 64TEA(n) による正規分布の平均値

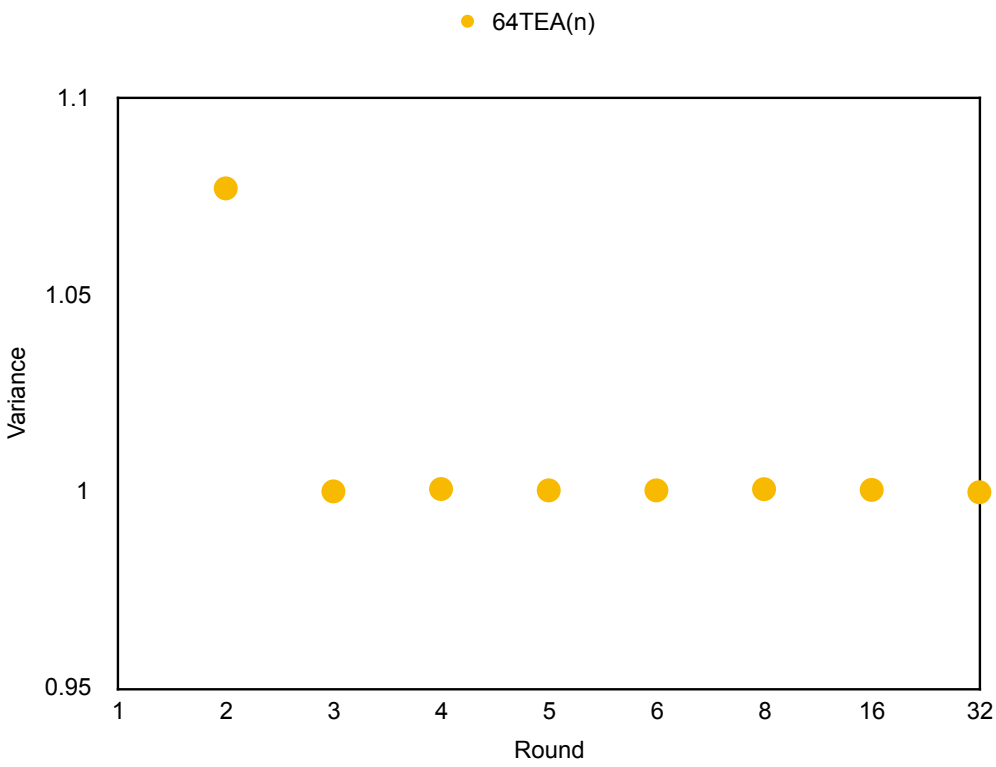


Figure 4.26: 64TEA(n) による正規分布の分散

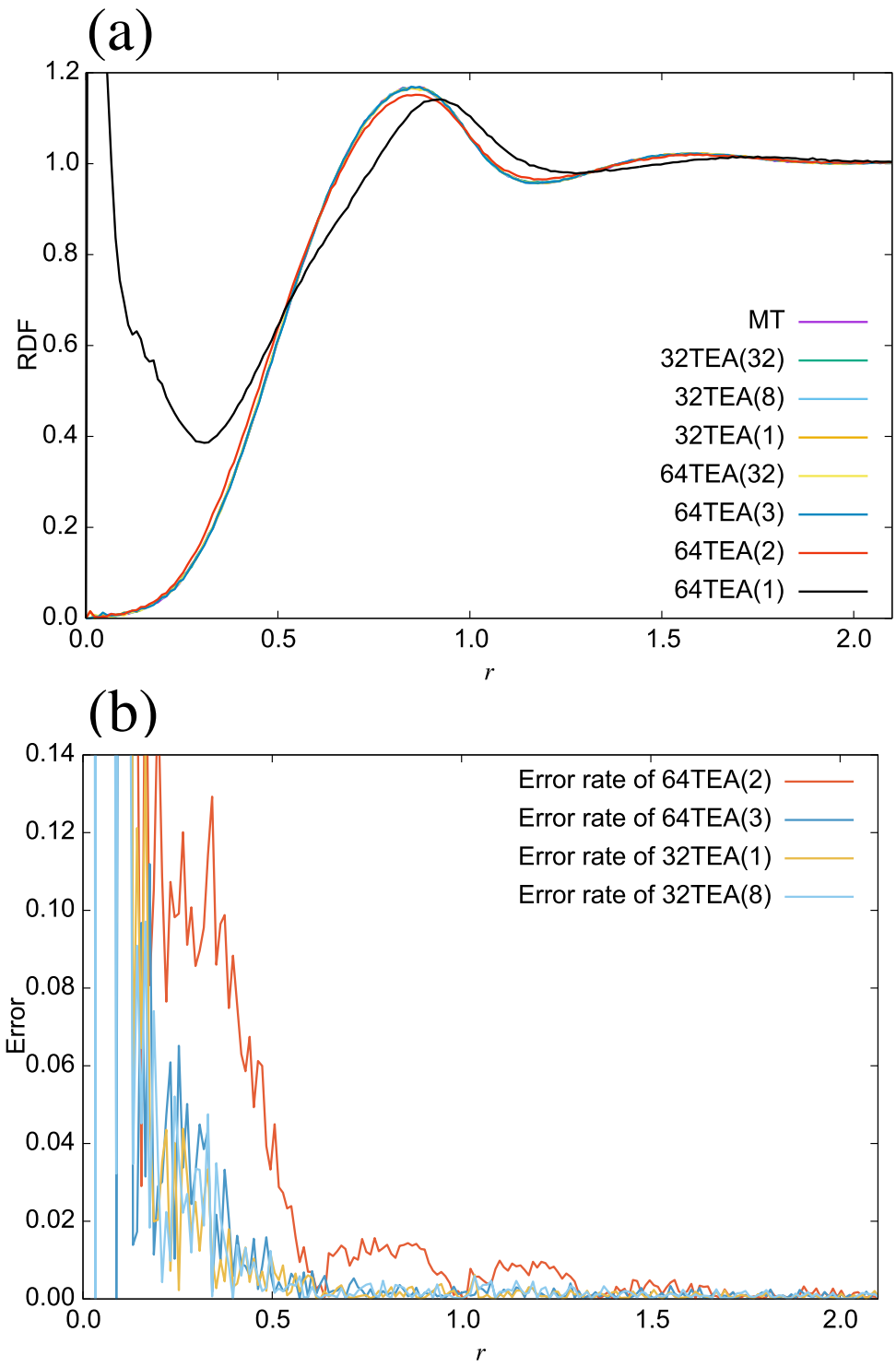


Figure 4.27: 32TEA(n), 64TEA(n) からの正規分布に従う乱数を用いた場合に，DPD 法で再現した水分子系の (a) 動径分布関数と (b) その誤差率

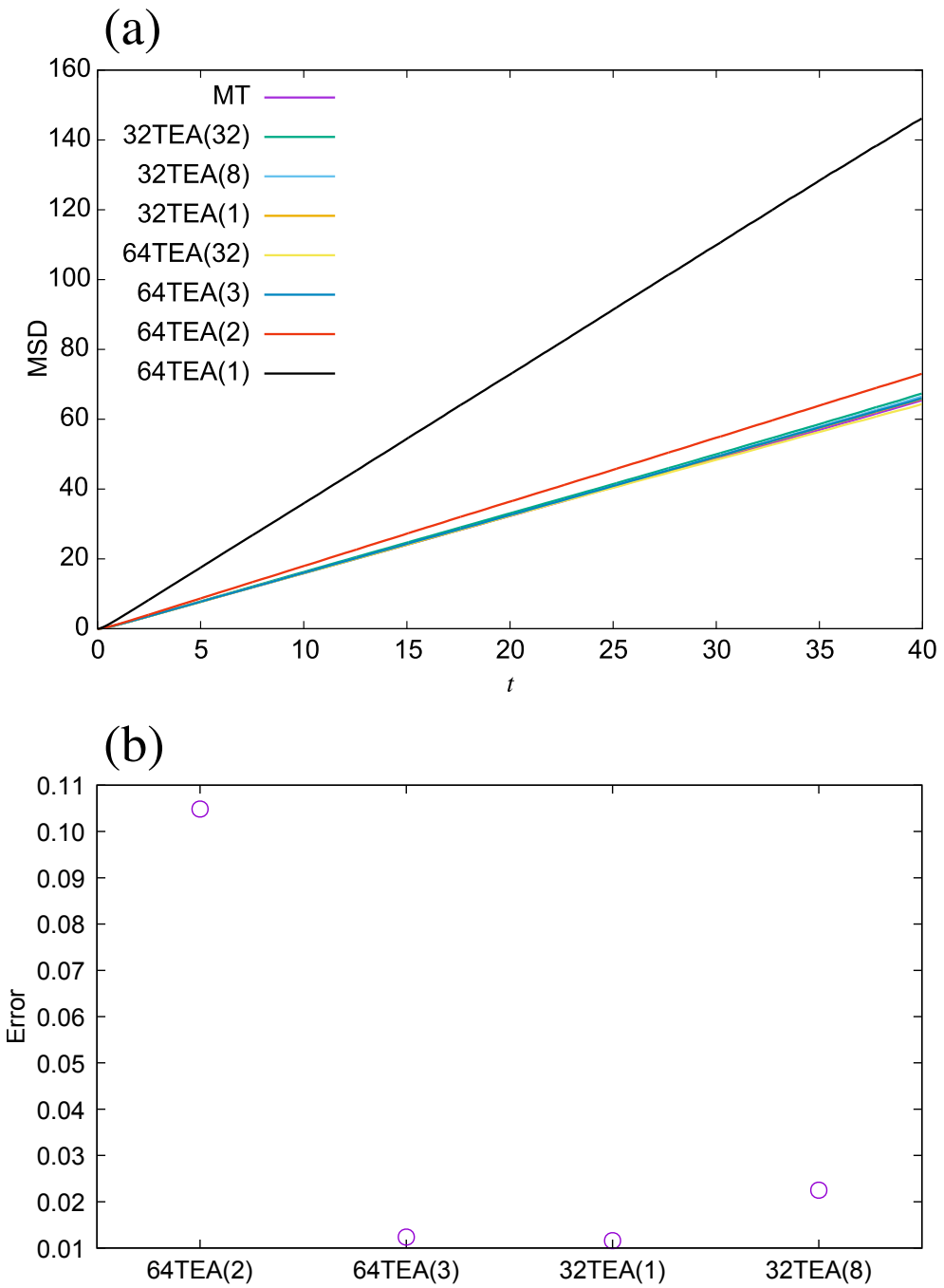
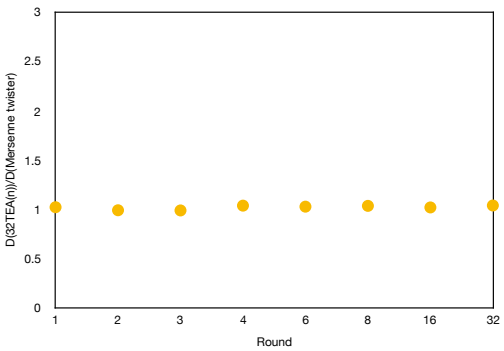


Figure 4.28: 32TEA(n), 64TEA(n) からの正規分布に従う乱数を用いた場合に，DPD 法で再現した水分子系の (a) 平均二乗変位と (b) 拡散係数の誤差率

(a)



(b)

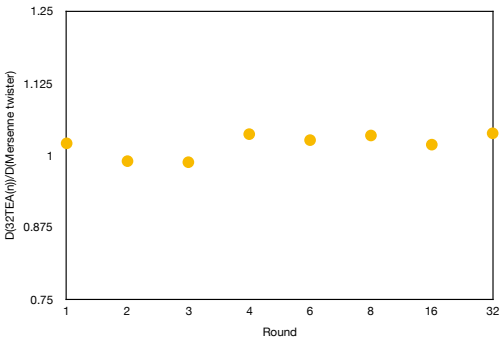
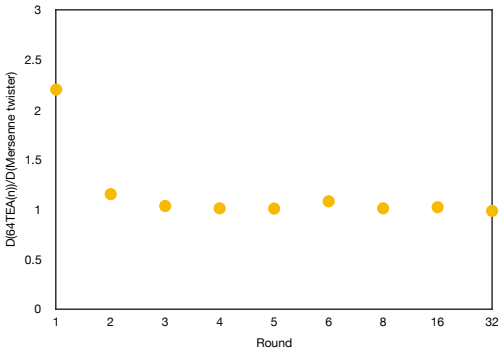


Figure 4.29: 32TEA(n) による拡散係数の比 (a) 区間 [0,3], (b) 区間 [0.75,1.25]

(a)



(b)

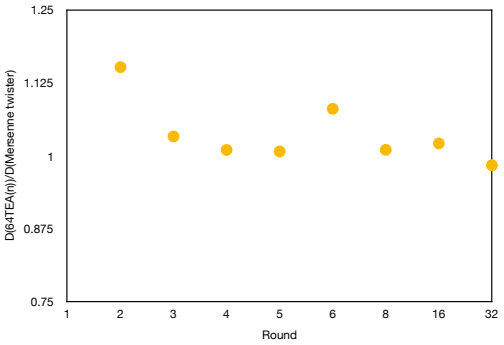


Figure 4.30: 64TEA(n) による拡散係数の比 (a) 区間 [0,3], (b) 区間 [0.75,1.25]

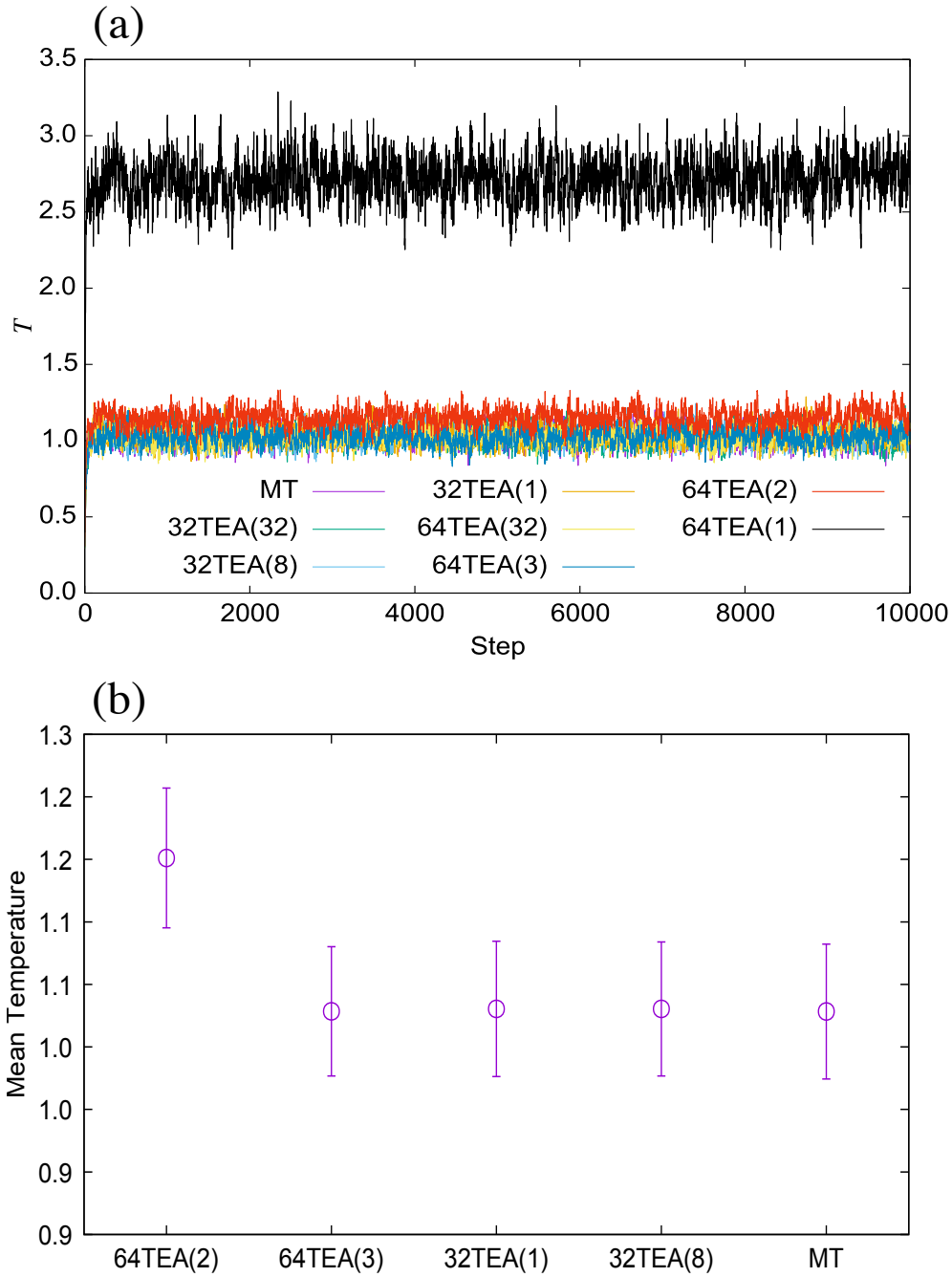
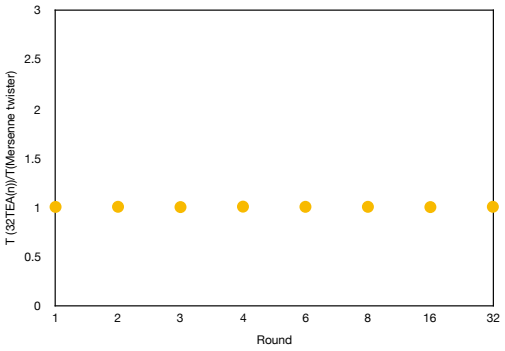


Figure 4.31: 32TEA(n), 64TEA(n) からの一様分布に従う乱数を用いた場合に，DPD 法で再現した水分子系の (a) 温度と (b) 温度平均及び分散

(a)



(b)

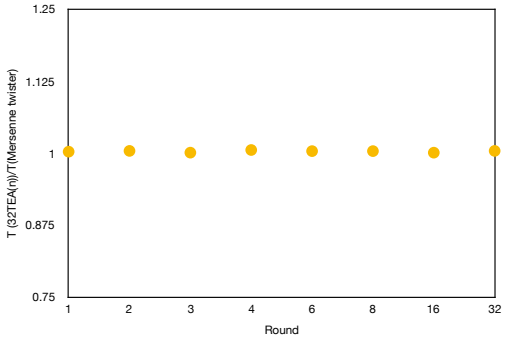
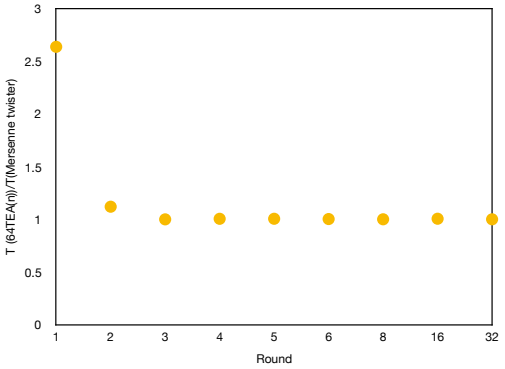


Figure 4.32: 32TEA(n) による温度の比 (a) 区間 [0,3], (b) 区間 [0.75,1.25]

(a)



(b)

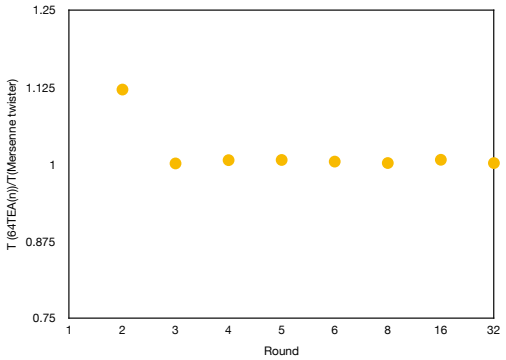
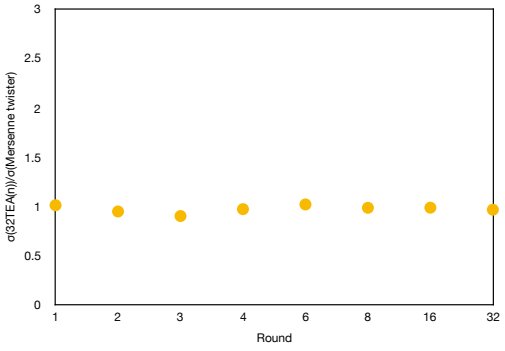


Figure 4.33: 64TEA(n) による温度の比 (a) 区間 [0,3], (b) 区間 [0.75,1.25]

(a)



(b)

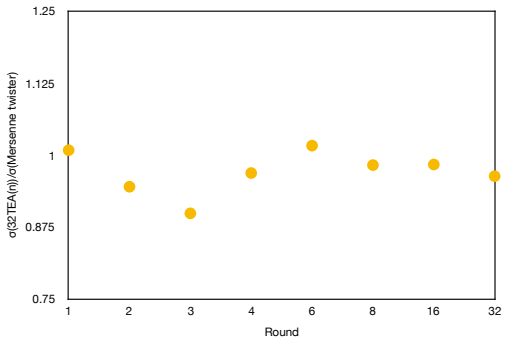


Figure 4.34: 32TEA(n) による温度ゆらぎの比 (a) 区間 [0,3], (b) 区間 [0.75,1.25]

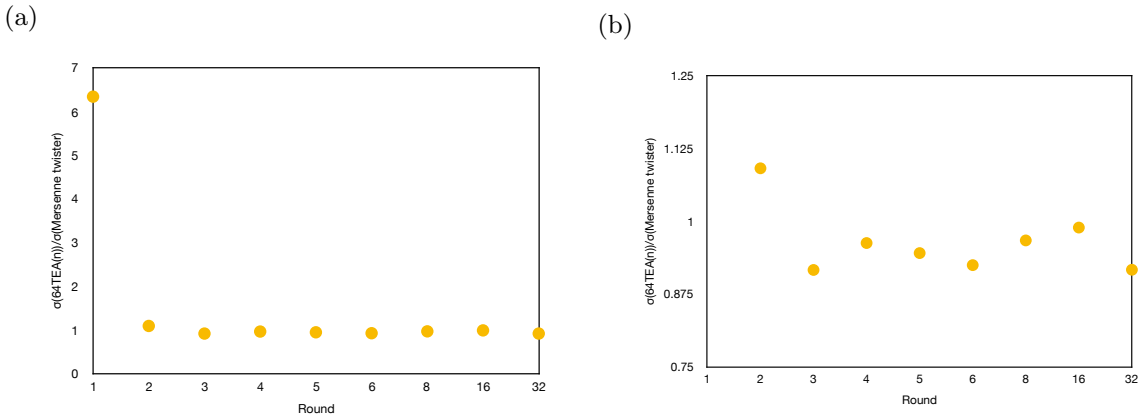


Figure 4.35: 64TEA(n) による温度分散の比 (a) 区間 [0,7], (b) 区間 [0.75,1.25]

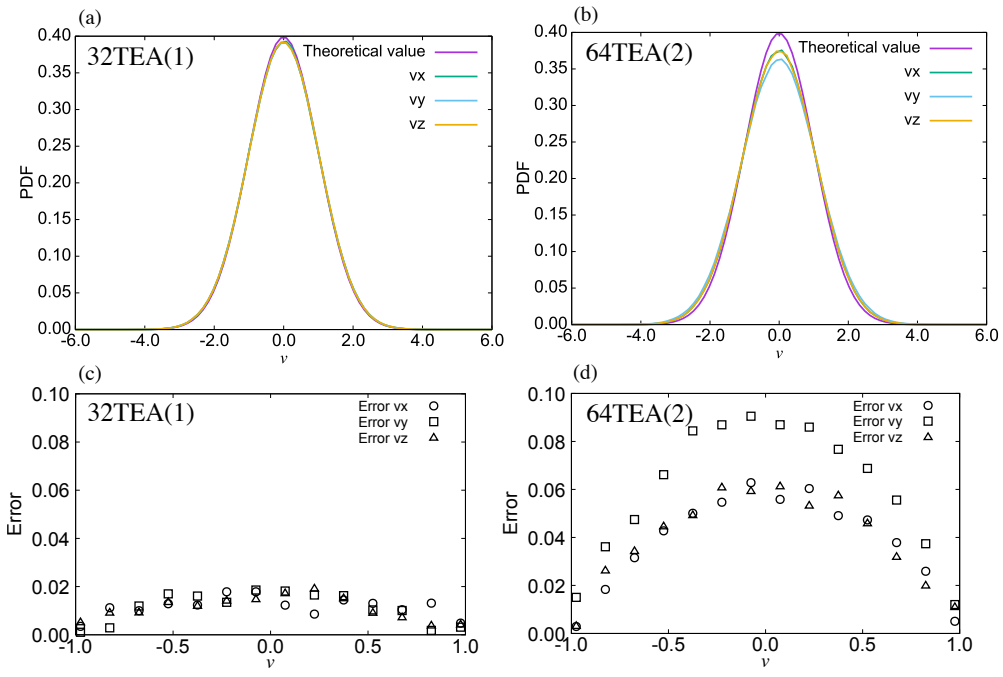


Figure 4.36: 32TEA(n), 64TEA(n) からの正規分布に従う乱数を用いた場合に，DPD 法で再現した水分子系の (a)32TEA(1) の各方向の速度分布，(c) 理論値からの誤差率，(b)64TEA(2) の各方向の速度分布，(d) 理論値からの誤差率

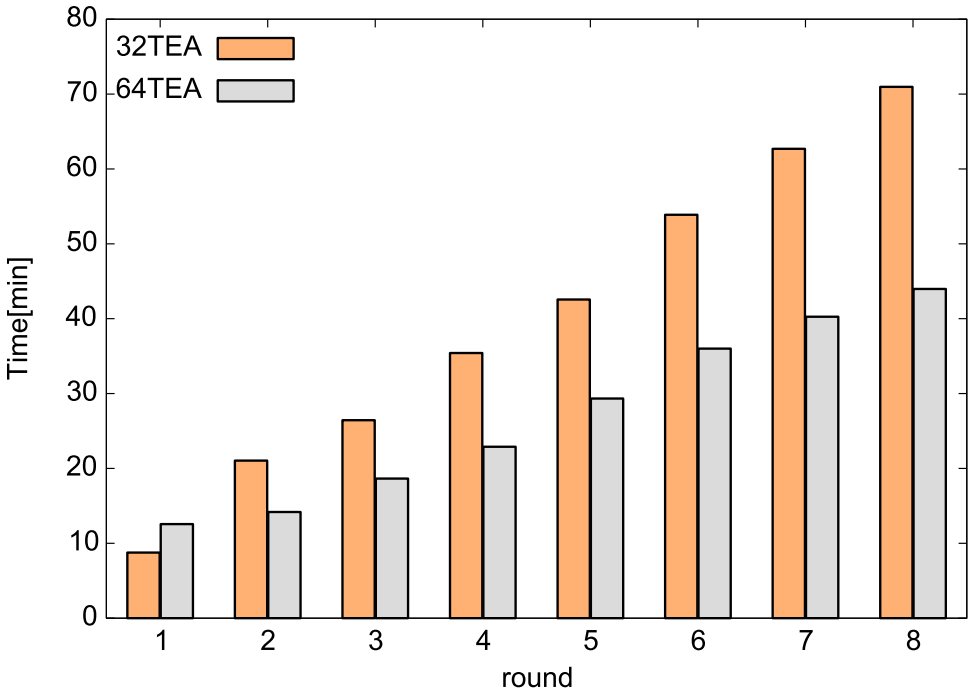


Figure 4.37: 32TEA(n), 64TEA(n) を用い，乱数を 1.0×10^11 個生成するのに要する時間

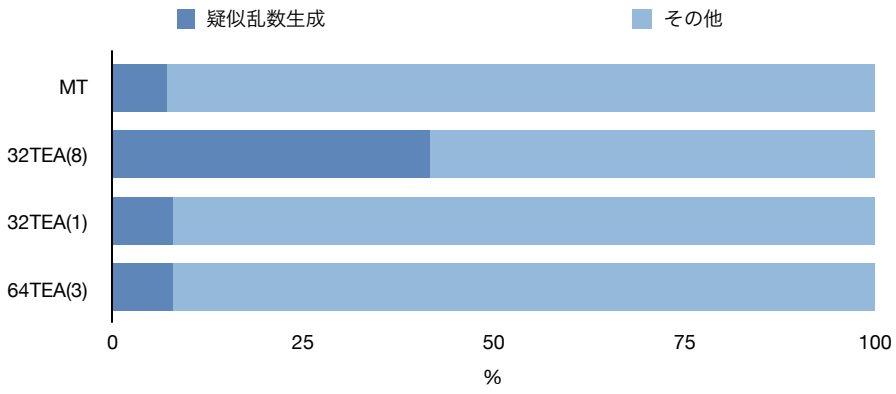


Figure 4.38: 乱数生成の計算コスト比

第 5 章

散逸粒子動力学法のための新規乱数生成手法の開発

実験 1 で行った解析結果をもとに，更に効率的な乱数生成器を設計，検討する．ここでは，主に乱数シード値に現れる乱雑性の評価と，その乱雑性をどの様に用いて乱数を生成するのかを検討する．また実験 1 と同様，DPD 法に適用する実験も行う．（先頭の文章と同じ）

実験 1 では，的確に乱数シードを選択することで，TEA のラウンド回数が 1 回においても DPD 法によるシミュレーションが精度良く実行できることがわかった．ここで，なぜ，TEA の軽量化にもかかわらず，性質の良い乱数が生成できたのかをより詳細に考察する．TEA は前章でも説明したとおり，暗号化のためには 32 ラウンド回数が提案されており，bit 列に Avalanche 効果が現れるのにも，少なくとも 6 ラウンド回数の処理を行う必要がある．その 1/6 の処理である 1 ラウンドの処理は Fig. 3.16 で示したとおりであり，連続した bit 列の相関を打ち消すのに十分ではない．このことから，乱数シードが既に相関が打ち消されたものであった可能性が示唆される．そこで，以降では乱数シードに用いた粒子座標の乱雑性について詳しく調査する．

5.1　粒子絶対座標の乱雑性の調査

ここでは，暗号化において乱数シードとして用いた，粒子座標の乱雑性について調査する．

Fig. 5.1 に示すように，すべての粒子座標は 64bit の double 型，倍精度浮動小数点表現として記録されている．これらの bit 列の内，各部分の乱雑性を調査するために，下 bit 列を仮数部 (mantissa) から 1-32bit を抽出した．そして，この bit 列を連続する時間ステップ，特定の粒子のみから得られる 32bit を約 1Gbit(1048576000bit) を集め，NIST 乱数検定による検定を行った．さらに同様の処理を bit 列の内，下 bit 列仮数部 (mantissa)2-33bit，3-34bit，4-35bit，...，25-57bit に対しても行った．

検定結果を Fig. 5.2 に示す．この図から，15-46bit から 26-57bit にかけて，NIST 乱数検定による評価が下がっていくことがわかる．ここから，bit 内にある規則性，また，連続する時間ステップにおける相関の影響が出てきていることが見て取れる．粒子座標を表す double 型表現の内，数値の符号を表現する符号部 (sign)，数値の指数を表現する指数部 (exponent) は連続する時間ステップの中で大きく変化しない．また同様に，仮数部の前半部も粒子の動きが微小であればあるほど，変化が小さくなる．そのため，この bit 部分を乱数として利用することはできない．

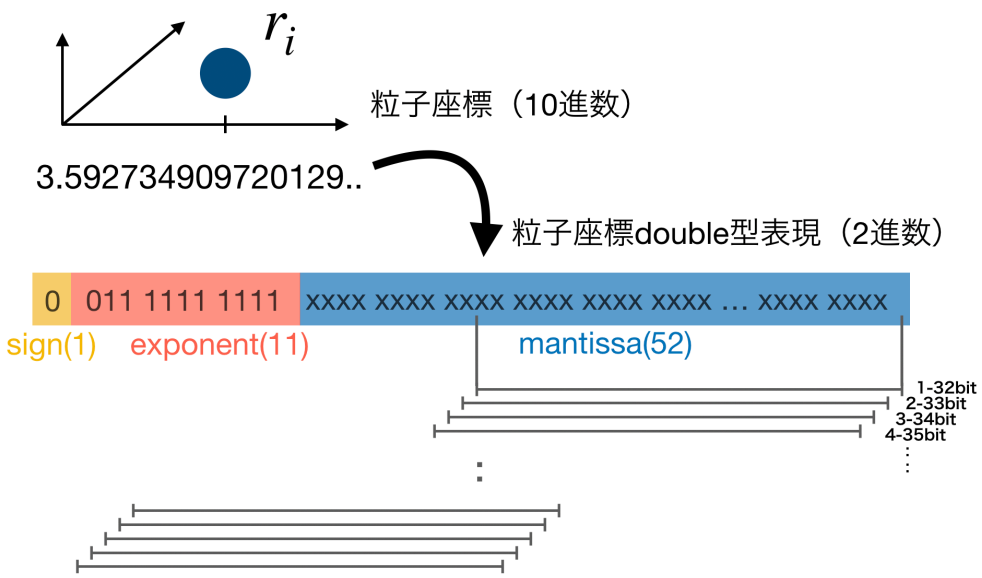


Figure 5.1: 粒子座標を表現する 64bit の内，NIST 乱数検定に用いたかく bit 列抽出法の概略図

一方で，7-38bit から 14-45bit の範囲で NIST 乱数検定による高い評価を得ており，性質の良い乱数性を有していることを示している．これは，連続する時間ステップにおける粒子座標の相関が現れない部分があることがわかる．以降ではこの部分を用いた乱数生成手法を考える．

また，1-32bit 付近では NIST 乱数検定による高得点を得ることはなかった．また 7-38bit の部分にかけて NIST 乱数検定の得点が上昇していることがわかる．Fig. 5.3 に示すように，計算に伴う数値の繰り上がりが影響していると考えられる．64bit 全体は 0 または 1 の数値を取りうるが，数値繰り上がりの際 64bit 以降は 0 が存在していると仮定し，実行される．そのため，常に 0 だけによる影響が下 bit 部に現れ，乱雑性の低下につながったと考えられる．

Fig. 5.4(a) に示すように現段階で，各計算ステップにおいて，粒子数 N 個の乱数資源を準備できた．しかし，DPD 法では，Fig. 5.4(b) に示すように各粒子ペア N^2 個の乱数を生成する必要がある．そこで以降ではどの様に各ステップで N^2 個の乱数資源を生成するかを説明する．

Fig. 5.5(a) に示すように，乱雑性を含む対応した粒子座標の組を計算上高速な演算によって処理し，その後生成される bit 列の乱雑性をそれぞれの bit 毎に NIST 乱数検定により評価した．演算に和算を用いた場合，生成された bit 列は 1-32bit から 15-46bit にかけて約 11 点の評価を得た．（Fig. 5.6(a)）一方で，演算に乗算を用いた場合，同様の bit 列範囲において高い評価を得た．（Fig. 5.6(b)）この結果より，乗算は和算に比べ，より広い bit 範囲での繰り上がりに影響しており，乱雑性の生成に適していることを示している．したがって，以降ではこの乱数資源を用いて乱数生成手法を設計する．また，Fig. 5.5(b) に示すように，演算子による処理を乱数資源を抽出した後に行う実験も試みた．しかし，NIST 乱数検定の結果，どの演算子を用いた場合においても，生成された bit 列は性質の良い乱雑性を有していないことがわかった．

以上の実験結果より，粒子座標の組に対して乗算処理を行ったものを乱数シードとして，乱数を生成する．Fig. 5.7 に乱数生成器の概要図，Algorithm. 5.6 に乱数生成手法の疑似コード（これは後で作成）を示す．

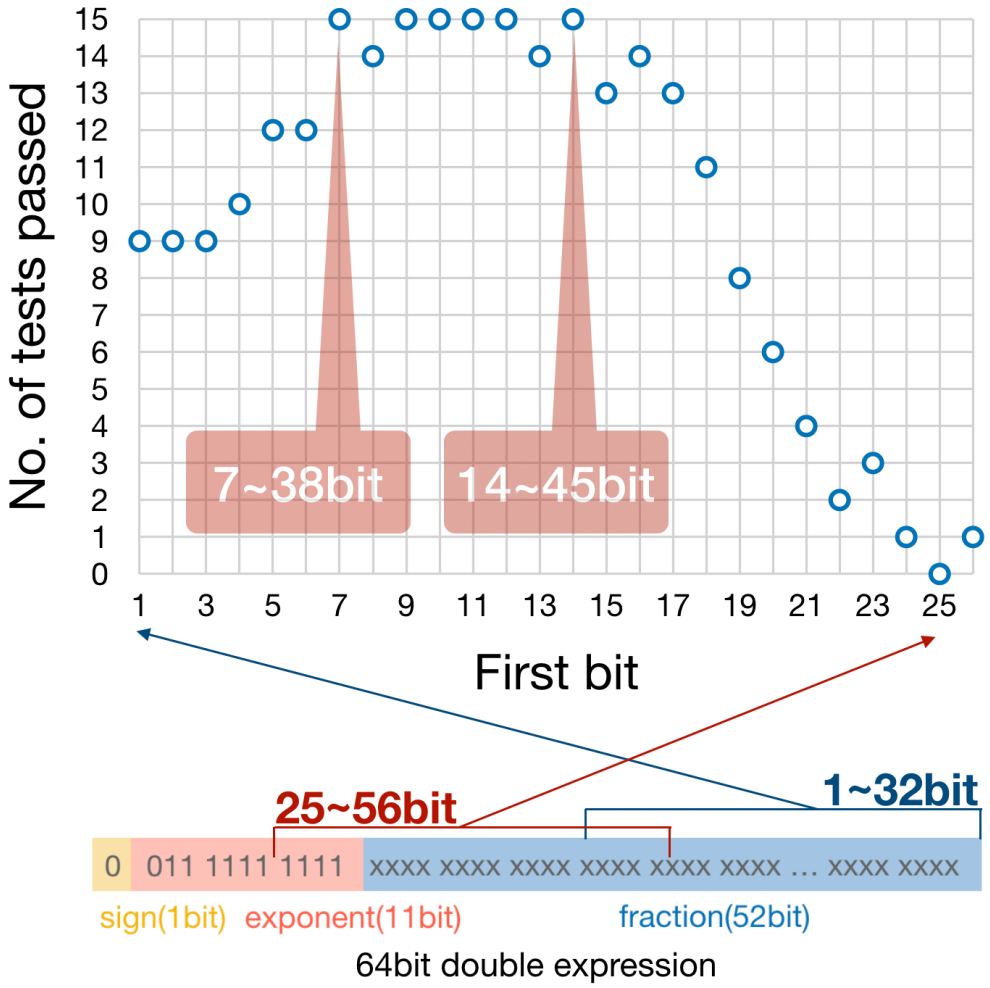


Figure 5.2: 粒子座標 64bit の各部分に対する NIST 乱数検定を用いた検証結果

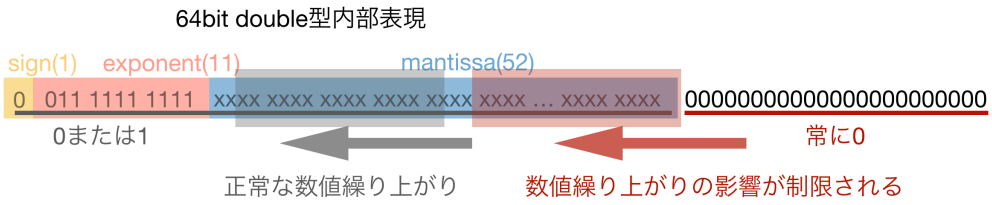


Figure 5.3: 粒子座標の double 型内部表現における繰り上がりの影響

乱数生成は以下の 6 つのステップによって行われる．

(1) 粒子座標の組に対して乗算処理を行い，乱数シードを生成する．

(2) 生成した乱数シードを共用体を用いて，整数型として扱う．

(3) 整数型として扱う bit 列に対して，32bit の bit shift 処理を行う．

(4) 処理後の bit 列を整数型 unsigned long long 型として読み込む

(5) unsigned long long 型の整数値を double 型にキャストする．

(6) $[0, 1]$ の範囲の数値に変換するため，double 型の数値に対して，unsigned long long 型の最大

値で除す．

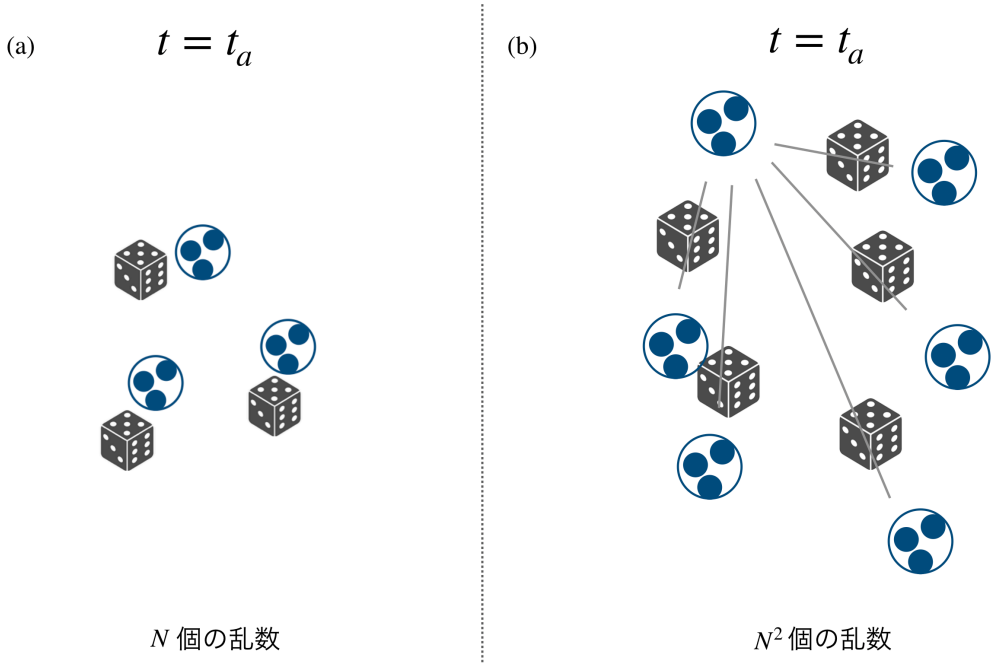


Figure 5.4: 粒子座標から得られる N 個の乱数資源と，力の計算に必要な N^2 個の乱数資源の概略図

Algorithm 8 提案する新規乱数生成手法のアルゴリズム

```
union Uni
  double HIRA
  unsigned long long ANGO
end union
uni.HIRA = position[i] × position[j]
uni.ANGO = uni.ANGO << 32
rand = (double)uni.ANGO
rand = rand / ULLONG_MAX
```

以上の方法で生成した乱数は 32bit の精度である．Fig. 5.8(a) に示す用に double 型では仮数部に 52bit が割り当てられているので，double 型にキャストした際，20bit 分の表現枠の余りが生じる．また，Fig. 5.8(b) に示すように，64bit 精度の乱数を生成した場合，double 型にキャストした際に，12bit 分の情報が失われることがわかる．上記の事項を考慮した上で，Fig. 5.8(a) に示す 20bit の余りがどの様に保管されるのかを考える．ランダム力で用いた粒子座標は保存力の計算でも用いられており，さらに，散逸力は相対速度に基づいて計算される．これらの演算にも同様の乱雑性が含まれており，3 項の和算によって，20bit が補間される．

5.2　新規乱数生成手法の開発

実験 2 で設計した乱数生成手法に対して，より厳しい条件のもとでの検証を行う．どのような場合にこの手法は適用できなくなるのか，また一般的に DPD 法で再現されるどのような系に対しても利用可能であるかどうかを検証する．

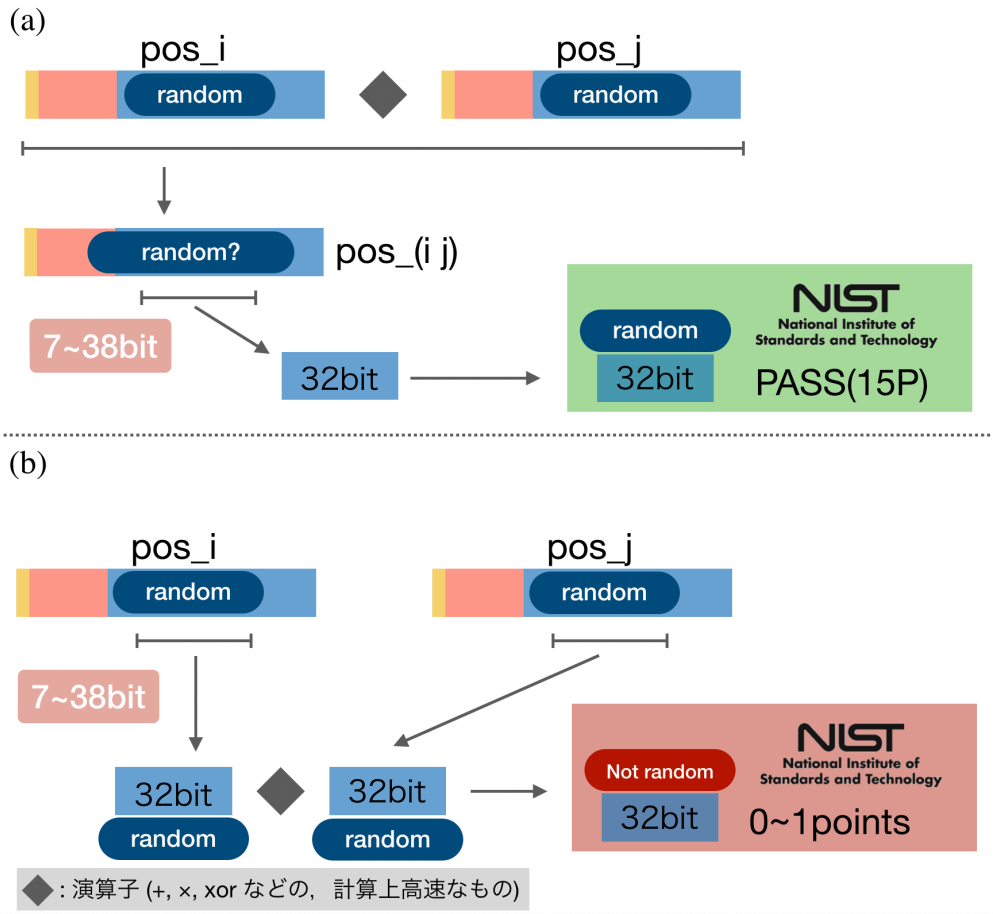


Figure 5.5: (a) ペアの粒子座標に対して，演算後から得られる乱数資源を用いた場合と，(b) 演算前に乱数資源を抽出し，その後演算した乱数資源を用いた場合の概略図

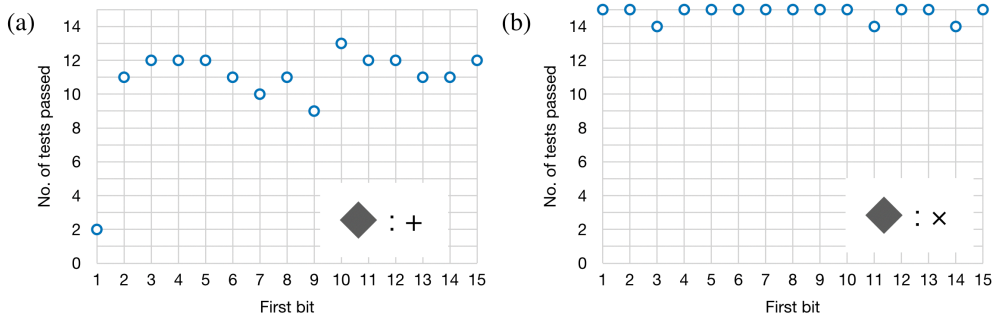


Figure 5.6: 演算後から得られる乱数資源を用いた場合について，(a) 和算を用いた場合と，(b) 乗算を用いた場合の NIST 乱数検定の結果

DPD 方では様々な物質の解析が行われいる．中には複数の粒子と結合した重合体など，連続する時間ステップの中で座標が大きく変わらない物質も存在する．そのような系のシミュレーションに対しても，本手法は問題なく使用できるのかを検証する．

時間方向に強い相関を持つ運動を疑似的に再現するために，本実験では，一般的に用いられている計算ステップ幅をより小さくして得られる乱数を検証した．Fig. 5.10 に示すように，時間ステップ幅を小さく設定することで，位置座標の更新に伴う変化が小さくなり，結果として粒子座標に含まれる乱雑性は少なくなる．そこで，時間ステップをどこまで小さくすると，生成

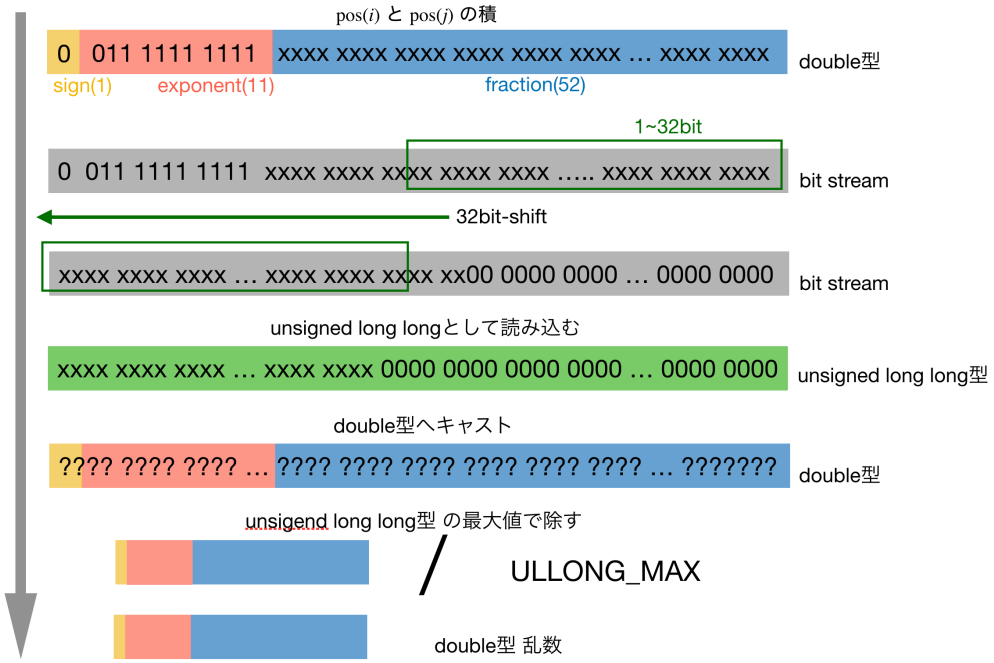


Figure 5.7: 提案する新規乱数生成手法の概略図

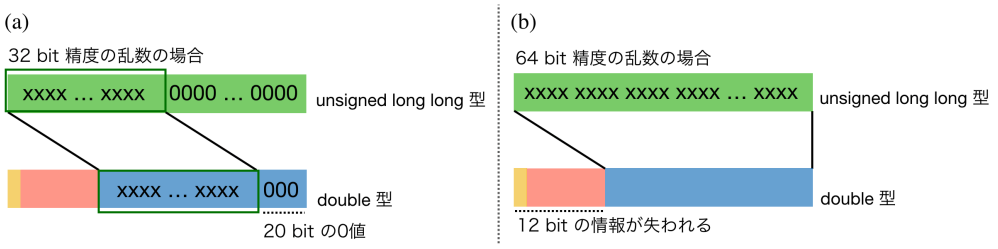


Figure 5.8: (a)32bit 精度の整数乱数を double 型にキャストした場合に残る相関のある bit 数についてと，(b)64bit 精度の整数乱数を double 型にキャストした場合に失われる bit 数についての概略図

する乱数に影響が現れるのかを調べた．実験では，一般的に用いられる計算ステップ幅 0.04 と $1/10, 1/100, 1/1000, ..., 1/10^{14}$ のそれぞれの時間ステップ幅で同様の水の系を計算し，得られる乱数の乱雑性を NIST 乱数検定によって評価した．この実験で用いた bit 列は，ある特定の組を選び，粒子座標の乗算処理によって生成される bit 列の後半 32bit を，連続する時間ステップの中から集めたものを利用した．Fig. 5.11 にその結果を示す．ここから，従来の計算ステップ幅から 1/1000 のもので行った場合においても，各時間において良質な乱数を生成することが確認できた．この結果はつまり，通常の時間ステップにおいて，水分子の 1/1000 のスケールで運動する物質を取り扱っても，問題なく乱数が生成されることを示している．

Fig. 5.12 には，従来の方法 (GPUTEA の論文を引用) と同様，特定粒子の組を選ばず，系全体から生成される乱数を集め，NIST 乱数検定を行った結果を示す．この実験の場合，検定結果から，時間ステップを従来のものより $1/10^7$ のもので行った場合でも問題なく乱数を生成できることを示している．しかし，最も相関が大きく現れるものを抽出するためには，特定の粒子ペアを選択し，検定する必要があると考えられる．

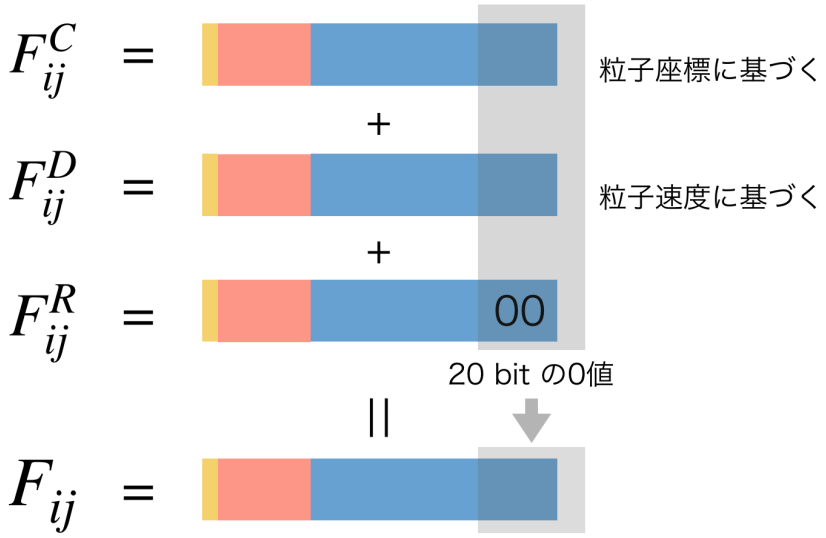


Figure 5.9: 相関のある bit 列が保存力，散逸力の和算によって保管される流れの概略図

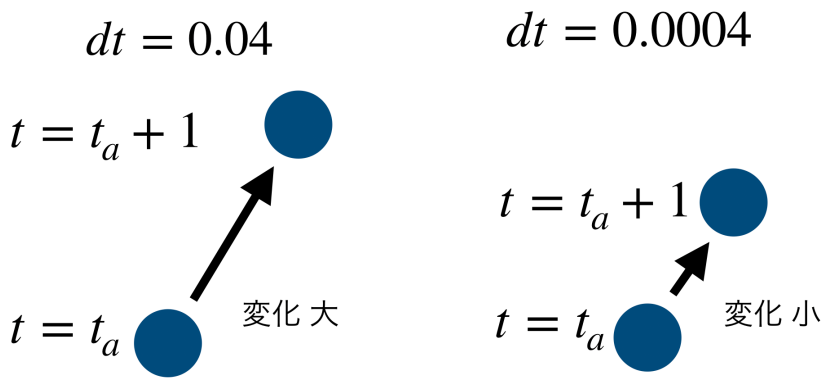


Figure 5.10: 時間ステップ幅が小さくなる場合，粒子座標の変化が小さくなることを示す概略図

5.3　新規乱数生成手法の適用例とその評価

ここまでで紹介した新規乱数生成手法の適用例として，先行研究と同様の系を再現し，問題なくシミュレーションが実行できるかどうかを確認する．

先行研究 [91] に倣い，Fig. 5.16 に示す，脂質分子間の反発パラメータを Table 5.1 のように定めた．

先行研究で報告されている通り，POPC の体積分率を変化させることで，Fig. 5.13 に示すように，それぞれ POPC9.7% においてベシクルの形成，POPC17% において脂質二重層の形成を確認した．

また，Fig. 5.13 に示すように，膜圧力を計測し，をの異方性を異なる POPC の体積分率ごとに求めた．この計測値は従来の手法を用いた場合と同等の精度つまり，従来の手法との差が分散内（0.005%）で得ることができた．なお，2000 万ステップの平均値をプロットした．

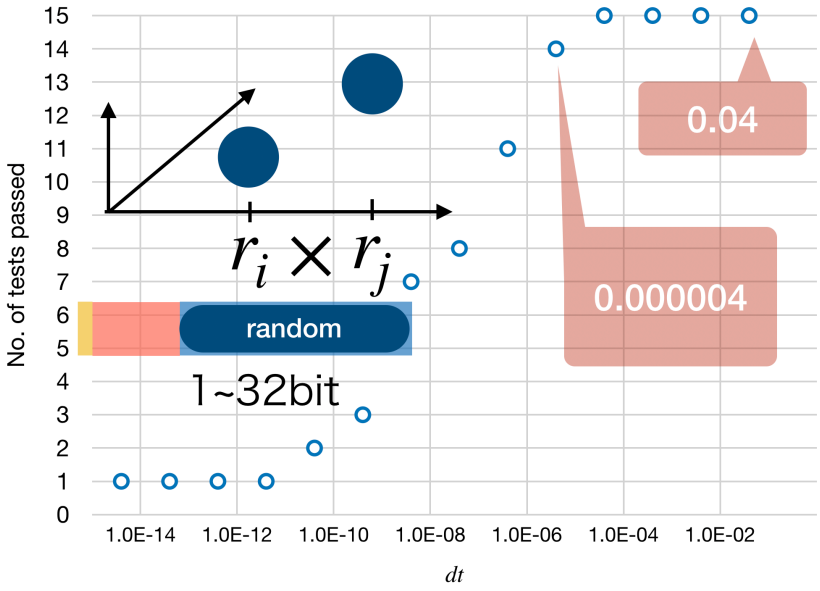


Figure 5.11: 異なる時間ステップ幅を用いた場合に，得られる乱数資源の乱雑性に対する NIST 乱数検定の結果（入力した乱数として，連続するステップの中である特定の粒子ペアに生成される乱数のみを用いた場合）

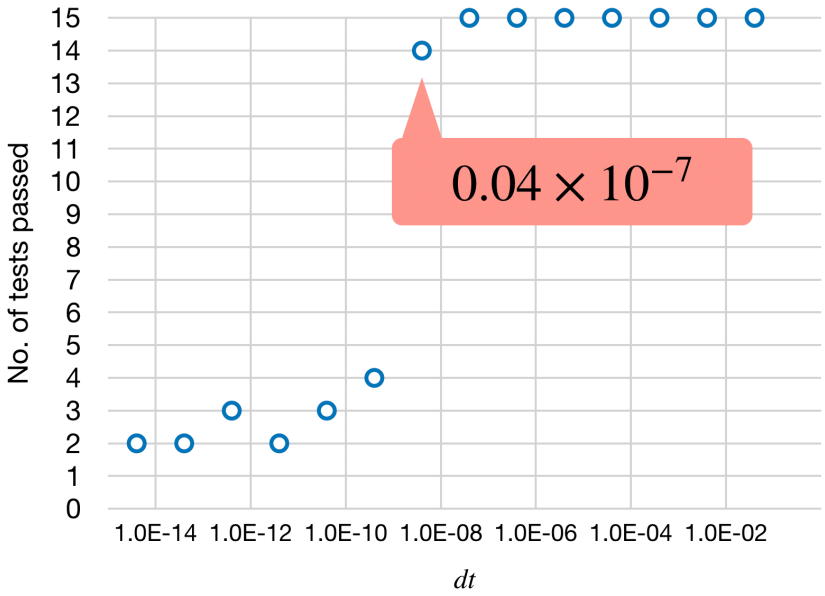


Figure 5.12: 異なる時間ステップ幅を用いた場合に，得られる乱数資源の乱雑性に対する NIST 乱数検定の結果（入力した乱数として，連続するステップの中で系のすべて粒子ペアに対して生成される乱数を用いた場合

5.4　新規乱数生成手法の計算コスト比較

本章で提案する新規乱数生成手法を従来の手法と比較し，計算コストの観点でどれだけの利点があるのかを検証する．

Table 5.1: $300K$ における各粒子間の反発パラメータ a_{ij}

a_{ij} の値	A	B	C	D	E	F	Water
A	25.00	24.41	24.22	29.55	40.53	40.57	64.53
B		25.00	22.97	27.32	42.85	44.00	66.40
C			25.00	28.89	40.73	43.84	59.56
D				25.00	31.51	11.92	56.82
E					25.00	33.34	3.29
F						25.00	44.62
Water							25.00

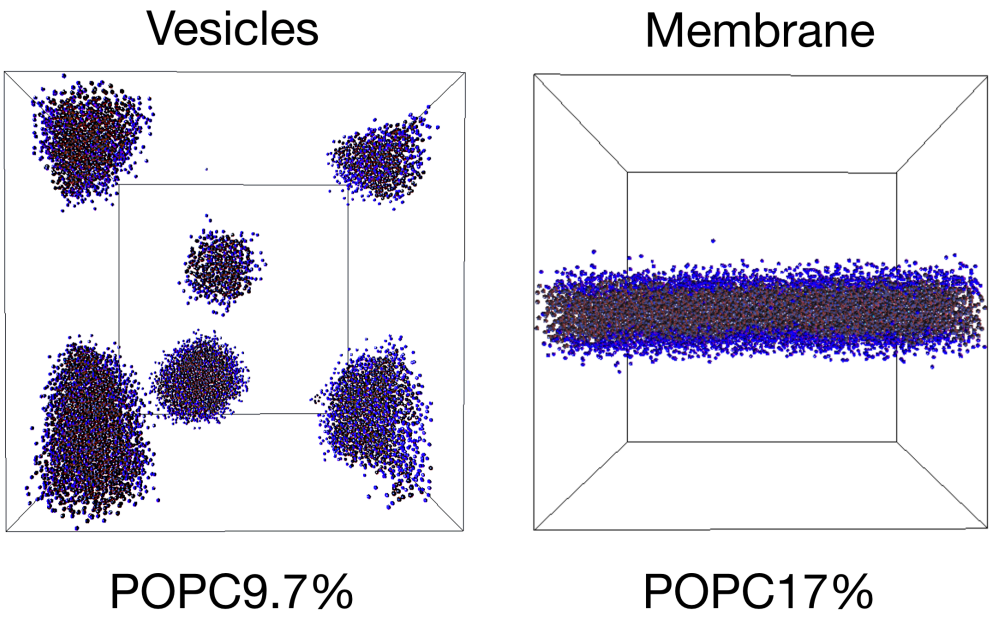


Figure 5.13: POPC 脂質を用いて再現した系．POPC9.7% では Vesicles，POPC17% では Membrane の形成を示す．

まず，乱数生成速度を調べるために，従来の手法と新規手法のそれぞれを用いて，区間 $[0,1]$ の一様分布に従う乱数 (double 型) を 100 億個生成し，要する時間を比較した．また，実験 1 と同様のパラメータを用いて，粒子数 108000 の系を再現し，力の計算に要する時間を比較した．また，効率的な分子シミュレーション計算を行うために FDPS を用いた．計算には，CPU : Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz, コンパイラ : Intel compiler version 18.0.0 を用いて行った．また，FDPS のパラメータとして，n_leaf_limit $= 40$, n_group_limit $= 64$ を用いた．この系のスナップショットを Fig. 5.15 に示す．

計算コストの比較結果を Fig. 5.16 に示す．なお，従来の 8 回ラウンドを用いる TEA を TEA(8)，1 回ラウンドを用いる TEA を TEA(1)，そして，新規提案手法を “SHIFT” と示す．5.16(a) より，乱数生成に要する時間は従来のものと比較して，ラウンド数が 1 回の TEA の場合は 1/8 に，新規手法を用いた場合は 1/62 になることがわかった．この速度は，32bit 精度の MT 手法で生成する場合に比べて約 6 倍高速である．5.16(b) では力の計算に要する時間の比較を示している．ここからわかるように，新規乱数生成手法を用いることで，従来の手法に比べて，計算

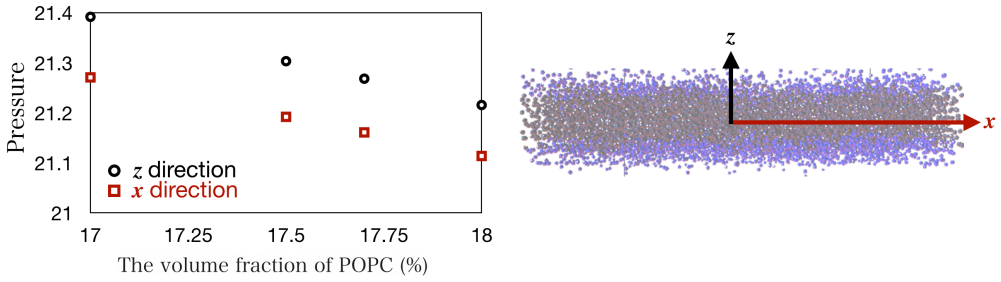


Figure 5.14: POPC 脂質の体積分率ごとの，生体膜の z 方向（膜に垂直方向），x 方向（膜に並行方向）のそれぞれに計測した膜圧

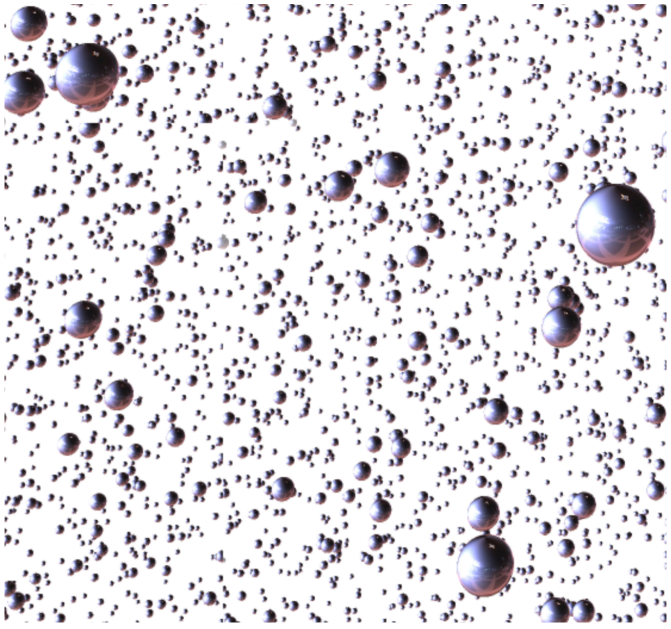


Figure 5.15: 108000 の粗視化水粒子系の DPD シミュレーション

コストが 27 %削減できている．

上述した力の計算コストの関係は，分子シミュレーションで用いるアルゴリズムに依存する．乱数生成に要するコストが 1/62 であるにも関わらず，力の計算コストに大きな影響が見られない理由として，粒子間距離の測定に時間を要していることが考えられる．前章で説明したとおり，分子シミュレーションではカットオフ半径を定義して，カットオフ半径距離よりも外に存在する粒子に対しては力の計算を行わない．したがって，それらの粒子の組に対しては乱数生成を行わない．カットオフ距離よりも外側に存在する粒子に対しては，距離判定のための計算を行わないような効率的な計算アルゴリズムを用いれば，更に力の計算コストを削減できる可能性がある．

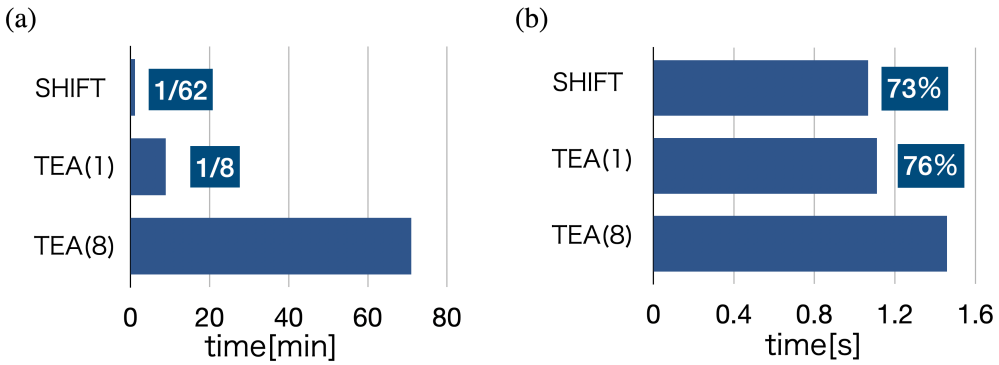


Figure 5.16: (a)100 億個の乱数を生成するのに要した時間の比較と，(b)108000 の粗視化水粒子系の DPD シミュレーションを行い，相互作用の計算に要した時間の比較（SHIFT は提案する新規乱数生成手法を示す.）

第 6 章

MC 法のための新規乱数生成手法の開発

本章では，新規に開発した乱数生成手法を MC 法に拡張し，その適用法と結果について述べる．新規乱数生成手法の特性と有用性を具体的に示すため，疑似乱数生成の影響が顕著に現れる小さい系へ適用し，検証する．

まず，新規乱数生成手法の設計と実装の詳細について，そしてそれが MC 法にどのように適応できるかについて詳しく説明する．

次に，新規乱数生成手法を用いたシミュレーションと既存の手法を用いたシミュレーションの物性値を比較する．それらの誤差を評価し，本手法の有用性を明確にする．

最後に，新規乱数生成手法を NIST 乱数検定により評価する．これにより，新規乱数生成手法が統計的なランダム性においても十分適用可能であることを示す．

6.1　新規乱数生成手法を実装したアルゴリズム

本章では，散逸粒子動力学法で開発した新規乱数生成手法を MC 法に適用する方法を説明する．これまでの章で検討した新規乱数生成手法の効率性と高速性は，多種多様なシミュレーション手法に適応可能であるという事実を示している．そこで，この章では，具体的にその手法を MC 法にどのように適用するかを説明し，新規乱数生成手法の拡張性と多様性について考察する．まず，一般的な MC 法は Fig. 6.1 の流れに従って計算し，実行される．

1. 初期条件の設定: 分子系の初期状態を設定する．初期状態には分子の位置や向き，速度などの情報が含まれる．また，シミュレーションパラメータ（例えば，温度，圧力，体積など）もこの段階で設定する．
2. 乱数生成と分子選択: 次に，乱数生成器を使用して特定の分子をランダムに選択する．これは，状態変化（移動，回転など）の対象となる分子を指す．
3. 試行移動: 選択した分子に対して，試行的な状態変化（例えば，位置のランダムな移動やランダムな回転）を行う．この操作も乱数を用いる．
4. エネルギーの計算: 試行的に変化させた後の新しい系の状態でのエネルギーを計算する．
5. Metropolis-Hastings 法による操作: 新旧の状態におけるエネルギー差に基づいて，試行的

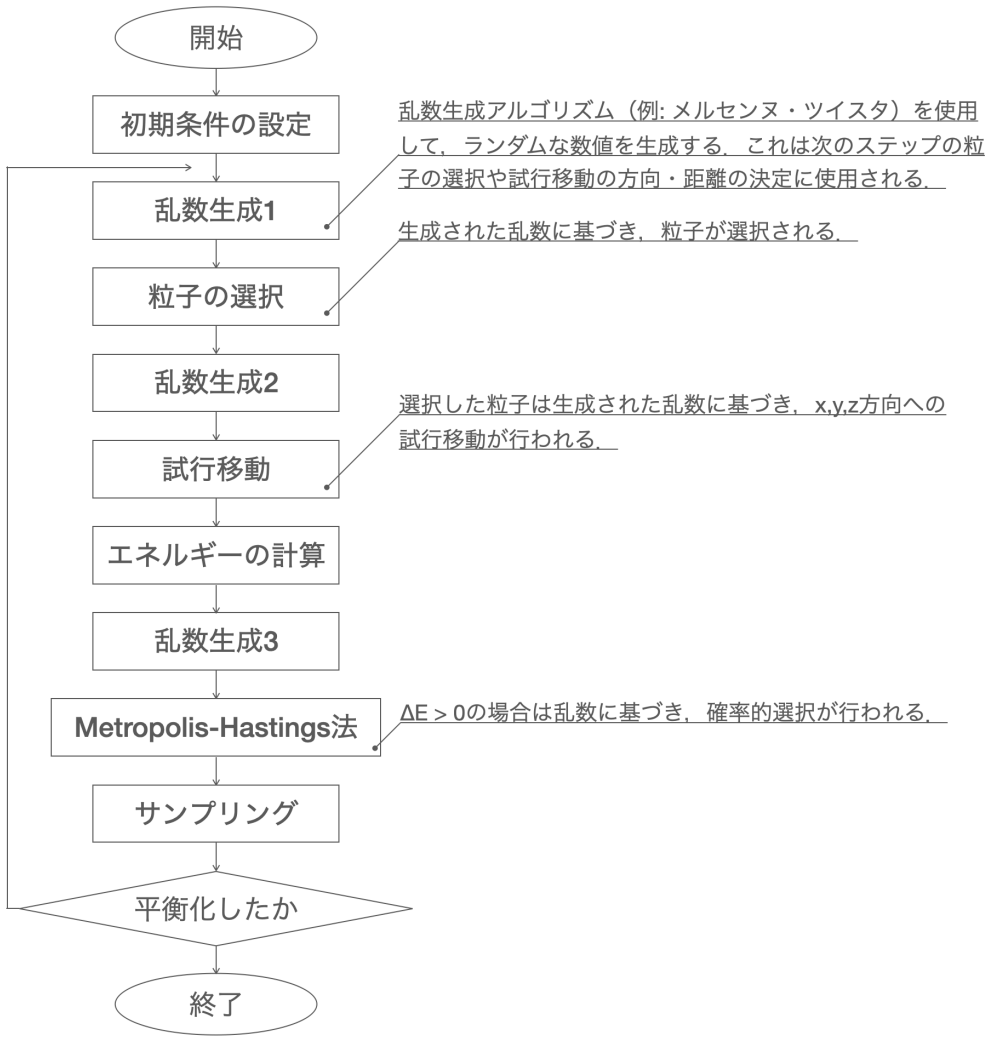


Figure 6.1: モンテカルロ分子シミュレーションの流れ

な変化を受け入れるか否かを決定する．ここではメトロポリス基準と呼ばれるアルゴリズムに基づいて行う．エネルギーが下がる（系がより安定になる）場合では変化は常に採択される．エネルギーが上がる場合では，ある確率に基づき，変化が採択される．

6. 更新: 試行的な変化が採択されると，系の状態は更新される．棄却される場合，系の状態は変化前の状態を保持する．
7. 出力と分析: 一定のステップ数ごとに，系の情報（例えば，エネルギー，分子の配置など）を出力する．
8. 反復: ステップ 2 から 7 を所定の反復回数，または所定の条件が満たされるまで（例えば，エネルギーが収束するまで）繰り返す．

ここまでにおいて，乱数生成は 3 回の場面で実行される．つまり，粒子選択の場面，選択した粒子に対して，試行移動を実行するための場面，Metropolis-Hastings 法において利用する場面の 3 回である．特に 2 回目の試行移動では，選択した粒子座標の x,y,z 成分全てに対して実行されるため，3 回の乱数生成が実行される．全体として，これらの乱数生成がシミュレーション分析に必要なステップの量だけ繰り返される．他のシミュレーション手法に比べても乱数生成の回数とそのコストが大きい手法といえる．

本研究では，ここで必要とする乱数生成を最低限の計算コストで実行するために Fig. 6.2 に示すようにアルゴリズムを提案した．一般的に利用されている乱数生成手法：MT 法の利用をすべて廃止し，新たな方法で乱数生成が実行できるように設計した．ここで，新たに 2 つの関数を定義し，用意する．

1. 排他的論理和関数, xor(x,y)：2 つの bit 列 x,y を排他的論理和による演算を実行する．
2. bit 抽出関数, shift-cut(x)：bit 列 x の 8-40 番目を抽出後，1-32 番目に配置，残りを 0 で埋める演算を実行する．

この関数を利用し，6.1 のアルゴリズムに 7 点の変更を加える

1. 初期条件の設定において，
   5 つの [符号なし 64bit 整数型変数: $seed_x, seed_y, seed_z, seed_{id}, seed_{pot}$] を用意し，乱数により初期化する．
2. 乱数生成 1 において，[rnd_{id} =shift-cut(rnd_{id})] を生成する．
3. 粒子の選択において，選択する粒子を [rnd_{id}] により決定する．
4. 粒子の選択の直後に下記を実行する．
   xor($seed_x$, 粒子 $id = [rnd_{id}]$ の y 軸座標)
   xor($seed_y$, 粒子 $id = [rnd_{id}]$ の z 軸座標)
   xor($seed_z$, 粒子 $id = [rnd_{id}]$ の x 軸座標)
   xor($seed_{id}$, 粒子 $id = [rnd_{id+1}]$ の y 軸座標)
   xor($seed_{pot}$, 粒子 $id = [rnd_{id+1}]$ の z 軸座標)

5. 乱数生成 2 において，粒子 $[id = rnd_id]$ の試行移動量をそれぞれ下記に基づき生成する．
   rnd_x =shift-cut($seed_x$)
   rnd_y =shift-cut($seed_y$)
   rnd_z =shift-cut($seed_z$)

6. エネルギーの計算の直後に下記を実行する．
   xor($seed_x$, 試行移動による局所エネルギー更新値)
   xor($seed_y$, 試行移動による局所エネルギー更新値)
   xor($seed_z$, 試行移動による局所エネルギー更新値)
   xor($seed_{id}$, 試行移動による局所エネルギー更新値)
   xor($seed_{pot}$, 試行移動による局所エネルギー更新値)

7. Metropolis-Hastings 法において，必要に応じて乱数 [$shift - cut(rnd_{pot})$] を生成する．
   生成した場合，下記を実行する．
   xor($seed_x$, メトロポリス移動確率)
   xor($seed_y$, メトロポリス移動確率)
   xor($seed_z$, メトロポリス移動確率)
   xor($seed_{id}$, メトロポリス移動確率)

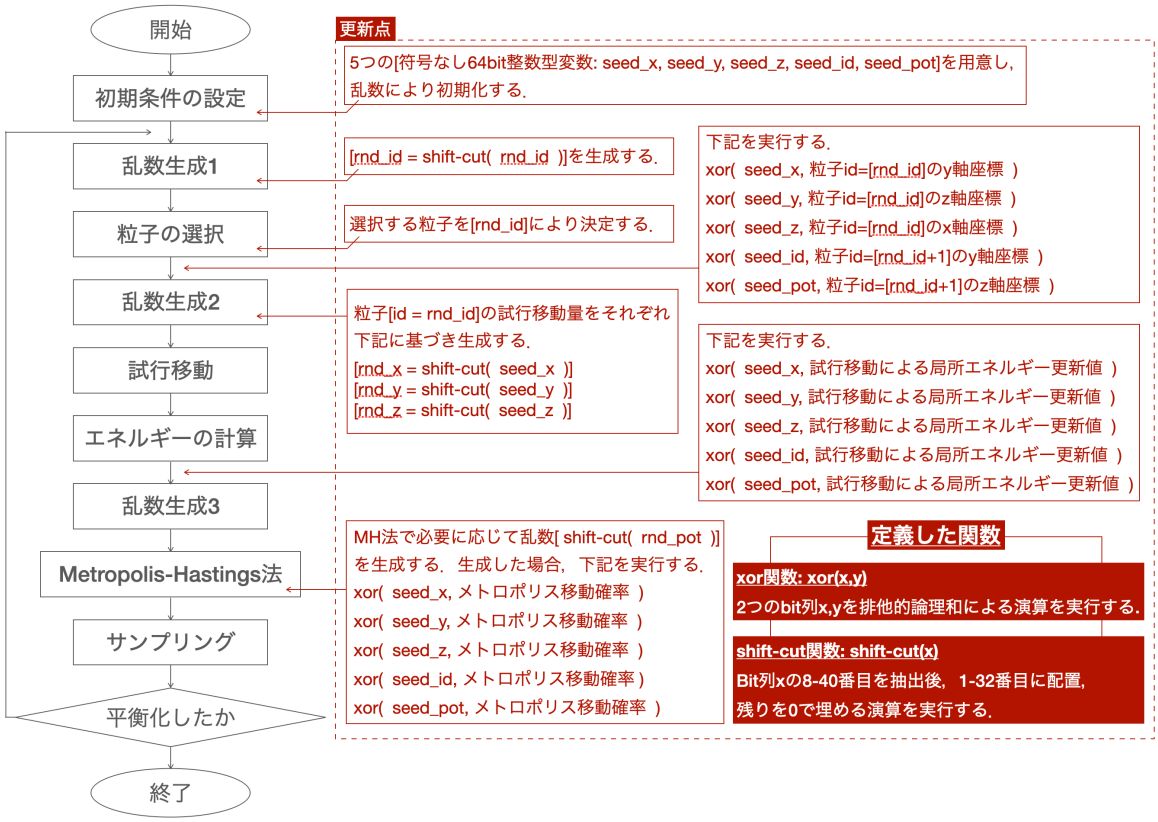


Figure 6.2: 新規乱数生成手法を実装したモンテカルロ分子シミュレーションの流れ

$xor(seed_{pot}$, メトロポリス移動確率$)$

以降の MC シミュレーションでは，MT などの乱数生成手法を用いて計算する従来の計算方法（Fig. 6.1）と，ここで新たに提案した，新規乱数乱数生成手法を実装した MC シミュレーション（Fig. 6.2）を用い，それぞれの計算結果を比較した．

6.2　物性値の比較

本章では，新規乱数生成手法を用いたモンテカルロ分子シミュレーションの物性値と，従来手法によるシミュレーションの物性値を比較する．エネルギー値，動径分布関数で評価される性質の比較は，シミュレーションが物理的現象を適切に再現できているかを示す重要な指標である．それらの指標を用いて，新規乱数生成手法の有効性を示す．

6.2.1　エネルギー値の比較

シミュレーション中に計算されるエネルギーは，物質の状態を理解するための指標の一つである．エネルギーは，分子間の相互作用の強さを表し，分子の動きや配列に影響を与える．ここでは，異なる密度と温度で MC シミュレーションを行い，それぞれの条件でのエネルギーを計算し比較した．密度と温度は物質の状態を決定する基本的なパラメータである．エネルギーはこれらのパラメータに反応し，物質の振る舞いを決定する．Figs. 6.3，6.4 に従来の手法 MT を用いた

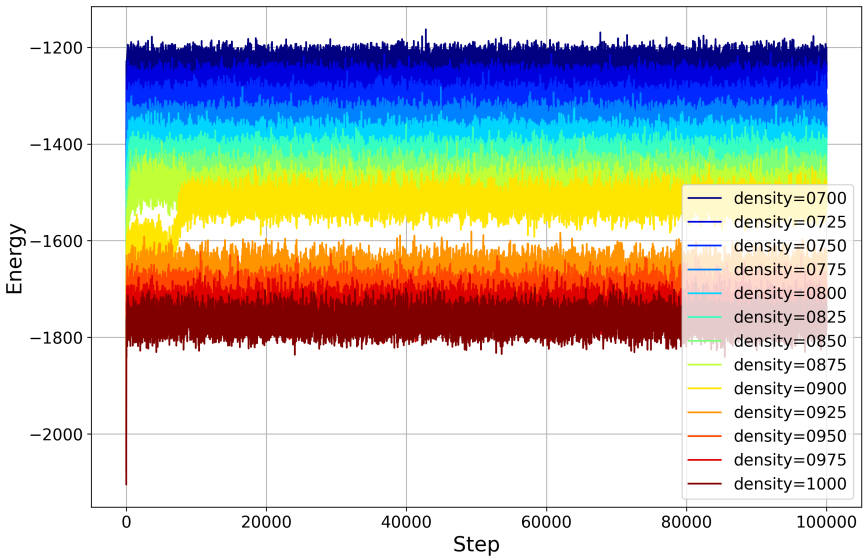


Figure 6.3: 各密度におけるエネルギーの値（MT を利用）

場合に計算した各密度，各温度におけるエネルギーの値を，Figs. 6.5，6.6 に本研究で開発した乱数生成手法を用いた場合に計算した各温度，各密度におけるエネルギーの値について示した．各密度，温度において，平衡状態に至るまでの差があることが観察できた．次に，それらのエネルギー値から，平衡状態に達した後の 50000 ステップにおいて平均値を計算した．Figs. 6.7，6.10 に従来の手法 MT を用いた場合に計算した各密度，各温度におけるエネルギー平均値を，Figs. 6.9，6.10 に本研究で開発した乱数生成手法を用いた場合に計算した各温度，各密度におけるエネルギー平均値を示した．気体，液体それぞれにおいて，また気液相転移の前後におけるエネルギー平均値のが類似していることが観察できた．また， Figs. 6.11，6.12 に従来の手法 MT を用いた場合と，新規手法を用いた場合で得られたエネルギー差を各密度，各温度において示した．従来手法 MT を用いた場合の計算値と，新規提案手法を用いた場合の計算値の差は全てにおいて，0.6% 以下であることが確認できた．なお，物性値の差は実験 1 と同様の方法を用いて計算した．

6.2.2　動径分布関数の比較

同様に MC シミュレーションにて，動径分布関数の解析を行った．異なる条件下での動径分布関数の解析と比較を行うことで，物質の各状態の構造と性質の考察が可能であり，シミュレーションの妥当性を確認することができる．

Figs. 6.13，6.14 に従来の手法 MT を用いた場合に計算した各密度，各温度における動径分布関数を，Figs. 6.15，6.16 に本研究で開発した乱数生成手法を用いた場合に計算した各温度，各密度における動径分布関数について示した．また， Figs. 6.17，6.18 に従来の手法 MT を用いた場合と，新規手法を用いた場合で得られた動径分布関数の差を各密度，各温度において示した．従来手法 MT を用いた場合の計算値と，新規提案手法を用いた場合の計算値の差は概ね，0.5% 以下であることが確認できたが，一部の状態（density= 1.000, 0.95, 0.925, temperature = 0.200）においてはその差が 1% を超える場合が観測された．これらはいずれも気液相転移点の前後であ

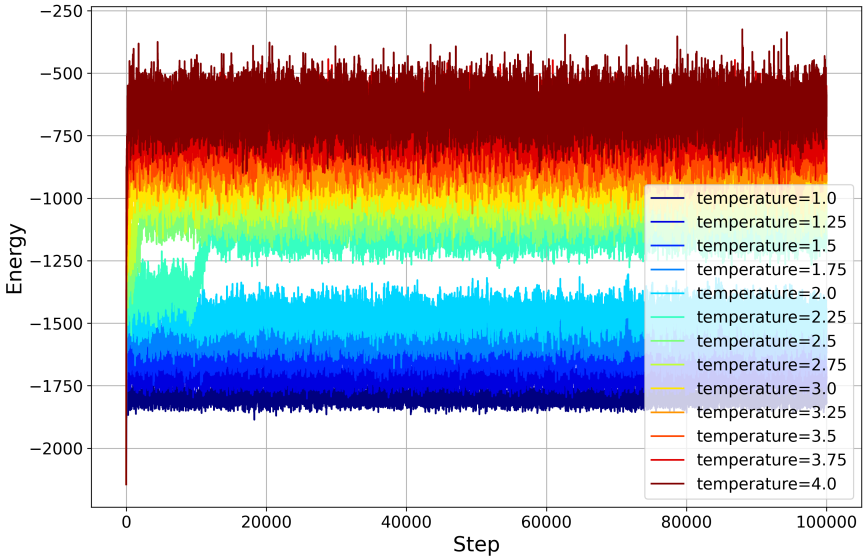


Figure 6.4: 各温度におけるエネルギーの値（MT を利用）

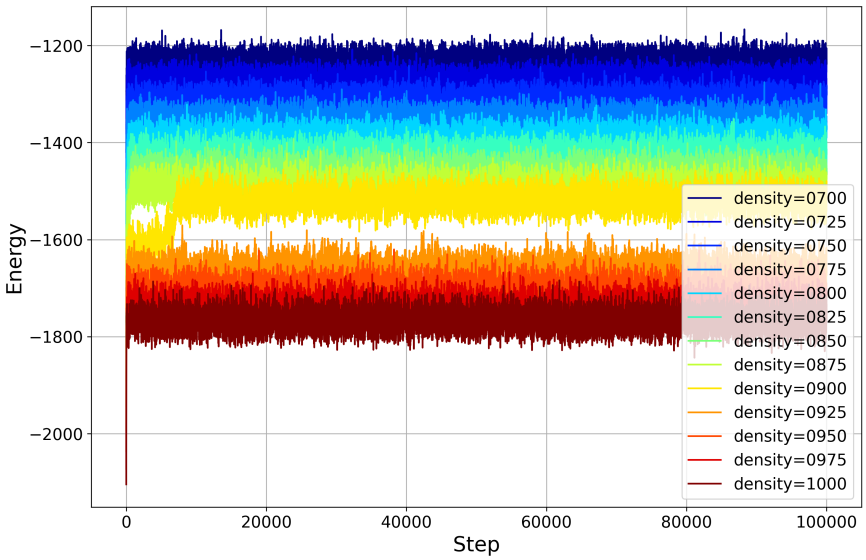


Figure 6.5: 各密度におけるエネルギーの値（新規乱数乱数手法を利用）

る．分子配置が気体の状態と液体の状態を行き来し，この設定条件における動径分布関数の値が他の状態に比べて不安定であることがうかがえる．なお，物性値の差は実験 1 と同様の方法を用いて計算した．

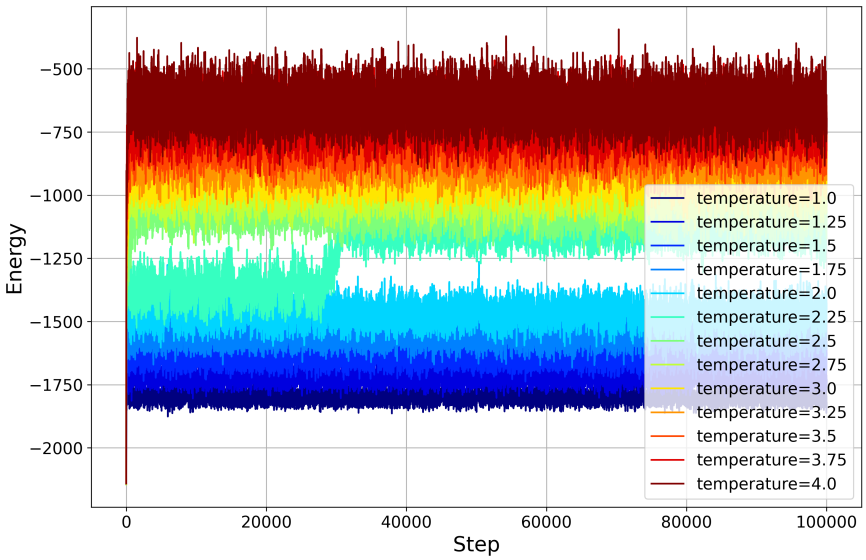


Figure 6.6: 各温度におけるエネルギーの値（新規乱数乱数手法を利用）

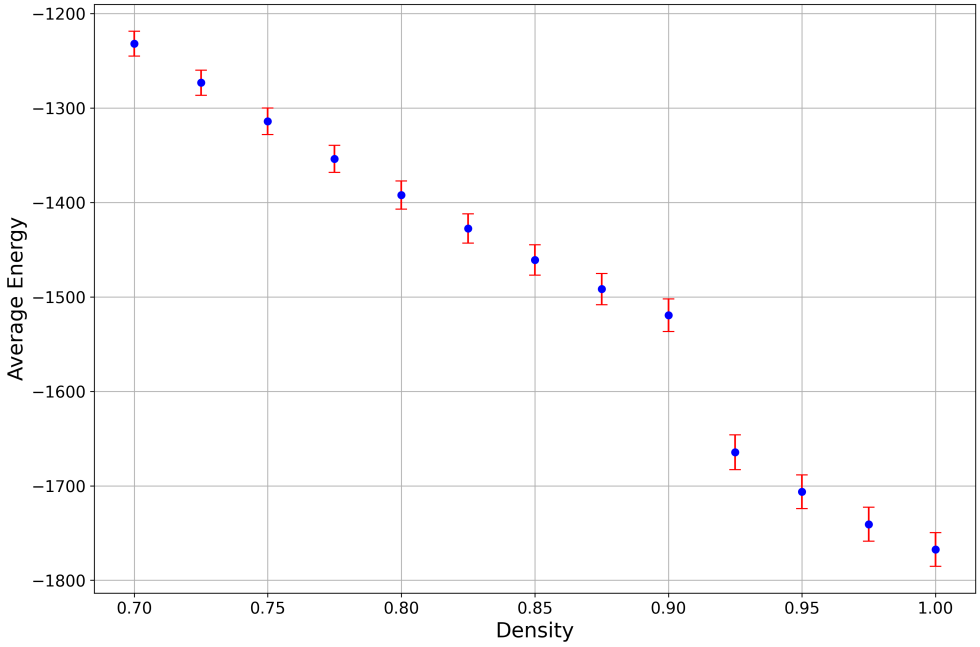


Figure 6.7: 各密度におけるエネルギーの平均値（MT を利用）

6.3　乱数生成手法 3XORs の汎用性と拡張性

本研究では，新しい乱数生成手法「3XORs」を提案した．3XORs 法は，排他的論理和関数（xor）とビット抽出関数（shift-cut）を基本演算として使用し，分子シミュレーションの過程で導出される物性値の乱雑性を利用して乱数を生成する方法である．3XORs は従来の乱数生成器のような複雑な演算を利用しないため，計算コストの削減と効率の向上が期待できた．一方で，3XORs が

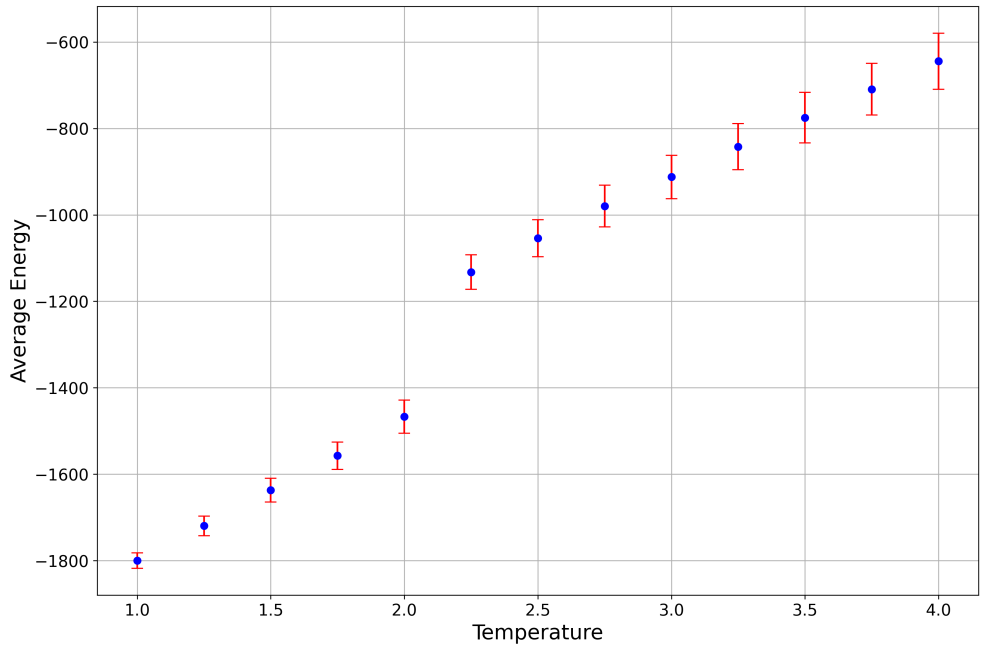


Figure 6.8: 各温度におけるエネルギーの平均値（MT を利用）

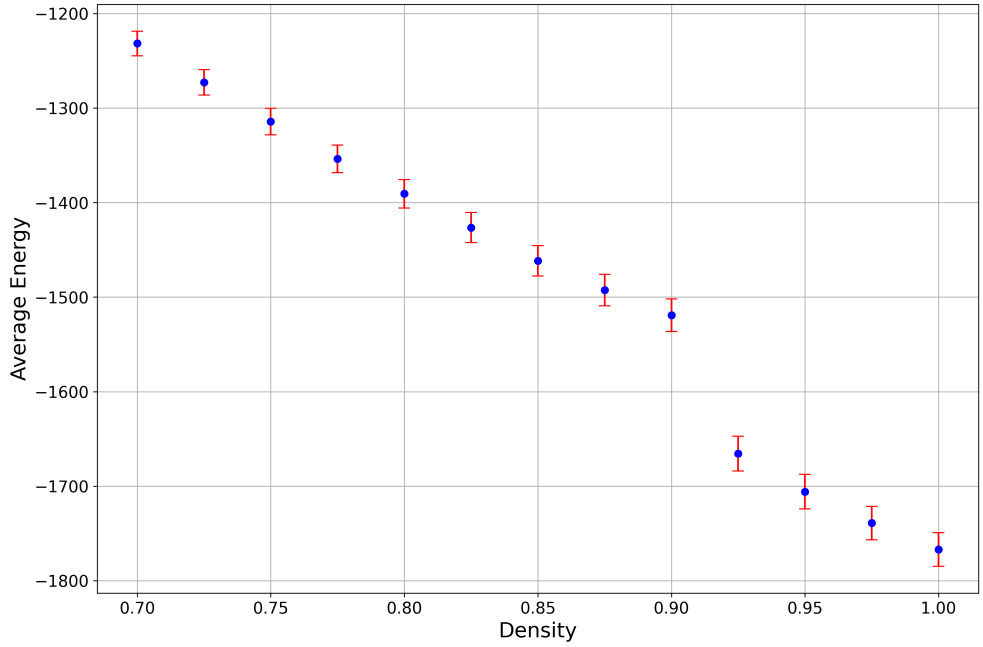


Figure 6.9: 各密度におけるエネルギーの平均値（新規乱数乱数手法を利用）

生成する乱数がどのようなシミュレーション系においても安全に使用できるかどうかの汎用性と，水分子などの異方性のある分子を取り扱う場合にどう利用するかの拡張性について検討する必要がある．ここではその汎用性と拡張性についてそれぞれ説明する．

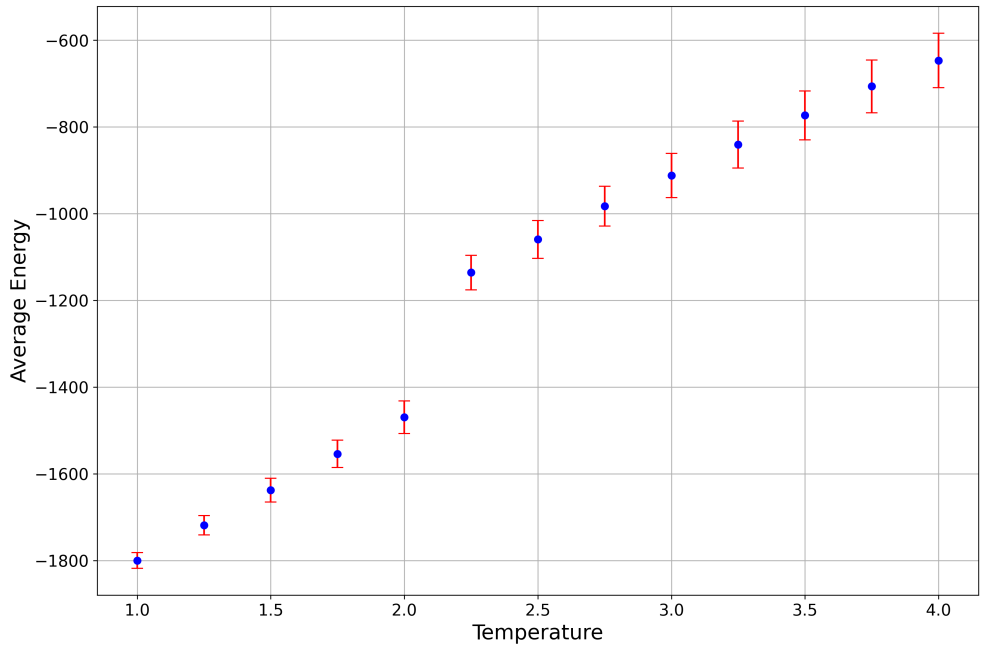


Figure 6.10: 各温度におけるエネルギーの平均値（新規乱数乱数手法を利用）

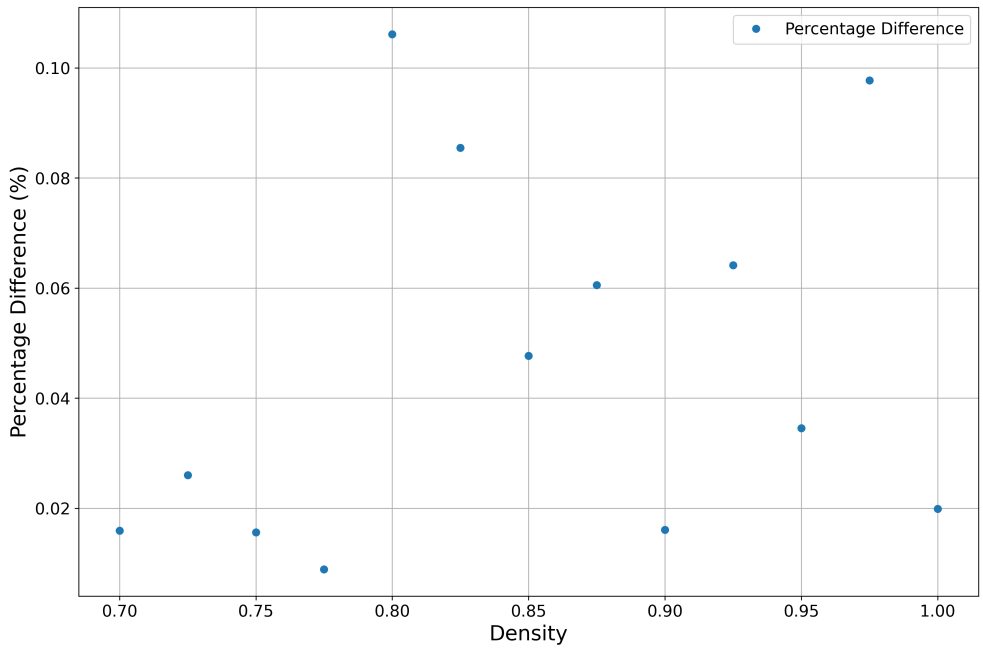


Figure 6.11: 各密度におけるエネルギーの差（%）

6.3.1　汎用性

この乱数生成アルゴリズムの重要な点は，シミュレーションの計算過程で必然的に導出される粒子の座標，粒子の試行移動に伴う局所的なエネルギーの変化量，またメトロポリス法に基づく採択確率に対して乱数シードと 3 回の排他的論理和関数（xor）を利用して，乱数シードに乱雑性を付与する点であった．この手順は，シミュレーションの各ステップで繰り返され，必要な乱

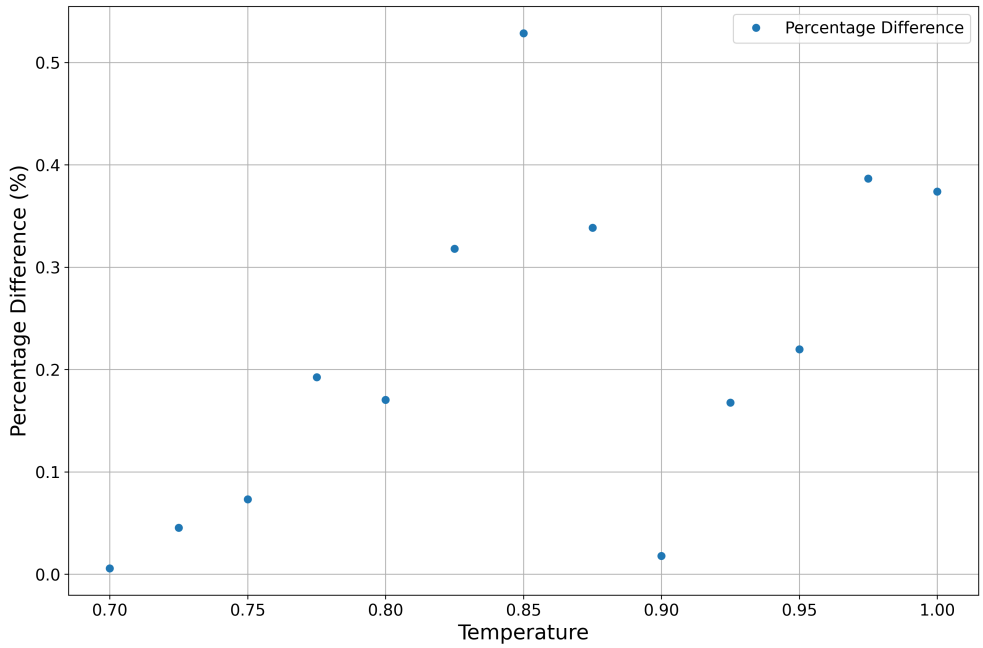


Figure 6.12: 各温度におけるエネルギーの差（%）

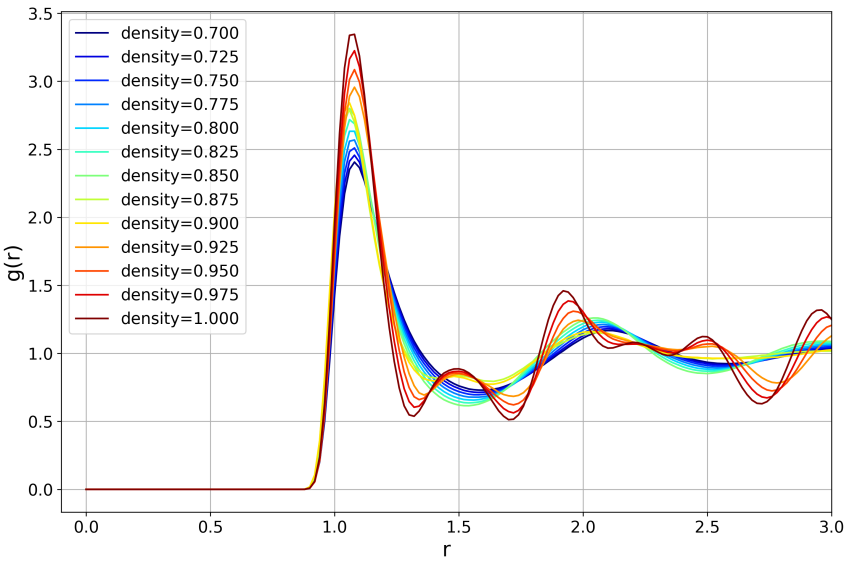


Figure 6.13: 各密度における動径分布関数（MT を利用）

数を提供する．性質の良い乱数が生成され，適切にシミュレーションが実行できているかどうかを確かめるために，エネルギーと動径分布関数等の物性値を対象として比較を行った．さらなる検証として，生成される乱数を評価し，十分な性質を有しているかどうかを NIST 乱数検定を用いて評価することが重要である．性質の良い乱数でなかったとしても，乱数性質にロバストな物理シミュレーション法では，その結果に大きな影響を与えない可能性があるからである．乱数性質を評価するため，本研究では特に，粒子座標が大きく変化しない計算系である個体の系 (密度 1.05, 温度 1.0) を対象に，生成される乱数性質を評価した．これは，液体などの粒子座標が大きく

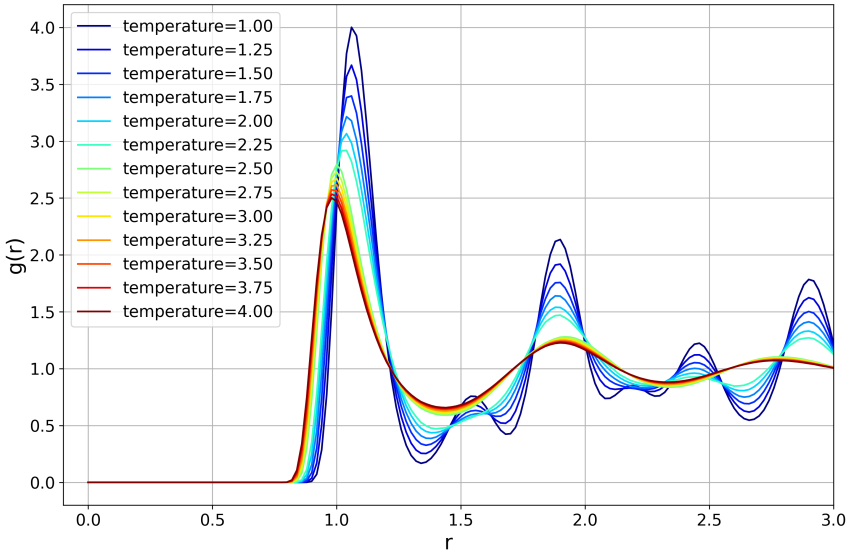


Figure 6.14: 各温度における動径分布関数（MT を利用）

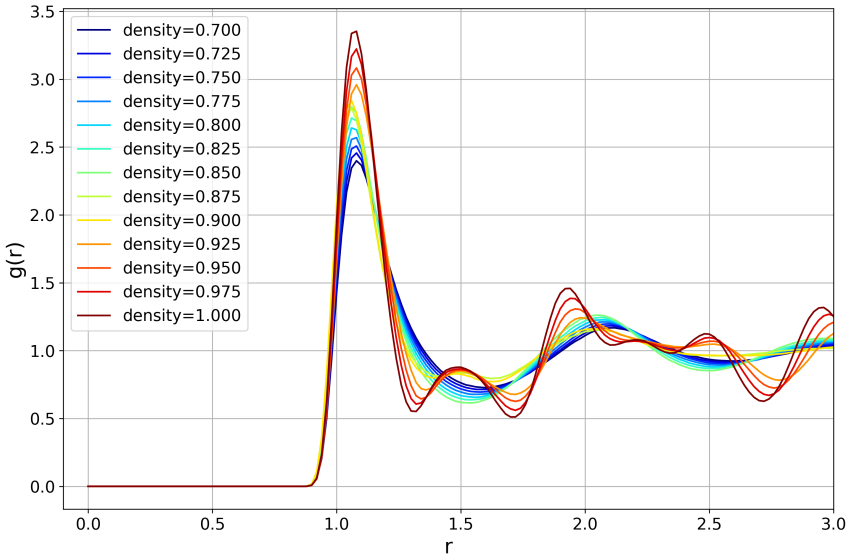


Figure 6.15: 各密度における動径分布関数（新規乱数乱数手法を利用）

更新される系に比べて，座標の変化量が小さく，乱数生成に比較的不利であると考えたためである．モンテカルロ法を用いて個体の系 (密度 1.05, 温度 1.0) のシミュレーションを行った場合に，3XORs 法によって生成した乱数を NIST 乱数検定で評価した結果を表 6.1 に示す．すべてのテスト項目において，乱数検定を合格したことを確認した．これらの結果から，特に次の 2 つの点が重要と考えた．

1. 個体の系の計算で粒子座標が大きく変化しなかったとしても，粒子座標は微細なゆらぎ

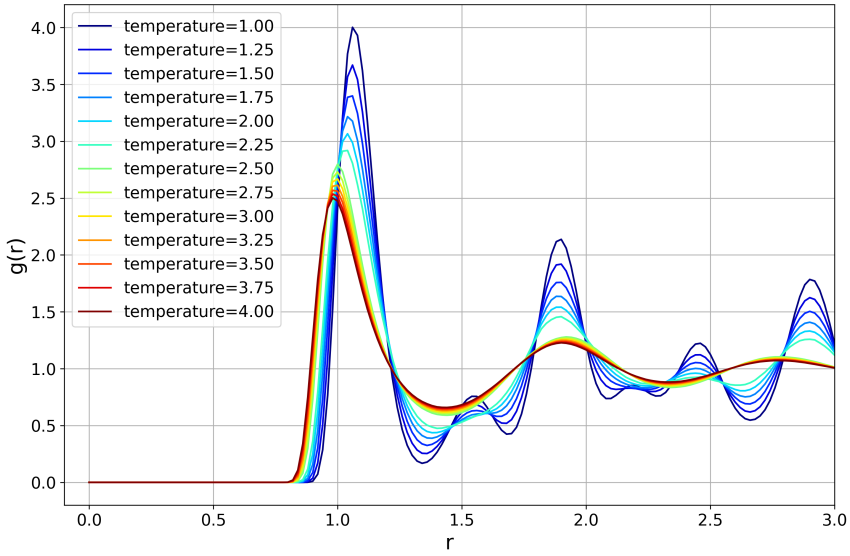


Figure 6.16: 各温度における動径分布関数（新規乱数乱数手法を利用）

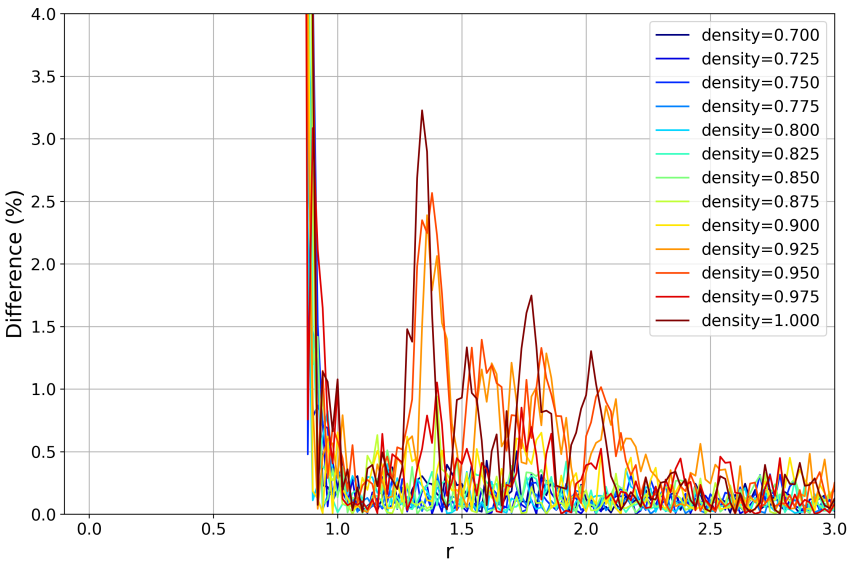


Figure 6.17: 各密度の動径分布関数の差（%）

を持っており，小数点以下が複雑に変化しているため，十分に乱雑性を取り込むことができた点.

2.　乱数により, 選択される粒子 id = i の座標だけでなく，粒子 id = i+1 の座標の乱雑性も取り込んだ点．長いシミュレーションの実行により，初期配置によらず粒子 id = i と粒子 id = i+1 の粒子座標の相関は小さくなるが，これはたとえ粒子番号が隣同士であっても，モンテカルロ法の各ステップにおいて選択する粒子は乱数によって選択されるため, 粒子 id=i と id = i+1 に計算順序に基づく相関は小さいと考えられる.

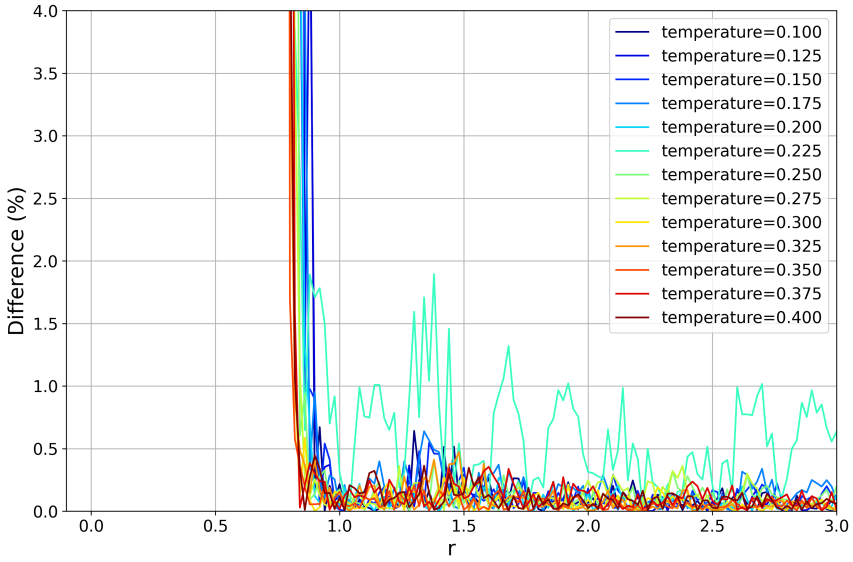


Figure 6.18: 各温度の動径分布関数の差（%）

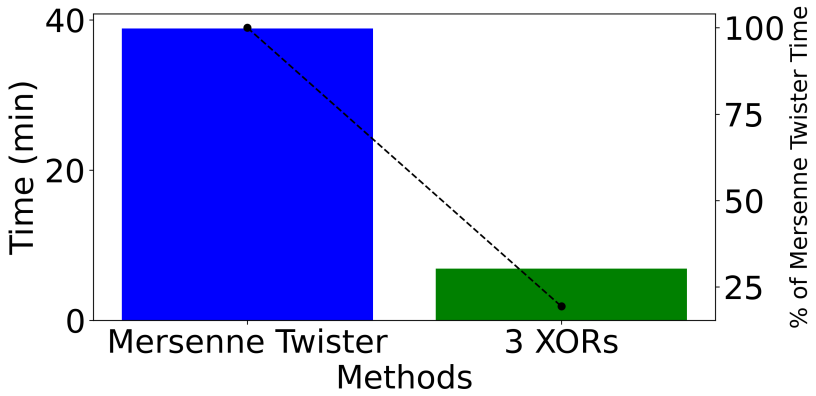


Figure 6.19: 3XORs 法と MT 法のそれぞれを用いて 100 億個の乱数を生成するのに必要とした時間

したがって，分子にゆらぎを持つ系を扱う限り，様々な計算系において従来よりも効率的に乱数を生成できる．Fig.6.19 に 3XORs 法と MT 法のそれぞれを用いて 100 億個の乱数を生成するのに必要とした時間を示す．本研究で提案した 3XORs 法は MT 法に比べて約 5.6 倍に乱数を生成できることを確認した．

6.3.2　拡張性

水分子などの異方性のある分子を取り扱う場合，例えば水分子を剛体で表すとすると，重心の並進運動に加え，剛体の回転運動を取り扱う必要がある．したがって，一つの分子が 1step で移動するのに必要とする乱数の個数が増える．本研究で開発した 3XORs 法のアルゴリズムでは一つの分子が 1step で移動するのに必要な乱数の個数は 5 個であり，次の内訳であった．

Table 6.1: NIST statistical test results for random numbers in MC simulations.

Statistical test	**Pass rate**	**Uniformity of p-values**	**Pass/fail judgment**
Frequency	0.994	Pass	Pass
BlockFrequency	0.988	Pass	Pass
CumulativeSums	0.991	Pass	Pass
CumulativeSums	0.991	Pass	Pass
Runs	0.988	Pass	Pass
LongestRun	0.986	Pass	Pass
Rank	0.992	Pass	Pass
FFT	0.993	Pass	Pass
NonOverlappingTemplate	0.990	148/148 Pass	Pass
OverlappingTemplate	0.989	Pass	Pass
Universal	0.990	Pass	Pass
ApproximateEntropy	0.989	Pass	Pass
RandomExcursions	0.992	8/8 Pass	Pass
RandomExcursionsVariant	0.991	18/18 Pass	Pass
Serial	0.990	Pass	Pass
Serial	0.988	Pass	Pass
LinearComplexity	0.998	Pass	Pass

1. 乱数 1 ←粒子 $id = i$ の x 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
2. 乱数 2 ←粒子 $id = i$ の y 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
3. 乱数 3 ←粒子 $id = i$ の z 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
4. 乱数 4 ←粒子 $id = i + 1$ の x 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
5. 乱数 5 ←粒子 $id = i + 1$ の y 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor

重心の並進運動に加え，回転運動を取り扱う場合は，剛体の回転を記述するために追加で 3 つのオイラー角 α, β, γ を用いる．したがって，剛体の水分子の座標は，位置ベクトル $\mathbf{r} = (x, y, z)$ とオイラー角 (α, β, γ) によって定義される．このとき，一つの分子が 1step で移動するのに必要な乱数の個数は 8 個となり，次のようにアルゴリズムを修正できる．

1. 乱数 1 ←粒子 $id = i$ の x 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
2. 乱数 2 ←粒子 $id = i$ の y 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確

率と xor

3. 乱数 3 ←粒子 $id = i$ の z 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
4. 乱数 4 ←粒子 $id = i$ のオイラー角 α と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
5. 乱数 5 ←粒子 $id = i$ のオイラー角 β と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
6. 乱数 6 ←粒子 $id = i$ のオイラー角 γ と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
7. 乱数 7 ←粒子 $id = i + 1$ の x 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor
8. 乱数 8 ←粒子 $id = i + 1$ の y 座標と xor, 局所エネルギー更新値と xor, メトロポリス採択確率と xor

新たに定義した変数のオイラー角と粒子 $id = i + 1$ との粒子座標を追加で利用することで，任意の個数だけ乱数を追加で生成することができる．

6.4 モンテカルロ分子シミュレーションのための乱数生成手法のまとめと今後の展望

アルゴリズムの概要本研究で開発されたアルゴリズムは, モンテカルロ分子シミュレーションにおける乱数生成の効率化を目的としている．このアルゴリズムは, 排他的論理和関数（xor）とビット抽出関数（shift-cut）を基本演算として使用し, シミュレーション中に生成される乱雑性のあるデータから直接乱数を生成する．この方法は, 従来の乱数生成器に依存しないため, 計算コストの削減と効率の向上が期待できる．

乱数生成の原理本研究で提案した手法は, シミュレーションの計算過程で副次的に発生する数値の乱雑性を利用して乱数を生成する点にある．本研究で利用した粒子の座標データは, 分子シミュレーションの基本的な物理量であるが，それ以外にも速度など，計算過程で出力する物理量を利用することで, 様々なシミュレーションモデルや条件下でも効率的な乱数生成が可能となることが期待できる．

乱数の品質保証本研究では, 粒子座標が大きく変化しない計算系として，個体系のシミュレーションにおいても,NIST 乱数検定に合格する乱数が生成されることを確認した．

結論良質な乱数性質を持つ bit 列が，シミュレーションの計算過程に生じることを発見した．そして，乱数性質を有する bit 列を利用することで，従来よりも高速な乱数生成を実現できる一例を示した．

第 7 章

背景：深層モデルを活用した乱数生成手法について

7.1　効率的な乱数生成手法の開発のために深層学習モデルを利用する意義

これまでに述べた実験により，特定のシミュレーションだけでなく，モンテカルロ法などを含む多種シミュレーション手法にも同様に効率的な乱数生成手法が適用可能であることを示唆している．それぞれの手法に特化した乱数生成手法を開発することは，シミュレーションの効率性を向上させる上で重要である．一方で，本研究で開発に取り組んだ乱数生成手法は，分子シミュレーションの計算過程で生じる乱雑性を，そのまま乱数として利用するアプローチであった．そのため，各シミュレーション手法で必要となる乱数の個数と，計算過程で生じる乱雑性に応じて，アルゴリズム全体を再度設計し直す必要がある．このような各種法に特化した乱数生成手法を都度開発することは，多くの労力と時間を必要とするだけでなく，開発者の専門性に依存する問題がある．そこで本研究では深層学習技術に着目した．深層学習技術は，開発者の明示的な指示なしに，人為的には困難な統計的特徴を自動抽出することが可能であるため，画像識別処理を始めとして，近年多くの領域で利用されている．深層学習モデルを用いた半自動的な用途特化型の乱数生成アルゴリズムの開発が達成できれば，各需要に合わせた乱数生成手法を開発する際に，人的な負担を軽減できる可能性がある．また，深層学習を利用した乱数生成技術はシミュレーションへの利用にとどまらず，様々な応用先が考えられる．一方で，現状では深層学習を用いた乱数生成手法の開発は容易ではない．統計的に一定の特徴を持たない乱数の模倣を深層学習により達成するという課題は，深層モデルの能力を評価する上でも重要な研究の一つである．以降では，統計的に特徴を持たない乱数生成技術の背景を記述しながら，深層学習を用いてその模倣に取り組んだ先行研究に，本研究がどのように連結するのかを述べる．

7.2　深層学習モデルを用いた乱数生成手法開発の一般的な背景

さまざまな分野の情報システムにおいて求められる予測不可能で非恣意的な挙動への需要が増えつつある中で，乱数は予測不可能でダイナミックな振る舞いを実装する上で重要な役割を果た

している [92]．例えば，暗号化 [93, 94] への利用により情報の安全性を担保し，ゲーム [95] では公正さを実現するための基本的なツールとして，機械学習 [96, 97] では乱数を用いた効率的な計算に利用されている他，分子シミュレーション [74–76] やフェーズフィールドシミュレーション [98, 99] などの幅広いシミュレーション手法においても効率的に物理現象を取り扱うために乱数が利用されている．乱数を生成するために一般的に使用されるツールは 2 つあり，物理現象を利用した真の乱数生成器（TRNGs）とソフトウェアアルゴリズムとして実装された疑似乱数生成器（PRNGs）である．TRNGs は，オペレーティングシステムのユーザープロセスの時間的特性，熱雑音，ショットノイズ，電子ノイズ，放射性崩壊の放出タイミングなどのランダム性を持つ物理現象を採用して乱数を生成する．多くの研究者がより効率的な TRNGs を開発しており，低速なもので 300 Gb/s のランダムビット生成（Random Bit Generation: RBG）[100–102]，高速なもので 2 Tb/s の RBG [103–114]，そして，Kim らによって最速となる 250 Tb/s の RBG が達成された [87]．このような TRNGs は，セキュリティや金融など，真の予測不能性が求められる高リスクの領域でよく利用されている．TRNGs の主な欠点は，物理現象に依存し特定のハードウェアが必要なため，PRNGs と比較して多くの乱数を生成するのに時間がかかることである．PRNGs は多年にわたり汎用ソフトウェアモジュールとして開発され，セキュリティ分野など，情報システムで大量の乱数が必要な領域で広く採用されている．PRNGs は，十分に複雑でランダムに見える数列を高速に生成するアルゴリズムである．PRNG は，NIST 乱数検定 [85, 115] などの統計テストスイートを通じて検証され，これは生成された乱数の堅牢性と公平性を評価するために一般的に使用され，15 の統計テストから成り立っている．MT 法 [21] や Xorshift 法 [116] などの PRNG アルゴリズム，および暗号化 PRNG は決定論的であり，「シード」と呼ばれる同じ初期値が入力されると，一意の周期で同じ数列を生成する．

多くの PRNG アルゴリズムはオープンで一般に公開されているため，オンラインアプリケーションやサービスに PRNG を利用する際は，シードと実装コードを隠すために細心の注意が必要である．有識者であるエンジニアがリバースエンジニアリングによってシード番号とアルゴリズムを知れば，PRNG の周期的な振る舞いを予測することは容易である．多くのアプリケーション開発者は PRNG を利用する際の優れた取り組みとしてシードの暗号化とコードの難読化を採用しているが，最新のスマートフォンアプリケーションにおいて実装の詳細を完全に隠すことは困難である．高度な数学的スキルを持たないアプリケーション・ソフトウェア開発者が，公的に動作が予測不可能で，NIST 乱数検定を完全に満たすオーダーメイドの PRNG を作成できるようにするために，新しい乱数生成モデルを発明することが急務となっている．

機械学習（Machine Learning: ML）技術は，十分な利用可能データが揃えば，明示的な指示なしにコンピュータシステムがタスクを実行するための統計的な特徴を学習することができる．これらの ML 技術は新たな PRNG を作成するために有益と考えられてきた．現在広く使用されている統計モデルは，ニューラルネットワークと呼ばれる関数近似手法の一種である．多くの分野で深層ニューラルネットワーク（Deep Neural Network: DNN）が成功裏に採用されたことを受けて [117]，多くの研究者が既存の PRNG の数学関数を近似することで PRNGs を作成しようと試みている．例えば，Elman ニューラルネットワーク [23]，リカレントニューラルネットワーク [24, 25]，長短期記憶ユニット（LSTMs）を持つリカレントニューラルネットワーク [26, 27]，Hopfield ニューラルネットワーク [28–30] は，PRNG を生成するために適用されてきたよく知ら

れたリカレントニューラルネットワーク (Recurrent neural network: RNN) である．過去 10 年間で，強化学習 [27,118] や生成的敵対ネットワーク（GANs）[31,32] などの新しい深層学習モデルを使用した PRNG 方法が登場し，研究者たちの注目を集めている．

多くの ML 技術が利用可能になったことを受けて，最近の重要な課題は，既存の PRNG の振る舞いを前処理や後処理なしに直接学習する ML 技術によって，エンドツーエンドの学習により新たな PRNG をどのように作成するかというものである [31]．エンドツーエンドの深層学習は，初期データ入力から最終結果出力までの多段階の処理を伴う複雑な ML システムを，さまざまなプロセスを実行する複数の層とモジュールを備えた単一の大規模なニューラルネットワークに置き換わることができる．このようなエンドツーエンドで完全に自動化された PRNG の生成は，一般的なアプリケーション開発者が自身のアプリケーションやシステムにカスタマイズした PRNG を作成することを可能にする．これは分子動力学法を始めとした一般的な分子シミュレーション手法も例外ではない．エンドツーエンドのニューラルネットワーク PRNGs の例として，De Bernardi ら [31] や Oak ら [32] は，GANs [34] を適用して，エンドツーエンドの乱数生成を可能にする既存の PRNGs の振る舞いを学習した．しかし，これらの研究 [31,32] は，1,000 回のテスト実行（例えば，前述の研究では 10 回のテスト実行）を含む推奨される NIST 乱数検定の設定を採用していない．したがってこれまでは，推奨される評価基準の下で NIST 乱数検定を完全に満たす ML ベースまたは DNN ベースの PRNG は存在しなかった．NIST 乱数検定の公式に推奨される設定を完全に満たす学習した乱数生成器を実現する新しい方法は，実用レベルの信頼性を持つ PRNG のエンドツーエンド学習を達成するために求められている．

7.3 深層学習を用いた新規乱数生成手法に関する研究目的

これまでの章では，乱数生成手法に着目し，各分子シミュレーション手法に特化したアルゴリズムを開発してきた．従来よりも高速なシミュレーションを達成する一方で，手法ごとに設計し直す必要があり，開発コストが大きい．そこで，深層学習モデルを用いた乱数生成手法の開発方法を示し，データ駆動型の半自動的な乱数生成手法を提案し，新しい乱数生成のあり方とその応用方法を提案する．本研究では具体的に以下の項目を達成する．

1. 深層学習モデルを用いた，データ駆動型で，半自動的な乱数生成手法の設計を行う．
2. 貧弱な乱数性質を持ちながらも，敵対的生成ネットワークへの入力として利用可能な入力値を開発する．
3. 深層学習モデルの学習の遷移と，生成された疑似乱数の性質を NIST 乱数検定によって評価する．

第 8 章

深層学習に関する理論

8.1　深層学習の基礎

深層学習は，脳のニューロンの機能に触発された，人工ニューラルネットワークの一部である．この章では，深層学習の基本的な概念を簡単に説明する．

8.1.1　ニューラルネットワークの基礎

深層学習は，人工ニューラルネットワークの一種である．これらのネットワークは，互いに接続された層の集まりから構成されており，それぞれの層は多数の「ニューロン」または「ノード」を含む．入力層，一つ以上の隠れ層，そして出力層の 3 つの主要な部分がある．ノードは，自身が受け取った入力に基づいて活性化するか非活性化するかを決定する．この決定は，それぞれの入力に対して与えられる「重み」によって大きく影響を受ける．重みは学習中に調整され，訓練データに対するネットワークのパフォーマンスを最適化するために使用される．これらのノードの活性化は，活性化関数によって制御される．活性化関数は，ニューロンが発火するかどうかを決定する閾値を設定する．よく使われる活性化関数には，シグモイド関数，双曲線正接関数，ReLU（Rectified Linear Unit）などがある．

8.1.2　フィードフォワードとバックプロパゲーション

深層学習の主要なメカニズムは，「フィードフォワード」と「バックプロパゲーション」の 2 つのプロセスによって実行される．フィードフォワードは，ネットワークが入力を受け取り，それを出力層まで伝播させるプロセスである．それぞれのノードでは，入力と重みの積の総和を計算し，その結果に活性化関数を適用する．バックプロパゲーションは，ネットワークが予測エラーを計算し，そのエラーをネットワークの各重みに逆伝播させるプロセスである．これにより，重みは次のエポックのために更新され，ネットワークの予測精度が改善される．バックプロパゲーションは勾配降下法の一種であり，誤差関数（または損失関数）の勾配に基づいて重みを更新する．

8.1.3 勾配降下法と学習率

勾配降下法は最適化の手法で，目的関数（通常は損失関数）の最小化を目指す．損失関数の値を最小にするような重みの値を見つけることが目的となる．パラメータ更新は，

$$W = W - \eta \nabla J(W) \tag{8.1}$$

で行われる．ここで，W は重み，η は学習率（ステップサイズ），∇ J(W) は重みに対する損失関数の勾配である．学習率は，重みの更新の際のステップサイズを決定するパラメータである．この値が大きすぎると，最適な解に到達する前に解が振動し，発散してしまう可能性がある．一方，この値が小さすぎると，収束に時間がかかるか，局所的な最小値にトラップされる可能性がある．

8.1.4 過学習と正則化

深層学習モデルが訓練データに対して高いパフォーマンスを達成しても，それが新しいデータに対しても同様のパフォーマンスを達成するわけではない．モデルが訓練データに対して過度に最適化されると，過学習（オーバーフィッティング）という問題が生じる．これを防ぐための一般的な手法が正則化である．正則化は，モデルの複雑さに対してペナルティを与えることで，過学習を防ぐ手法である．重みの大きさにペナルティを与える L1 正則化，重みの二乗にペナルティを与える L2 正則化，両者を組み合わせた Elastic Net などがある．

8.1.5 オートエンコーダ

オートエンコーダは，入力データを低次元の潜在空間にエンコードし，その後デコードして元のデータを再構築するネットワークである．目標は，入力データを効率的に符号化し，その情報を維持しながらデータの次元を削減することである．これにより，データの本質的な特性や構造を抽出し，ノイズや冗長な情報を除去することができる．

オートエンコーダはエンコーダとデコーダの 2 つの部分からなる．エンコーダは入力データ x を潜在空間の表現 z にマッピングする関数である．これは通常，ニューラルネットワークで表され，$z = f_\theta(x)$ と書くことができる．ここで，θ はエンコーダのパラメータである．デコーダは潜在空間の表現 z を再構築されたデータ $\hat{x}$ にマッピングする関数である．これも通常，ニューラルネットワークで表され，$\hat{x} = g_\phi(z)$ と書くことができる．ここで，ϕ はデコーダのパラメータである．オートエンコーダの訓練は，入力データ x と再構築されたデータ $\hat{x}$ との間の差（通常，二乗誤差またはバイナリクロスエントロピー損失）を最小化するように，エンコーダとデコーダのパラメータ θ と ϕ を更新することで行われる．

8.1.6 変分オートエンコーダ

変分オートエンコーダ（VAE）は，生成モデルの一種で，オートエンコーダを拡張したものである．VAE は，エンコーダ部分で入力データを潜在空間の分布のパラメータ（平均と分散）にマッピングし，デコーダ部分で潜在空間からサンプリングした点を元のデータ空間にマッピングする．

具体的には，エンコーダは入力 x を平均 μ とログ分散 $\log\sigma^2$ にマッピングする．この平均と分散は，潜在空間 z の事前分布 $q_\theta(z|x)$ のパラメータ

$$\mu, \log\sigma^2 = \text{Encoder}(x) \tag{8.2}$$

となる．次に，潜在空間からサンプリングする．これは

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{8.3}$$

のように行われる．ここで，$\odot$ は要素ごとの乗算を表し，ϵ は標準正規分布からのランダムなノイズである．

最後に，デコーダはこのサンプリングされた z を入力として受け取り，元のデータ空間にマッピングして x を次のように

$$\hat{x} = \text{Decoder}(z) \tag{8.4}$$

と再構築する．VAE の損失関数は，再構築誤差と KL ダイバージェンスの 2 つの項から成る．再構築誤差は，入力データと再構築データとの間の差を表し，KL ダイバージェンスは，エンコーダによって推定された潜在空間の分布と事前分布（通常は標準正規分布）との間の距離を表す．

8.1.7　深層生成モデルの利用

深層生成モデルは，新しいデータを生成するだけでなく，データの欠損値の補完，データの異常検出，データのクラスタリングなど，様々なタスクに使用することができる．また，深層生成モデルは，人工的なデータセットを生成し，それを使用してモデルを訓練するための手法としても利用できる．

8.2　Generative Adversarial Networks （GAN）

生成対立ネットワーク（GAN）は，Goodfellow らによって最初に提案された深層生成モデルの一種である [34]．GAN は，例えば画像生成やシミュレーション結果の予測など，さまざまな分野で広く使用されている．GAN の学習プロセスはゲーム理論に基づいており，生成器ネットワーク G と識別器ネットワーク D がミニマックスゲームで競い合う．生成器は，識別器を欺くサンプル x を生成するように訓練される．一方，識別器は，トレーニングデータからのサンプルと生成器からのサンプルを正しく識別するように訓練される．GAN が成功裏に訓練されると，生成器は区別できないサンプルを生成する．$D(x)$ を識別器がサンプル x を生成器によって生成された偽のサンプル（$x \sim p_g$：偽のサンプルの確率分布）ではなく，実際のサンプルから生成されたサンプル（$x \sim p_r$：実際のサンプルの確率分布）と判断する確率とし，$V(D, G)$ を GAN の目的関数とし，

$$V(D, G) = \mathbb{E}_{x\sim p_r}[\log D(x)] + \mathbb{E}_{x\sim p_g}[\log(1 - D(x))] \tag{8.5}$$

と表す．GAN の学習は，

$$\min_G, \max_D, V(D, G) \tag{8.6}$$

のミニマックスゲームに従う．

8.2.1 GAN の挑戦と改良

GAN は強力な生成モデルである一方で，いくつかの問題が議論されている．特に，モード崩壊と呼ばれる現象がよく問題となる．これは生成器がデータの多様性を十分に捉えられず，限られた種類のデータしか生成できなくなる現象である．

この問題を解決するために，多くの改良版 GAN が提案されている．次に，ネットワークの損失関数を改良することでモード崩壊を緩和し，安定した学習を可能にした Wasserstein GAN (WGAN) [33]，勾配ペナルティを導入することで WGAN の性能をさらに向上させた WGAN-GP [119]，スペクトラルノルム正則化を導入し，安定した学習と高品質な画像生成を可能にした Spectral Normalization GAN (SNGAN) [120] について述べる．

8.3 Wasserstein Generative Adversarial Network（WGAN）

学習の不安定性，モード崩壊，勾配消失などの問題があるため，伝統的な GAN の代替として，Arjovsky らによって Wasserstein GAN（WGAN）が提案された [33]．WGAN の損失関数は，最適輸送問題から Wasserstein 距離を使用して定義される．W は荷物を 1 つの場所から別の場所へ運ぶためのコストを表す．もし地球が確率分布関数である場合，W は 1 つの分布を別の分布に変換するために必要な最小の作業量を表す．Wasserstein 距離の双対形を使用することで，GAN の目的関数 $V(D,G)$ は次のように

$$V(D,G) = W(p_r, p_g) \ = \max_{\phi} \mathbb{E}_{x\sim p_r}[D(x;\phi)] - \mathbb{E}_{z\sim p_z}[D(G(z;\theta);\phi)] \tag{8.7}$$

と書き直すことができる．ここで $D(x;\phi)$ は Lipschitz 連続性 $||D(x_1) - D(x_2)|| \leq ||x_1 - x_2||$ を満たす関数です．WGAN は一般的に目的関数 $V(D,G)$ を最小化するように訓練される．識別器はパラメータ ϕ を更新して Wasserstein 距離 $W(p_r, p_g)$ をより正確に近似し，生成器はパラメータ θ を更新して $W(p_r, p_g)$ を減らす．

8.4 WGAN with Gradient Penalty（WGAN-GP）

Wasserstein GAN with Gradient Penalty (WGAN-GP) [119] は，元の GAN モデルのモード崩壊と学習の不安定性に対処するために開発された改良版 GAN である．このモデルは Wasserstein GAN (WGAN) という方法を拡張しており，より安定した学習と品質の高い生成結果を実現している．WGAN では，識別器（通常はクリティックと呼ばれる）がリプシッツ連続であることを保証するために，重みのクリッピングが行われる．しかしこの方法は，学習中に識別器の能力を制約し，結果的に生成器の学習に悪影響を及ぼす可能性がある．この問題を解決するために，WGAN-GP では，重みのクリッピングの代わりに勾配ペナルティが導入される．このペナルティは，真のデータと生成データの間のランダムな点におけるクリティックの勾配のノルムが 1 になるように制約を加える．具体的には，WGAN-GP の目的関数は

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))] \\ +\lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]\end{aligned} \tag{8.8}$$

のように表せる．ここで，$\hat{x}$ は真のデータ x と生成データ $G(z)$ の間のランダムな点を表し，λ は勾配ペナルティの重みである．この目的関数を最小化することで，WGAN-GP は生成器が真のデータ分布をより正確に模倣するように学習する．以上のように，WGAN-GP は GAN の学習の安定性を向上させ，より高品質な生成結果を得ることができる．これは，WGAN-GP が生成器と識別器の間の学習のバランスをより効果的に保つことで，両者の学習を同時に進めることを可能にするからである．

8.5 Spectral Normalization

Spectral Normalization [120] は，深層学習における学習の安定性を向上させるための正規化技術である．特に，Generative Adversarial Networks (GAN) の訓練におけるモード崩壊と勾配消失問題に対処するために使用される．

Spectral Normalization は，ニューラルネットワークの重み行列の特異値による正規化技術である．具体的には，Spectral Normalization は各層の重み行列 W の全ての特異値の最大値（スペクトラルノルム）で W を正規化する．このスペクトラルノルムは，W がリプシッツ連続であることを保証し，これにより学習過程全体の安定性が向上する．

Spectral Normalization の方法として，重み行列 W の特異値分解を行い，最大特異値（スペクトラルノルム）$\sigma(W)$ を求める．具体的には，$W = U\Sigma V^T$ と分解される．次に，W をスペクトラルノルム $\sigma(W)$ で除算することで，正規化された重み行列 $\hat{W}$ は，

$$\hat{W} = \frac{W}{\sigma(W)}. \tag{8.9}$$

と求められる．以上のように，Spectral Normalization は深層学習のモデル，特に GAN の学習安定性を向上させるための有効な手法である．本研究では，GAN のより安定した学習と性能を期待し，WGAN-GP と併用して用いた [121].

第 9 章

深層生成モデルを利用した乱数生成手法の開発

9.1 本章の目的と概要

本研究では既存の研究をブラッシュアップするだけでなく，新たなエンドツーエンドの乱数生成モデルを提案する．これを学習型 PRNG（Learned Pseudo Random Number Generator: LPRNG）と呼ぶ．これは，すでに洗練されたバージョンの GAN として知られている，Wasserstein 距離（WGAN [33]）を採用した GAN を用いる．このモデルにより，よく知られた PRNG アルゴリズムである MT 法 [21] から得られる乱数から学習し，NIST 乱数検定の推奨設定，すなわち 1,000 回以上のテスト実行を含む NIST 乱数検定の全テストに合格する実用的な乱数を生成する．NIST 乱数検定を完全に満足させることで，生産環境で展開するのに十分な品質を持つ最初の自動生成 PRNG となる．本研究で用いた WGAN では，学習を通じて同じ教師データを繰り返し利用する必要が無いため，ドロップアウト層を取り除いたモデルを利用した．Fig. 9.1 に，この研究の概念図を示す．本研究では，Generator の潜在空間への入力として，わざと乱数を利用せず，乱数性能が低い三角関数を用いて作成した入力値を採用した．また，これらの入力値が NIST 乱数検定を用いて乱数性が低いことを確認した．NIST 乱数検定を用いた評価によって，WGAN の学習が進むにつれて，入力値をより良い乱数性を持つ乱数列に変換できるようになることを観察した．実験結果は，本研究の深層モデルが学習上，上限のない量のトレーニングデータから「乱数性」を学び取ることができることを示した．興味深いことに，上限のない量のトレーニングデータを用いても，学習を続けることによって過学習が発生することが示された．これは，固定サイズのニューラルネットワークの学習回数の上限が存在することを示している．

9.2 Generator の潜在空間への入力として乱数性質を持たない入力値の開発

9.2.1 Generator への入力として多様性を考慮した初期 seed の開発

本研究の目的は，機械学習，特に GAN を用いて特定の数値列から乱数のような特性を持つ出力を生成することである．GAN を用いて乱数を生成しようとする場合，Generator の潜在空間へ

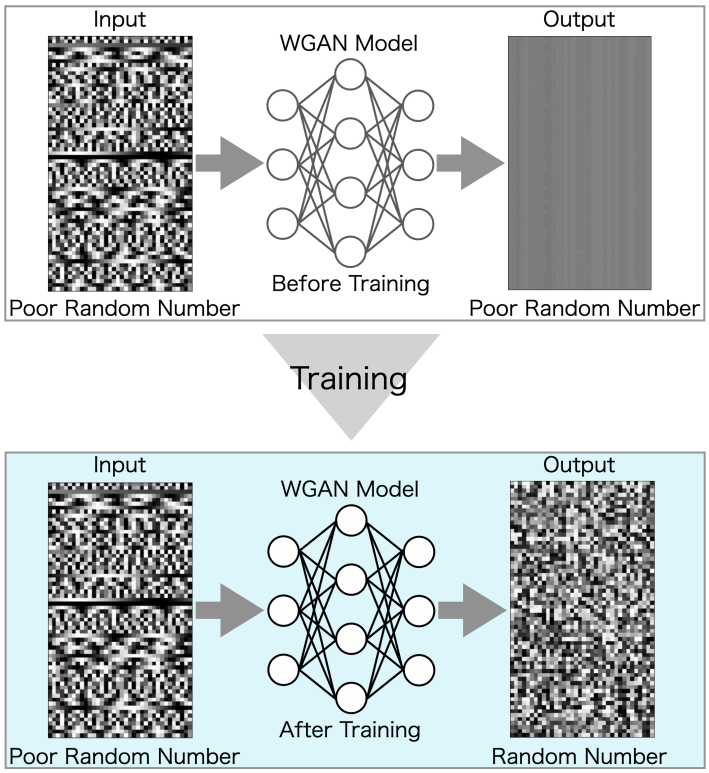


Figure 9.1: 本研究の概念図. 上の画像： 性質の悪い乱数を発生させる学習前の WGAN モデル. 下の画像： 性質の悪い乱数を性質の良い乱数に変換する学習後の WGAN モデル.

の入力値が乱数性質を持っていないことが重要である．潜在空間への入力値として乱数は一般的に使用されるが，本研究において，入力値として乱数を使用すると，GAN が出力を生成するために乱数特性を学習したのか，それとも乱数特性を学習せずに単に入力シードの特性を出力したのかを判断することができない．乱数性質を持たない数列を入力種とし，GAN による変換によってランダム性を持つ出力が得られた場合，GAN はランダム性を学習し，乱数性質を持つ数列に変換できたと考えることができる．そこで，GAN の学習に適した，つまり Generator の潜在空間への入力として十分な多様性を持ちつつも，乱数性質を十分に保たない入力値を考案した．ここでは，規則性や周期性のある数列を悪い乱数列と考え，三角関数用いて生成した数値列を考案した．本研究で用いる入力値は，36 個の 32 ビット浮動小数点数で構成される．各入力値は，次の式に従って生成される．

$$\text{input}_i = \cos(2R_1\pi i + 2R_2\pi) \times \frac{1}{2} + \frac{1}{2}, \quad i = 0, 1, \ldots, 35 \tag{9.1}$$

ここで，i は整数のインデックス，R_1 と R_2 は $[0, 1]$ の範囲の一様乱数であり，各入力シーケンスに固有の値とする．この方法により，数値列の類似性を削減しながらも，乱数列とは異なる特性を持つ入力値を生成する．そして，乱数を用いることで，バッチごとに生成する入力値を独立同分布の入力値として生成可能である．Generator の潜在空間への入力値として，独立同分布の性質を満たしながら与えた理由として，3 つの理由がある．1 つ目に，GAN は学習のため，入力値に多様性のあるデータを要求するからである．2 つ目に，GAN の学習のため，入力するすべてのデータは互いに独立性を満たしていることが，学習に有効であるからである．また学習においても人為的な選択が介入しない上でも重要である．3 つ目に，Generator により出力された数値列

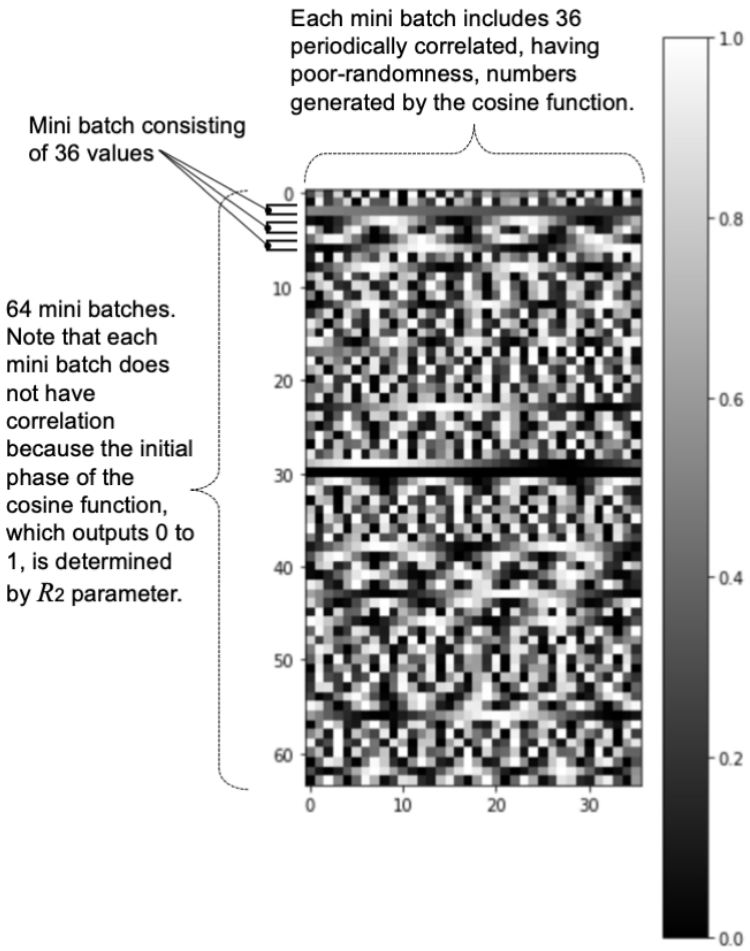


Figure 9.2: Eq.(9.1) で示す cos 関数によって生成された数列．横軸は 1 つの入力シードを表す貧弱な乱数列を表し，縦軸は 64 バッチそれぞれのデータを表す．

をまとめて NIST 乱数検定を実行するため，事前に異なるバッチ間の相関を除去するためである．異なるバッチへの操作は GAN の性能外である．したがって，二つの乱数を利用して，バッチごとに周期，初期位相が異なるものを入力として用いた．cos 関数によって生成された 64 の異なる周期を持つ数列を Fig. 9.3 に示す．水平方向に周期性があることが図からも確認できる．この数値列を表 9.1 の NIST 乱数検定で評価した結果，この数値列は十分な乱数性質を持たないことがわかった．この数列を機械学習の初期入力シードとして使用した．これは従来の研究とは異なるアプローチである．

9.2.2 Generator の潜在空間へのへの入力として多様性を欠いた入力値の利用とその影響

従来の GAN では，Generator への入力として乱数を使用するのが一般的であるが，本研究では異なる方法を試みた．具体的には，三角関数を用いて生成したバッチごとに独立同分布（i.i.d）の数値列を入力として使用し，その効果を検証したが，乱数と比較して GAN の学習に十分な多様性を持つのかどうか比較するために，自由度の異なる入力を用意し，同様に実験を行った．

Table 9.1: 入力シードに対する NIST 乱数検定の結果.

Statistical test	**Pass rate**	**Uniformity of p-values**	**Pass/fail judgment**
Frequency	0.000	Fail	Fail
BlockFrequency	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
Runs	0.000	Fail	Fail
LongestRun	0.000	Fail	Fail
Rank	0.995	Pass	Pass
FFT	0.983	Pass	Pass
NonOverlappingTemplate	0.784	36/148 Pass	Fail
OverlappingTemplate	0.000	Fail	Fail
Universal	0.541	Fail	Fail
ApproximateEntropy	0.000	Fail	Fail
RandomExcursions	0.000	0/8 Pass	Fail
RandomExcursionsVariant	0.000	0/18 Pass	Fail
Serial	0.000	Fail	Fail
Serial	0.800	Fail	Fail
LinearComplexity	0.992	Pass	Pass

この入力の特徴は，バッチ内の 36 個の数値が三角関数によって生成されるため，一つのバッチ内で周期性が観察できる点にある．しかし，異なるバッチ間では三角関数の周期と初期位相がそれぞれ乱数で異なるものとし，独立同分布（i.i.d）の性質を持たせた．これは，上述したように多様な数値列を作成する目的と，WGAN が異なるバッチ間で i.i.d の性質を与える機能を持たないため，事前に異なるバッチに対して i.i.d の性質を付与する必要があったためである．三角関数を用いた数式において，独立で選べる変数の数を自由度と呼ぶと，周期と初期位相の決定に乱数を 2 つ用いた入力値の自由度は 2 である．そこで自由度 1 の場合を 2 通り検討し，実験に用いた．各入力値は，次の式に従って生成される．

$$\mathrm{input}_{\mathrm{phase-fixed}_i} = \cos(2R_1\pi i + \frac{\pi}{4}) \times \frac{1}{2} + \frac{1}{2}, \quad i = 0, 1, \ldots, 35 \tag{9.2}$$

$$\mathrm{input}_{\mathrm{period-fixed}_i} = \cos(\frac{\pi}{4} i + 2R_2\pi) \times \frac{1}{2} + \frac{1}{2}, \quad i = 0, 1, \ldots, 35 \tag{9.3}$$

ここで，R_1 と R_2 はそれぞれバッチごとの周期と初期位相に対応する乱数である．これらは 2 通りとも，異なるバッチ間で独立同分布（i.i.d）の性質を持ちつつ，自由度が 1 の入力値である．これらの条件下で，それぞれの入力を用いて WGAN の学習を行い，生成された数列の乱数性を NIST 乱数検定を用いて評価した．特に，周期と初期位相が乱数で決定された自由度 2 の場合と固定された自由度 1 場合の比較から，GAN の学習過程において入力の自由度がどのように作用するかを解析した．

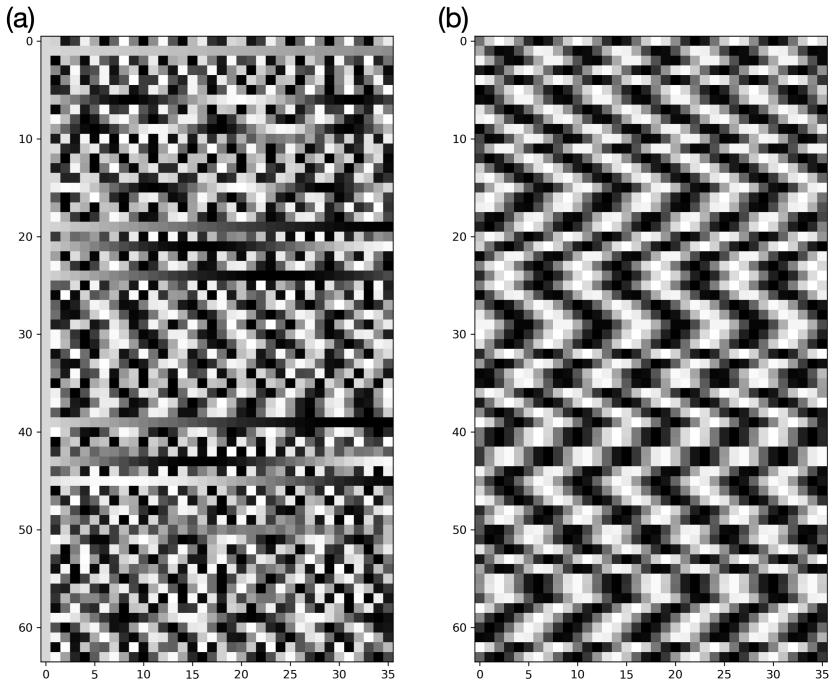


Figure 9.3: (a)Eq.(9.2) で示す cos 関数によって生成された数列．横軸は 1 つの入力シードを表す貧弱な乱数列を表し，縦軸は 64 バッチそれぞれのデータを表す．(b)Eq.(9.3) で示す cos 関数によって生成された数列．横軸は 1 つの入力シードを表す貧弱な乱数列を表し，縦軸は 64 バッチそれぞれのデータを表す．

9.3　深層生成モデルによる乱数生成とその評価

9.3.1　深層モデルの訓練結果

本研究では WGAN を使用して，MT によって生成された乱数の構造を予測するための機械学習モデルを構築した [33,34]．訓練データとして MT の乱数を使用し，WGAN をモデルとして乱数を出力するために WGAN を訓練した．WGAN は以下の特徴を持つ：(1) 訓練データに類似したデータを生成する． (2) 訓練データの特徴は潜在変数空間で獲得する． (3) 訓練は，類似したデータが潜在変数空間で近くに分布するように進行する．さまざまなランダムな画像で訓練された WGAN は，多くの種類のランダムな構造を持つ画像を生成することが期待される．

WGAN のアーキテクチャを Fig. 9.4 に示す．各層では，Leaky ReLU 関数が入力層と隠れ層の活性化関数として使用される．生成器の出力は，シグモイド関数を使用して範囲 (0,1) にスケーリングされる．また，訓練プロセスの概略図は Fig. 9.5 に示される．質の低いランダムな数値である余弦関数（Eq.(9.1)）によって生成された入力シードを，生成器の入力として使用した．入力シードは 36 次元で，64 バッチを使用した．この入力シードは，NIST 乱数検定の結果においてランダム性が低いものである．その後，識別器は生成された数値がランダムな数値か質の低いランダムな数値かを判断した．この研究では，訓練データとして MT の乱数を使用した．潜在変数の次元は 36 とし，各潜在変数を 0 から 1 の範囲で生成される浮動小数点数として定義した．識別器は，生成器の訓練ごとに 5 回訓練した．Fig. 9.8 に示されているように，識別器のパラメータを 5 回更新した後に生成器のパラメータを 1 回更新することを 1 イテレーションと定義した．識別器

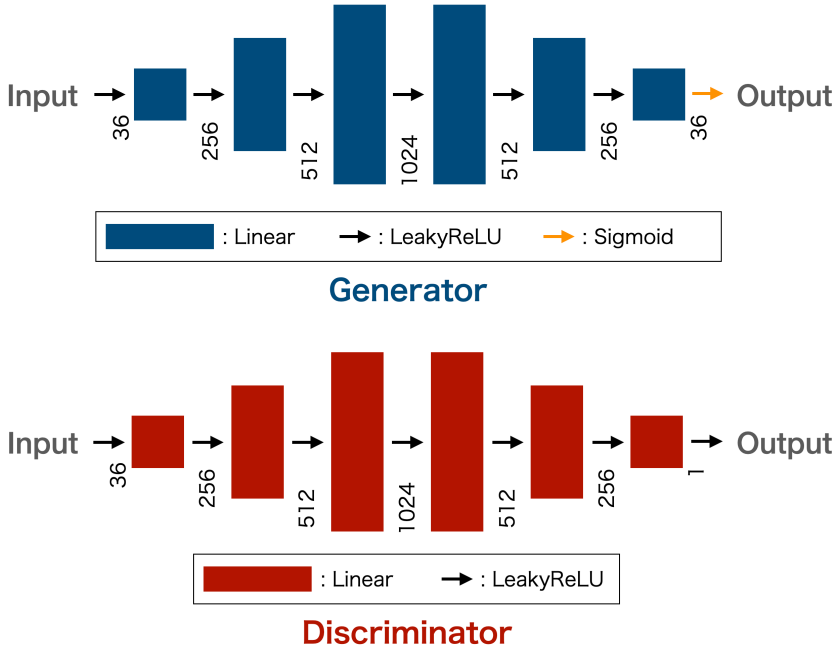


Figure 9.4: WGAN のアーキテクチャ

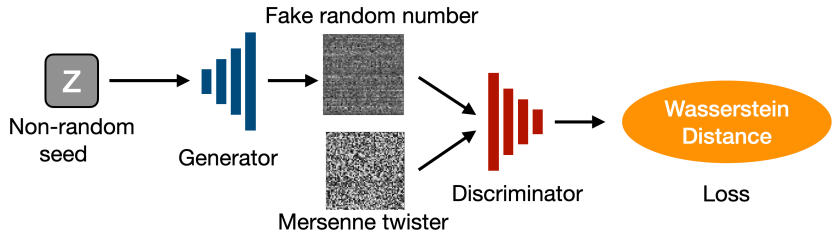


Figure 9.5: 訓練プロセスの概略図

と生成器の最適化には RMSprop オプティマイザを使用し，識別器のすべての線形層にはスペクトル正規化 [120] が実装と，また，勾配ペナルティ [119] を適用した．

ドロップアウト層の削除

本研究では，機械学習による乱数生成のチューニングにより，ドロップアウト層を排除した WGAN モデルを用いた．もともとドロップアウト層を持つモデルを使用していたが，ドロップアウト層を除去することで生成される乱数の質が著しく向上した．一般に，ニューラルネットワークの構造が複雑になるにつれて，ニューロンの重みは学習データセットに対して最適化されるようになる．そのままでは，モデルは汎化性能に欠け，貧弱になり，データを一つ一つ記憶するようにしか訓練データセットに使用できなくなる．学習が進むにつれて，訓練データの正答率は徐々に上がっていくが，テストデータの誤差は減少しなくなり，再び増加し始める．このような状況をオーバーフィッティングと呼ぶ．この現象を防ぐためにドロップアウト層が使われる．本研究の場合，学習データとして無限の乱数を用意した．さらに，ドロップアウト層を取り除くことで学習プロセスを加速させた．

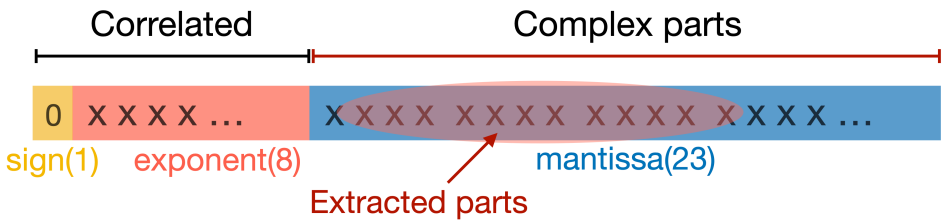


Figure 9.6: Bit window extraction.
仮数部からは，MT が生成した乱数の場合に十分なランダム性を持つことが判明したビットをいくつか抽出した．

Bit window extraction

バイナリデータは，NIST 乱数検定を使って評価するために準備しなければならない．一方，機械学習では 10 進浮動小数点表現が一般的である．本研究では，浮動小数点表現から 2 進数データを抽出した，符号部，指数部，仮数部から構成される．0 と 1 の間の乱数を考えた場合，符号部と指数部はほとんど変化しない．これは，ビット列のランダム性が低いことを意味します．一方，仮数部の一部にはランダム性が含まれている可能性がある．このランダム性を利用するため，仮数部から，MT の場合の NIST 乱数検定の結果で十分なランダム性を持つことが判明したビットを，Fig. 9.6 に示すようにいくつか抽出した．NIST 乱数検定の結果は，生成された乱数の仮数部からこれらの数ビットを抽出したものである．

学習の進行過程

Fig. 9.7 に損失関数の推移を示す．学習の初期段階（約 25 万反復）では，学習が進むにつれて損失が減少していることがわかる．しかし，学習の後半になると，損失は再び増加し始める．Fig. 9.8(a) に各学習ステップで出力された乱数の NIST 乱数検定の結果を示す．各ステップで生成された乱数を複数個連結し，3000 回反復した後に NIST 乱数検定を行った．学習過程を W，X，Y，Z の 4 つの領域に分けたところ，領域 W では学習が進むにつれて NIST 乱数検定の点数が向上した．

領域 X では，NIST 乱数検定の点数は 11 前後で変動し，Frequency，CumulativeSums，Runs テストは常に不合格，NonOverlappingTemplate テストはしばしば不合格，OverlappingTemplate，Universal， ApproximateEntropy，Serial テストはほとんど不合格であった．領域 Y では，生成された乱数は最終的に NIST 乱数検定に完全に合格したが，Frequency，CumulativeSums，Runs，NonOverlappingTemplate テストにはしばしば不合格であった．用いたモデルでは Frequency と CumulativeSums のテストに合格するのは困難であった．一度良い乱数性質を獲得した後，領域 Z では，NIST 乱数検定の点数が低下し，過学習に似た現象を確認した．この領域では，Rank，LinearComplexity，FFT のテストには高い確率で合格していることが確認できた． 学習の結果，全体を通して，MT 乱数を学習するだけで，NIST 乱数検定の 15 項目の統計テストに合格することが可能になる．表 9.2 と 9.3 にそれぞれ，学習前と学習後に出力された乱数のうち満点が得られたもの（Fig. 9.8 の A と C のデータ）の NIST 乱数検定のスコアを示す．Fig. 9.8(b) の画像 A,B,C では，乱数の成長が見られる．なお，GAN は連続学習中に NIST 乱数検定の全ての

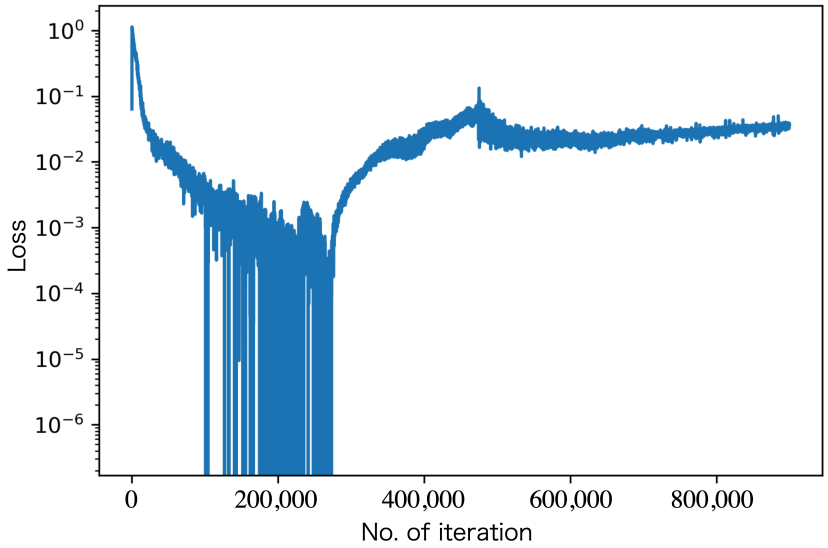


Figure 9.7: 反復回数の関数としての損失関数.

テストに安定して合格することは困難であった．しかし，36 個の float 型ビットストリームからなる出力を複数連結して生成した乱数が NIST 乱数検定に合格していることから，少量の乱数出力（ただしバッチ数は多い）を学習したモデルでも良好な乱数を生成できることがわかった．Fig. 9.9 は Dropout 層を追加した場合の結果である．見ての通り，ドロップアウト層を取り除いた方が学習結果が良くなっている．

この領域では，Rank，FFT，RandomExcursions，RandomExcursionsVariant，LinearComplexity のテストにのみ合格し，RandomExcursions と RandomExcursionsVariant のテストは不合格となる．学習を通して，Rank，FFT，LinearComplexity のテストは簡単にパスできるが，Frequency と CumulativeSums のテストはパスできない場合が多かった．Fig. 9.8(b) も長期学習 (D) 後のランダム画像がランダムでないことを示しており，NIST 乱数検定の点数は表 9.4 の通りである．この機械学習では，無限のランダムなデータセットを用いているにもかかわらず，オーバーフィッティングが発生する．オーバーフィッティングは一般に有限データセットを用いた場合に起こるが，この結果は無限データセットでも起こりうることを示している．したがって，学習の早期終了は，無限データセットを使用した場合でも有効であると考える．　下記では，入力シードのランダム特性が良い場合と悪い場合の 2 つのケースについて議論した．

1. 良質な乱数性質を持つ入力シードを使うとどうなるか．この場合，入力シードは一般に広く使われているものと同じになり，機械学習に適している．その結果，出力も NIST 乱数検定をクリアする乱数が得られると考えられる．しかし，これらの乱数が三角関数を用いた入力シードを用いて生成された乱数よりも優れているかどうかを判断することは難しい．今回の研究では乱数性質の評価のために NIST 乱数検定を用いており，満点の範囲を超えて 2 つの出力の品質を比較することができないからである．したがって，この場合では，機械学習モデルが乱数の性質を学習せず，入力シードの性質をそのまま出力するように学習する可能性があり，機械学習モデルがランダム性の性質をうまく模倣できるどうかの判断は困難である．
2. 乱数としての性質が悪い入力種を使うとどうなるか．今回は三角関数を用いて入力シード

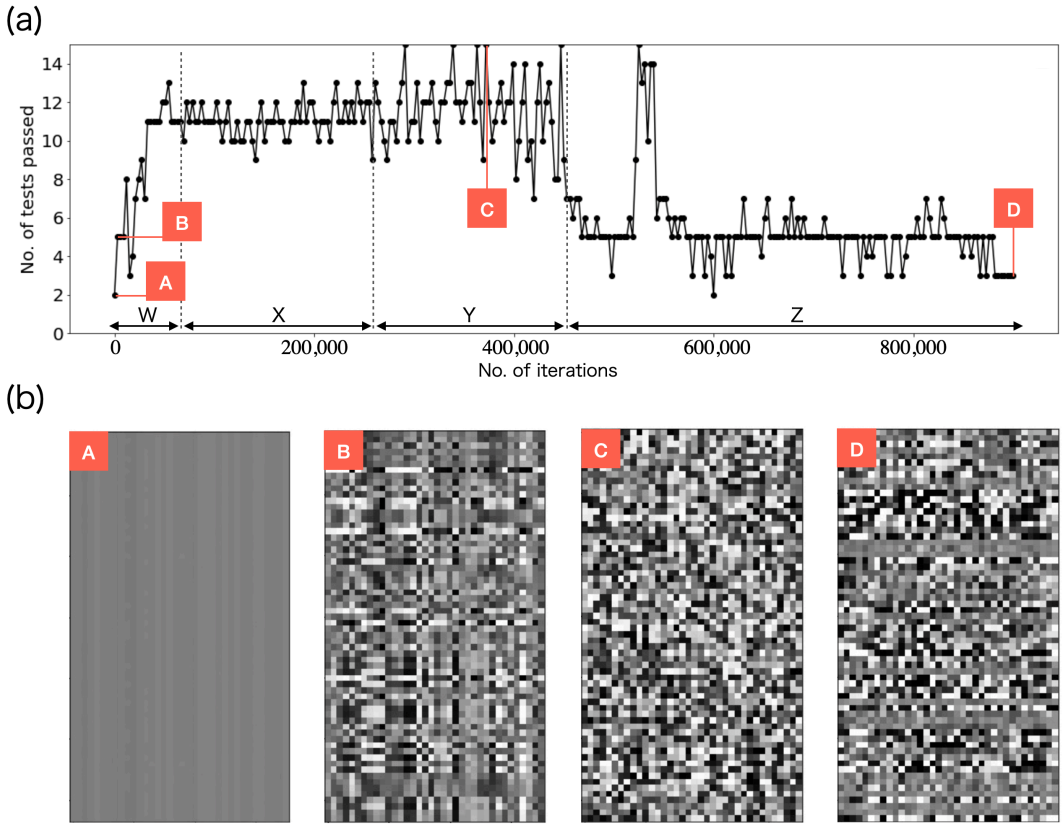


Figure 9.8: NIST 乱数検定の試験結果と，繰り返し回数を変えて生成した乱数の画像．乱数の NIST 乱数検定の点数は A< B < C の順に向上している．乱数の性能は C から D に低下しており，無限の学習データを使用しているにもかかわらず，過学習と同様の現象が発生していることがわかる．識別器のパラメータを 5 回更新した後に発生器のパラメータを 1 回更新することを 1 反復とした．

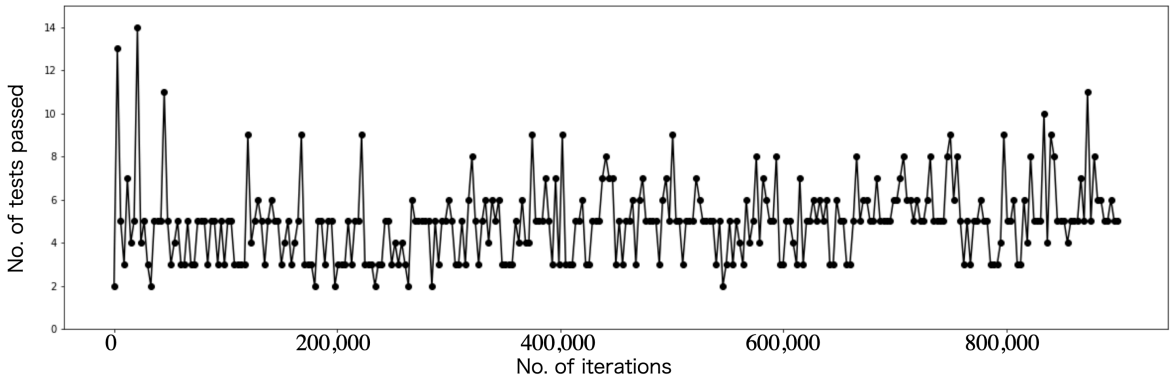


Figure 9.9: WGAN がドロップアウト層で学習した場合の NIST 乱数検定の結果

を開発したが，これよりも悪い性質を持つ入力シードを作成する方法はいくつかある．例えば，1 が常に連続して現れるような数列を検討する．この場合，入力種子の値が制限され，学習が非常に困難になり，結果として出力として良好な性質を持つ乱数を得ることが困難になる．本研究の貢献の一つとして，機械学習過程に大きな影響を与えないランダムな性質を持たない数値列を作成する方法を開発したことを強調できる．

Table 9.2: NIST statistical test results for data A in Fig. 9.8 (before learning).

Statistical test	**Pass rate**	**Uniformity of p-values**	**Pass/fail judgment**
Frequency	0.000	Fail	Fail
BlockFrequency	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
Runs	0.000	Fail	Fail
LongestRun	0.000	Fail	Fail
Rank	0.990	Pass	Pass
FFT	0.000	Fail	Fail
NonOverlappingTemplate	0.113	0/148 Pass	Fail
OverlappingTemplate	0.000	Fail	Fail
Universal	0.000	Fail	Fail
ApproximateEntropy	0.000	Fail	Fail
RandomExcursions	0.000	0/8 Pass	Fail
RandomExcursionsVariant	0.000	0/18 Pass	Fail
Serial	0.000	Fail	Fail
Serial	0.000	Fail	Fail
LinearComplexity	0.987	Pass	Pass

9.3.2　Generator の潜在空間へ与える初期値の違いによる学習の推移

本研究では Generator への入力として乱数を利用しなかった．三角関数を用いて作成した数値列を利用したが，GAN は異なるバッチから得られた出力に対して操作することができないので，事前に入力するバッチごとに独立同分布の性質を付与することが重要であった．ここでは，三角関数を用いて生成された数値列のうち，自由度が 1 の「周期を固定・初期位相を乱数」及び「周期を乱数・初期位相を固定」とする条件の下での学習結果を示す．これらの入力値は共にバッチごとに独立同分布の性質を持つにも関わらず，Fig. 9.10,9.11 に示す通り，そのどちらにおいても，GAN の出力結果は NIST 乱数検定を合格する出力は観察できなかった．これは GAN の学習過程において十分多様性のある入力値を与えられなかった可能性が考えられる．

t-SNE を利用した入力値の多様性の定量的評価

Generator の潜在空間への入力値において，多様性は重要であるが，高次元空間に分布する数値列の多様性を直接定量的に計測する手法は普及していない．ここでは，自由度 2 の入力の多様性と自由度 1 の入力の多様性が一般的に用いられる乱数の多様性と比較してどのように異なるのかを定量的に計測する手法を考案した．まず，下記に示す 5 種類の入力が 36 次元にどのように存在しているかを t 分布型確率的近傍埋め込み法 (t-distributed Stochastic Neighbor Embedding: t-SNE) [122] を利用して 2 次元空間へ可視化した（Fig 9.10）．

Table 9.3: NIST statistical test results for data C in Fig. 9.8 (successful area).

Statistical test	**Pass rate**	**Uniformity of p-values**	**Pass/fail judgment**
Frequency	0.992	Pass	Pass
BlockFrequency	0.988	Pass	Pass
CumulativeSums	0.992	Pass	Pass
CumulativeSums	0.995	Pass	Pass
Runs	0.991	Pass	Pass
LongestRun	0.988	Pass	Pass
Rank	0.982	Pass	Pass
FFT	0.989	Pass	Pass
NonOverlappingTemplate	0.990	148/148 Pass	Pass
OverlappingTemplate	0.989	Pass	Pass
Universal	0.991	Pass	Pass
ApproximateEntropy	0.987	Pass	Pass
RandomExcursions	0.990	8/8 Pass	Pass
RandomExcursionsVariant	0.990	18/18 Pass	Pass
Serial	0.988	Pass	Pass
Serial	0.987	Pass	Pass
LinearComplexity	0.991	Pass	Pass

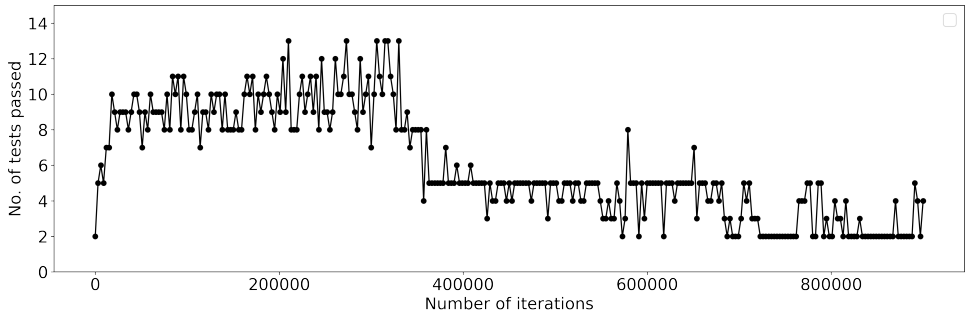


Figure 9.10: Generator の潜在空間へ与える初期値: 初期位相を固定，周期を乱数による決定にした場合

1. MT 法を用いて区間 [0,1] に生成した 36 個の数値列を 2000 サンプル
2. 周期，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列を 2000 サンプル (Eq.(9.1))
3. 周期を乱数で決定，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列を 2000 サンプル (Eq.(9.2))
4. 周期を固定，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列を 2000 サンプル (Eq.(9.3))

Table 9.4: NIST statistical test results for data D in Fig. 9.8 (over fitting area).

Statistical test	**Pass rate**	**Uniformity of p-values**	**Pass/fail judgment**
Frequency	0.000	Fail	Fail
BlockFrequency	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
CumulativeSums	0.000	Fail	Fail
Runs	0.000	Fail	Fail
LongestRun	0.000	Fail	Fail
Rank	0.992	Pass	Pass
FFT	0.984	Pass	Pass
NonOverlappingTemplate	0.900	75/148 Pass	Fail
OverlappingTemplate	0.000	Fail	Fail
Universal	0.752	Fail	Fail
ApproximateEntropy	0.000	Fail	Fail
RandomExcursions	1.000	0/8 Pass	Fail
RandomExcursionsVariant	1.000	0/18 Pass	Fail
Serial	0.370	Fail	Fail
Serial	0.989	Fail	Fail
LinearComplexity	0.991	Pass	Pass

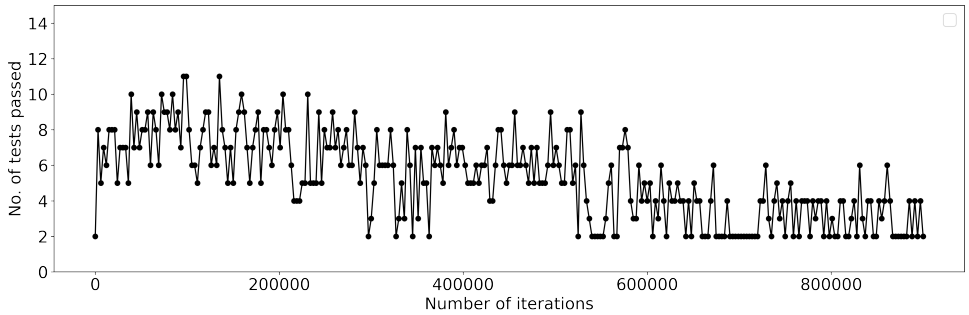


Figure 9.11: Generator の潜在空間へ与える初期値: 初期位相を乱数による決定，周期を固定にした場合

5. 周期，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列を 2000 サンプル

SNE は高次元データに対して，各データが持つ特徴，偏りを低次元空間へプロットし，データの種類ごとに分離して可視化する方法である．高次元空間におけるデータポイント x_i と x_j の類似度は

$$p_{j|i} = \frac{\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma_i^2)} \tag{9.4}$$

と表される．ここで，σ_i はデータポイント x_i における正規分布の分散である．

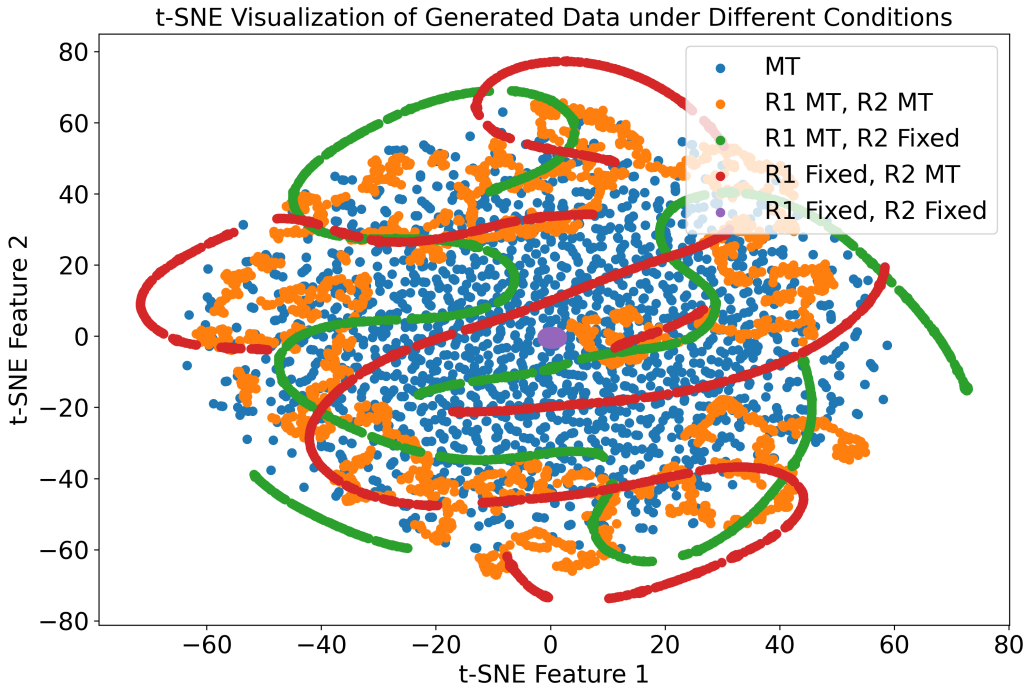


Figure 9.12: t-SNE を用いた各種入力値の可視化

一方，低次元空間でのデータポイント y_i と y_j の類似度は

$$q_{j|i} = \frac{(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_k||^2)^{-1}} \tag{9.5}$$

と表される．SNE の目的は，高次元空間の類似度 $p_{j|i}$ と低次元空間の類似度 $q_{j|i}$ の差異を最小化する低次元表現 $\boldsymbol{y}_i$ を見つけることであり，一般に Kullback-Leibler ダイバージェンスを使用して最適化される．具体的には，損失関数

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{9.6}$$

の最小化を目指すものである．ここで，P_i と Q_i はそれぞれ，x_i と y_i に対応する確率分布を表す．また，SNE において，高次元空間における各データポイント周辺の正規分布の分散を決定するパラメータ，Perplexity が定義され，

$$\text{Perplexity} = 2^{H(P_i)} \tag{9.7}$$

と表される．これは局所的なデータの密度を調整する役割を担う．ここで，$H(P_i)$ はデータポイント x_i に関連する条件付き確率分布 P_i のエントロピーである．エントロピーは

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \tag{9.8}$$

と表され，$p_{j|i}$ は高次元空間におけるデータポイント x_i と x_j 間の条件付き確率である．

ここで，t-SNE は対称なコスト関数を導入する．SNE が条件付き確率分布 $p_{j|i}$ と $q_{j|i}$ の KL ダイバージェンスの合計を最小化するのに対し，t-SNE では同時確率分布 p_{ij} と q_{ij} の KL ダイバー

ジェンスの合計を

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{9.9}$$

として最小化する．これにより，点間の「近さ」が対称に扱われる．しかし，同時確率分布を直接使用する場合，元の空間での外れ値の影響で p_{ij} が非常に小さくなり，適切なマッピングを得ることが困難になる場合がある．そのため，元の空間の「近さ」は

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{9.10}$$

と再定義される．圧縮後の空間での「近さ」は，

$$q_{ij} = \frac{\exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)}{\sum_{k,l \neq k} \exp(-\|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)} \tag{9.11}$$

のように同時確率分布が適用される．また，t-SNE では圧縮後の確率分布の計算に自由度 1 の Student-t 分布を採用している．

特徴のもたない乱数などを t-SNE で分析すると分離できず，一様に広がりのあるプロットとなる．一方で特徴的であればあるほど，局所的に点が凝縮した点群のプロットとなる．Fig 9.10 から分かる通り，乱数で与えた入力値に対しては t-SNE により一様に分布した点群で可視化されたのに対して，その他の入力値はの偏りのある点群で可視化された．t-SNE では多様性を評価するための定量的な方法は定義されていない．多様性とは深層学習の分野において，潜在空間への入力値として，どれだけ高次元空間に一様に分布しているかを表すために用いられるが，厳密な定義はされていない．そこで，入力値の多様性を定量的に評価するため，t-SNE が低次元空間へプロットした点群に対して，局所的な埋め込み密度 (Local Embedding Density:LED) を次に示すように定義した．

$$LED = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{1}{\left(\|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon\right)^2} \tag{9.12}$$

$$\log(LED) = \log \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{1}{\left(\|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon\right)^2} \right) \tag{9.13}$$

ここで，$\mathbf{x}_i$ と $\mathbf{x}_j$ は低次元空間における点の座標を表し，$\|\mathbf{x}_i - \mathbf{x}_j\|$ はこれらの点間のユークリッド距離を表す．また，ϵ は 0 除算を避けるための小さな正の定数である．定義した LED では，高次元空間に偏りがなく，一様に分布すればするほど LED が小さく偏りがあり，t-SNE によってクラスタリングされるほど LED が大きくなるように設計した．t-SNE は高次元空間に分布するデータを低次元に可視化する手法であり，定量的な解析に利用するものではない．本研究では，多様性を数値化する上で参考になる可能性を考え，定義した．t-SNE を利用する上で，$perplexity$ のパラメータの設定が重要である．本研究では推奨値されているパラメータセットを利用した．さらに乱数は高次元空間にも一様に分布しているという点と，周期，初期位相が固定の自由度 0 の数値列は高次元空間に大きく偏りのあるデータであるという点，特にこの場合では 36 次元空間で 1 点しか表現できない点を考慮して，それらがうまく再現できていることを確認し，$perplexity = 20$ とした．また，LED を比較するため，本研究では自然対数を用いた Eq.(9.13)

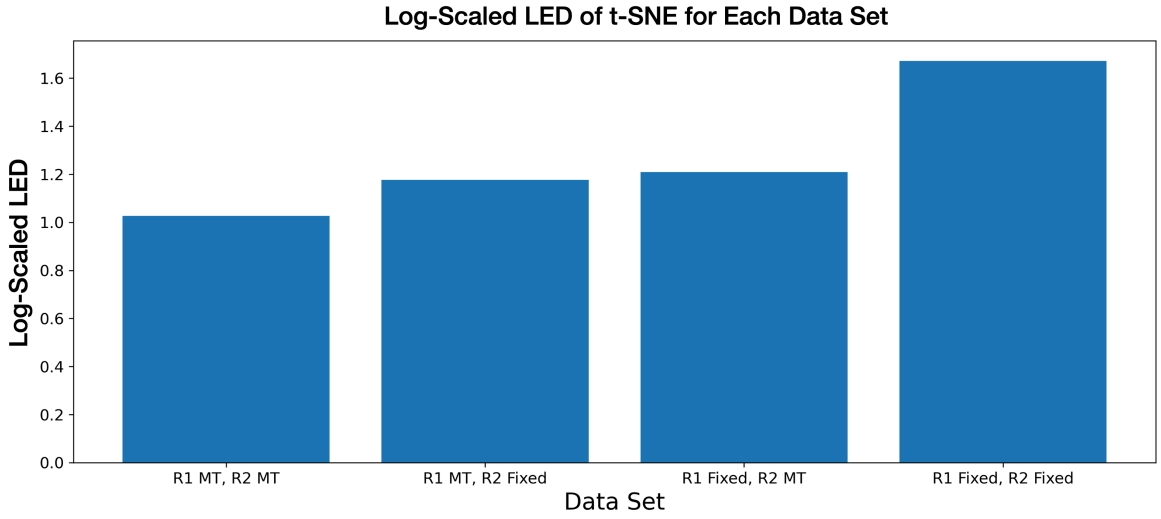


Figure 9.13: 各種入力値から計算した LED の比較

で比較した．Fig9.12 で可視化した各データに対する LED を計算し，MT の LED が 1 となるように正規化し，グラフに描画したものを Fig9.13 に示す．

導出した LED は，「MT 法を用いて区間 [0,1] に生成した 36 個の数値列」が最も小さく，「周期，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列」,「周期を乱数で決定，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列」,「周期を固定，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列」,「周期，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列」の順に大きくなるという結果が得られた．これは GAN の学習によって得られた最終的な NIST 乱数検定の点数の順序と一致している．本研究の結果，NIST 乱数検定の結果は「周期，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列」,「周期を固定，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列」,「周期を乱数で決定，初期位相を固定し，区間 [0,1] に生成した 36 個の数値列」,「周期，初期位相を乱数で決定し，区間 [0,1] に生成した 36 個の数値列」の順に到達した最高点数が高くなった．ここから，LED が低いほど，高次元空間に一様に分布し，多様性を持つ可能性があり，GAN の学習において有利に機能したことが示唆された．

次に，自由度が異なると LED はどのように変化するのかを調べた．そのために周期のみを乱数で決定した三角関数を作成し，それをいくつ足し合わせるかによって自由度を変更する数値列を評価した．

$$input_{i,s} = \frac{1}{\text{degree}} \sum_{d=1}^{\text{degree}} \left(\frac{1}{2} \cos(2R_{d,s}\pi i) + \frac{1}{2} \right), \quad i = 0, 1, \ldots, 35, s = 1, 2, \ldots, 2000 \quad (9.14)$$

ここで，degree は 36 個の数値を出力する上で用いた乱数の個数，つまり独立した変数の個数であり自由度を表し，degree が大きいほど自由度の大きい数値列を出力したことになる．また，i の値が 0 から 35 を取ることにより 36 個の数値を出力するが，36 個の数値を出力するたびに新しく乱数を決め直し，新たな数値列を作ることとした．s は 36 個の数値を 1 サンプルとした場合のサンプル数を表し，本研究では 2000 サンプルを対象に LED を計算した．MT 法を用いて生成した乱数と，degree$= 1, 2, 3 \ldots 8$ のそれぞれで 36×2000 サンプルの数値列を作成し，先述した t-SNE を利用して 2 次元にプロットした後，LED を計算した．Fig. 9.14 にその結果を示す．ま

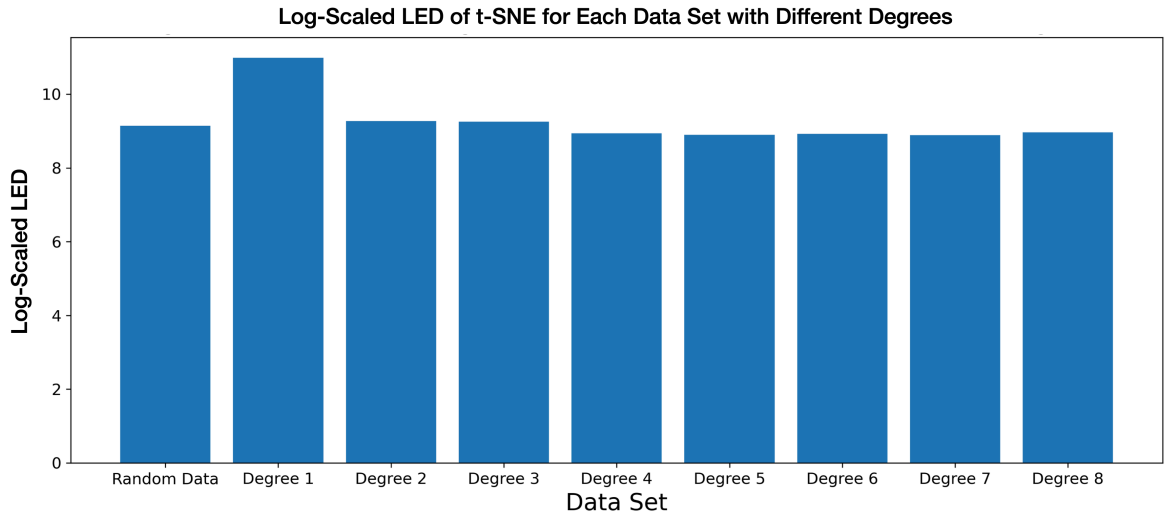


Figure 9.14: t-SNE を用いた 2 次元プロットから計算した，自由度の異なる入力値に対する LED

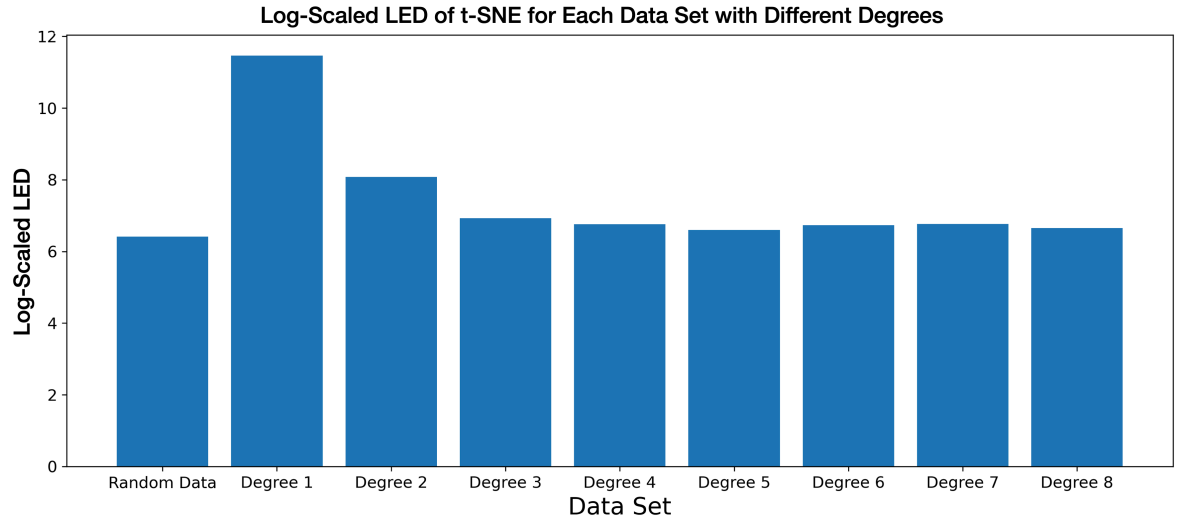


Figure 9.15: t-SNE を用いた 3 次元プロットから計算した，自由度の異なる入力値に対する LED

た，t-SNE を利用して 3 次元にプロットした後，LED を計算した値を Fig. 9.15 に示す．この結果から，それぞれの LED の大きさの順位を定量的に確認でいたことがわかる．自由度 1 のみが 10 を超える大きな LED を有し，その他の自由度は自由度 1 に比べて乱数との LED 差が比較的小さいことがわかった．自由度 1 の場合，乱数との LED 差が比較的大きいことがわかった．これはつまり，深層学習において潜在空間へ与える入力値として，自由度 2 以上の入力は自由度 1 に比べて乱数と近い LED を得られたことから，LED の比較検証においては同様の多様性を担保できる可能性を示唆している．

自由度が 2 の周期と初期位相を乱数にした場合，NIST 乱数検定を合格する乱数性質を持つように学習が進行した一方で，自由度が 1 の「周期を乱数・初期位相を固定」及び「周期を固定・初期位相を乱数」は共に NIST 乱数検定を合格する数値列を出力することはできなかった．

9.3.3 結論と今後の展望

結論

本研究では，貧弱な乱数種を持つ WGAN を用いて乱数を生成することに成功した．得られた乱数を NIST 乱数検定により評価したところ，全てのテストに合格した．また，学習上，上限のない量の学習データを用いたにもかかわらず，過学習と似た現象が観測された．学習を通じて，用いたモデルは Rank，FFT，LinearComplexity テストに容易に合格することがわかったが，Frequency と CumulativeSums テストには合格しなかった．深層学習の内部の挙動はまだ完全には明らかにされていないが，本研究では，合格したテストを調べることで，生成器がどのような順に乱数性質を学習するかを明らかにした．乱数検定などの統計的検定による学習の評価は，深層学習の理解に新たな知見をもたらす可能性がある．また，この研究は様々な応用が可能である．1. 任意の分布に従う乱数を学習の学習データとすることで，任意の分布に従う乱数生成法を設計できる可能性がある．2. 計算効率を考慮した機械学習の訓練により，計算効率の良い高速な乱数生成法の開発につながる．3. 機械学習を用いた乱数検定法は先行研究で検討されている [123]．GAN の学習で用いられる識別器は，乱数と貧弱な乱数を区別するため，乱数テストとしても利用できる．また，NIST 乱数検定と異なり，入力データ量に制限がないため，任意の入力データサイズの乱数テストに識別器を用いることができる．この結果は，機械学習の利用や今後の乱数生成手法の開発に有用な情報を提供する．

本研究の応用可能性と今後の展望

エンドツーエンドの深層学習のみを用いた本提案手法は，乱数生成器の開発へ誰もが取り組み，個別の目的に特化したアルゴリズムを開発できるようになること，つまり，一般の技術者がニーズに応じた独自の PRNG を開発することへ貢献した．提案手法では，従来必須とされてきた深い数学的知識や経験がなくても乱数生成アルゴリズムを開発できるため，使い捨ての新しい乱数生成アルゴリズムを簡単に作ることができる．つまり，一般に未知で予測不可能な新しい乱数生成アルゴリズムを安心して利用することができる．さらに，LPRNG で使用されるニューラルネットワークは，可搬性とリバースエンジニアリングに対する堅牢性を兼ね備えている．ここで，移植性とは，コンピュータアーキテクチャに依存せず，どのようなコンピュータでも操作可能であることを意味し，リバースエンジニアリングに対する頑健性とは，モデル内のアルゴリズムが人間によって解釈されにくいことを意味する．DNN は内部の挙動が明らかにされていない状態（ブラックボックス）で稼働する．LPRNG はブラックボックスの近似関数によって与えられたアルゴリズムを使用しているため，PRNG の内部機能に対する人間の洞察は得ることは困難である．

本研究の DNN ベースの乱数生成器には，リバースエンジニアリングへの耐性と，サーバやクライアント間のクロスプラットフォーム移植性という 2 つの利点がある．これらの利点を明らかにするために，まず PRNG がゲームやソーシャルネットワーキングサービスなどのモバイルアプリケーションでどのように一般的に使用されているかを概説する．疑似乱数は，暗号化やセキュリティのようなリスクの高い分野だけでなく，ユーザインタフェースにおける自然で滑らかなアニメーションのパラメータや，ゲームにおけるコンピュータ操作の非プレイヤキャラクタ（NPC）の制御パラメータなど，ランダム性や非恣意性が要求される分野でも広く利用されている．特に

PRNG は，ゲームの勝敗を左右するゲーム業界では必須の技術として認知されている．しかし，PRNG のシードに対するリバースエンジニアリング [124] を行い，次の乱数を予想するようなゲームレギュレーションに違反する行為は，Unity [125] などのゲーム開発ツールの汎用化により飛躍的に増加している．不用意に使用される PRNG のリバースエンジニアリングを防止するために，シンプルで移植性の高い方法を提供することが極めて重要である．

そのため，ゲーム業界では，リバース・エンジニアリングを防止する最も単純で実用的な方法として，乱数生成処理を暗号化し，復号鍵を数週間ごとに頻繁に変更する方法が採用されてきた．このようなブルートフォース方式は，鍵の変更頻度が復号に必要な時間やコスト，あるいは復号鍵を見つけるのに必要な時間やコストを上回る限り有効である．しかし，疑似乱数生成器のアルゴリズムは既に知られており，MT 法 [21]，Multiply-With-Carry (MWC) [126]，Xorshift [116]，The 64-bit Maximally Equidistributed F_2-Linear Generators with Mersenne Prime Period (MELG-64) [127] などのように，疑似乱数生成器のアルゴリズムは既に知られており，数も限られているため，メモリ上のシード位置が解読されると，次の乱数生成結果が単純に予想されてしまう可能性がある．新しい疑似乱数生成アルゴリズムは少なくとも数年ごとにしか開発されないため，乱数生成器の推測を防ぐために毎週新しい PRNG アルゴリズムを開発しようとするのは全く非現実的である．

LPRNG は，数学の専門家ではない一般の技術者が，乱数発生器を毎週自動的に作成することを可能にする． このため，学習済み PRNG を利用したアプリケーションの動作を予測することは不可能であり，たとえシードのメモリ位置が開示されていたとしても，一般的に利用可能なリバースエンジニアリング技術を使って学習済み PRNG の動作を調査することは困難である．学習済み PRNG は，乱数生成周期の更新頻度を高める新しいリバースエンジニアリング対策戦略を提供し，暗号鍵を頻繁に更新する通常のアプローチと併用することができる．LPRNG は，既存の手法で生成された乱数列のデータセットから，毎週定期的に新しい PRNG を生成することを可能にする．本研究は，スマートフォンなどのエッジデバイスに新しい PRNG を配信するコストを大幅に削減し，PRNG とその周辺システムをクラックして特定するのにかかるコストよりも頻繁に PRNG を更新できるようにする．

LPRNG モデルの移植性に関しては，特定のプログラミング言語や CPU アーキテクチャに依存しない「マシン／言語ニュートラル」なデータ構造である．今日のサービスは，Apple 社の iOS，MacOS，Google 社の Android，Microsoft 社の Windows，また一般的に普及している Linux といった複数のプラットフォームに対応していなければならない．さらに，クライアントサイドとサーバーサイドの両方で，x86-64，ARM，RISV-V など複数の CPU を採用しています．サービスの実装には，C/C+，Python，Java，JavaScript，C#，Swift など数多くのプログラミング言語が使用されている．そのため，新しい PRNG をすべてのプラットフォーム，CPU，プログラミング言語に適応させるコストは非常に高い．一方，LPRNG は，これらのプラットフォーム，オペレーティングシステム，プログラミング言語に依存しない，機械中立的かつ言語中立的なモデルである Open Neural Network Exchange: ONNX [128] として表現することができる．専用のランタイム [129] とコンパイラ [130] により，LPRNG は様々なプラットフォームやハードウェア上で効果的に乱数を推論することができる．現在のスマートフォンには，Apple Neural Engine のようなニューラルネットワークを高速に実行するための専用チップが搭載されており，NN モ

デルを効率的に実行することができる．したがって，学習済み PRNG をエッジデバイスに配信することは実用的である．

提案する ML を用いた乱数生成法は以下の応用が可能である：

1. LPRNG は，統計的に有効な PRNG を一度だけ生成することを実現する．このような 1 回限りの PRNG は，数学者やソフトウェア工学の専門家が PRNG から不当な利益を得て，その知識を悪用することを防ぐことが可能である．LPRNG は，PRNG の内部機能に対する人間の洞察を含まないため，一般人や専門家を含む全ての人に予測不可能な振る舞いを提供する．この特徴により，不正に最適化され調整された乱数を使用して望ましい結果を得ることは不可能である．
2. 本 LPRNG は，PC やスマートフォンなどのクライアントサイドで PRNG を利用するソフトウェア開発者向けに，高水準のアンチタンパー機能を提供する．LPRNG の振る舞いを調べるために必要な学習過程やパラメータを知ることができないため，たとえリバースエンジニアリングによって種番号が開示されたとしても，悪意のあるユーザが次に生成される乱数を予測することは非常に困難である．

第 10 章

結論

本研究では，現代の計算機科学において，密接な関係がある乱数生成手法に着目する．各計算機シミュレーション手法に特化したアルゴリズムを開発し，従来よりも高速なシミュレーションを達成した．そして，深層学習モデルを用いた乱数生成手法の開発方法を示し，新しい乱数生成のあり方とその応用方法を提案した．

10.1　分子シミュレーションと乱数生成手法

本研究では具体的に以下の項目を実施した．

1. 並列計算時に用いられる散逸粒子動力学法の暗号学的ハッシュ関数 TEA の内部を軽量化することによる疑似乱数の性質の変化を調べ，水分子系の DPD シミュレーションに与える影響を解析した．
2. TEA の乱数生成法をもとに，分子シミュレーションでの利用により適した新規乱数生成手法を設計した．
3. 新手法を用いて生成した乱数を NIST 乱数検定を用いて検証する．また，乱数を生成する上で，より厳しい条件の場合に対しても同様に NIST 乱数検定をおこない，乱数を評価した．
4. 新手法を用いて DPD シミュレーションを実行し，検証した．
5. 新規乱数生成手法を用いた場合の計算コストの変化を調べた．
6. より一般的な分子シミュレーションである，MC 法に適用した乱数生成手法を開発した．
7. 本手法を用いた場合の MC シミュレーションの精度を比較した．

これらの結果得られた結論は以下である．

1. 暗号学的ハッシュ関数である TEA のラウンド回数を変更，また 64bit 仕様の TEA にあえて 128bit の情報を平文として使用することにより軽量化したところ，64bit の平文はラウンド回数 1 回以上，つまり，すべてのラウンド回数で 1 つの等確率性，無規則性を満たす 64bit の一様乱数を生成することが確認できた．また，128bit の平文はラウンド回数 2 回で 1 つの，回数 3 回以上で 2 つの等確率性，無規則性を満たす 64bit の一様乱数を得ることができた．また， 再現した水の系の物性値について動径分布関数，拡散係数，温度，温

度分散を解析したが，等確率性，無規則性を満たす一様乱数を生成する TEA を用いている限り，MT 法を用いた場合の水の系と同様の結果が得られた．なお，当確率性，無規則性を満たさない一様乱数を用いた場合，それを正規乱数に変換する過程で，正規乱数の平均値，分散が設定していた値からずれたことによって，各物性値に生じる差異を明らかにした．

2. 軽量化された TEA のアルゴリズムから，乱数シードに存在する乱雑性の有用性が示唆された．したがって，乱数シードとして用いている粒子座標を NIST 乱数検定によって評価したところ，64bit 倍精度浮動小数点表現（double 型）の内，7-45bit によい乱雑性があることを確認した．その乱雑性を利用し，すべての粒子ペアに与える乱数生成手法を設計し，提案した．
3. 新規乱数生成手法を用いて生成された乱数があるステップにおいて，相関なく生成されているかどうか，また時間発展方向に対しても，相関なく生成されているかどうかを NIST 乱数検定で評価し，乱数として問題なく利用できることを確認した．さらに，時間発展方向の粒子座標により相関が現れるような系を想定し，時間ステップを一般的に利用されているものより，小さく設定し，同様に生成される乱数を評価したところ，一般的によく用いられている無次元時間幅 0.04 の 1/10000 でも問題なく良質な乱数を生成できることを確認した．
4. TEA の軽量化実験で検討した系にも適応し，同様の精度で各物理量が得られることを確認した．また，適応例として POPC 脂質を用いた系を再現し，ベシクルや生体膜の形成を確認した．物理量として，膜圧を生体膜に対して並行軸方向と垂直軸方向で解析し，従来の手法と同様の結果（誤差 0.01% 以下）で得られる事を確認した．
5. 乱数を 100 億個生成し，従来の手法と計算コストを比較したところ，8 回ラウンド処理を含む TEA に比べて，新規手法は約 62 倍の速度で乱数を生成できることを確認した．また，実際に大規模な粗視化水分子の系（108000 粒子）を並列計算で実行し，力の計算に要する計算コストを比較した．その結果，従来の手法に比べて約 27 %の計算コストを削減できたことを確認した．
6. より一般的な手法である MC 法に特化した効率的な乱数生成手法を開発し，MT 法を用いた場合の物性値と比較することで，正しく物性値が得られることを確認した．

10.2 深層学習と乱数生成手法

本研究では具体的に以下の項目を実施した．

1. 深層学習モデルを用いた，より自由な，半自動的な乱数生成手法の設計を行った．
2. 深層学習モデルの学習の遷移と，生成された疑似乱数の性質を NIST 乱数検定によって評価した．

これらの結果得られた結論は以下である．

1. 貧弱な乱数種を持つ WGAN を用いて乱数を生成することに成功した． 得られた乱数を NIST 乱数検定により評価したところ，全てのテストに合格した． また，無限の学

習データを用いたにもかかわらず，過学習と似た現象を観測した．訓練を通じて，深層モデルは Rank，FFT，LinearComplexity テストに容易に合格しやすいことがわかったが，Frequency と CumulativeSums テストには合格する回数が現象したことを発見した．ディープラーニングのメカニズムはまだ完全には解明されていない．本研究では，合格したテストを調べることで，生成器がどのように学習するかを明らかにした．

2. GAN の学習で用いられる識別器は，乱数と貧弱な乱数を区別するため，乱数テストとしても利用できる可能性がある．また，NIST 乱数検定と異なり，入 力データ量に制限がないため，任意の入力データサイズの乱数テストに識別器を用い ることができる．この結果は，機械学習の利用や今後の乱数生成手法の開発に有用な 情報を提供する．

10.3　まとめ

本研究では，主に分子シミュレーションを対象として，効率的な乱数生成手法の提案を行った．本手法のでは，乱数生成アルゴリズムによる乱数生成を実装するというよりも，乱数を利用するアルゴリズム側で現れるカオス的挙動を再利用することで効率的な乱数生成を達成しようとするものである．したがって従来のように複雑な演算を追加で必要とせず，少ない計算コストで乱数生成を実現できる．ここで紹介した手法はどれも少ない回数の bit 演算しか必要とせず，計算コストは大きく削減されるものと考えられる．本研究では取り扱わなかったが，計算機アーキテクチャを考慮した最適化を行えば，より高速な乱数生成を実現できると考えられる．本論文では，散逸粒子動力学法と MC 法での実装を示したが，実装するシミュレーション手法ごとに，最適な乱数生成アルゴリズムを再度開発する必要，また利用する上で必要な水準の乱数品質が維持されていることを注意深く観察する必要がある．このように，実装する手法ごとにアルゴリズムをはじめから検討することは高い人的コストを必要とするものである．そこで本研究では，深層学習技術に着目し，データ駆動型の乱数生成手法の開発に取り組み，NIST 乱数検定を合格するアルゴリズムを開発したことを示した．深層学習を利用した半自動的な乱数生成手法の開発を進めることで，各手法特化型の乱数生成手法開発がより容易になる可能性がある．これらの研究が，さらに効率的で容易な乱数生成手法開発を促進することを期待する．

謝辞

本研究は非常に多くの方々の暖かいご指導，ご支援のもと，進めて参ることができました．この場をお借りして，これまでの 6 年間お世話になった皆様に，心よりお礼申し上げます．

この 6 年間，泰岡顕治教授には特にお世話になり，様々な面で多くのご指導をいただきました．研究テーマ選びの段階では，私の興味を深く掘り下げ，これからの数年間どのようなことをしたいかをお話させていただきました．多くの相談を重ねた上で，自分にあったテーマを提案してくださったことはとても印象的でした．自分の研究に対する熱意が高まり，当時は良いスタートダッシュが切れたと思います．また，研究室に所属していたの 6 年以上もの間，研究を進める上で数多くの困難に頭を悩ましている時は，忙しい中でも先生からお声をかけて下さり，問題を解決するための数多くのアドバイスをくださったので，非常に心強く感じたことは今でも鮮明に覚えております．研究のことだけではなく，複数回に渡る留学生との交流や，企業の案件，海外への留学など，私が今後社会で生きていく上で良い刺激となるような機会をいつも提案して下さりました．心より感謝申し上げます．Cygames. Inc. 研究所所長の倉林修一博士には，企業における研究遂行方法の具体的な指導を頂き，自由に，かつ責任ある研究への取り組み方を学びました．また，博士課程という経済的に厳しい状況下で，その支援を頂いたことは，私にとって大きな助けとなりました．そして，何よりも，自分の興味を追求する研究への挑戦の機会を託していただき，最後まで私の仕事を信頼してくださりました．心より御礼申し上げます．村松眞由准教授には，研究に対する姿勢，また，どのような価値観で生活を送るのかについて，多くのご指導を頂きました．進学と就職で迷い，悩んでいたころも，時間を割いて親身に相談に乗って下さったのを覚えています．本当にありがとうございました．荒井規允准教授には神戸での勉強会や夏の勉強会などでお会いし，私の研究について熱心に聞いてくださいました．そして私の研究テーマに関するディスカッションやアドバイスを何度もいただきました．本当にありがとうございました．山本詠士専任講師には，研究はどのような姿勢で行い，何が重要で何が重要ではないのかを明確に教えてくださったので，自分が今後も研究を続けていく上で，意識する基準ができました．さらに，自分の研究に関係のある論文を多数紹介してくださったので，非常に参考になりました．また，フランスへの IPBS 研究所やオックスフォード大学への留学など非常に貴重な機会を提案し，先方へご紹介下さりました．これらの経験は私の人生においてとても大きなものとなりました．本当にありがとうございました．Oxford 大学の Syma Khalid 教授には半年間という長い期間，共同研究で大変お世話になりました．私が専門とする分野において，海外の最先端研究施設で，これまで取り扱ったことのないスケールでのシミュレーションの研究をご指導下さりました．分子生物化学の分野において，計算機科学のアプローチだけでなく，実験を専門とする研究者達と活発な議論と共同研究ができたことは私にとって大変刺激的な経験でした．本当にありがとうござい

ました．

同期の遠藤克浩博士，荻野健太さん，小野祐為さん，澤眞詩さんには，この数年間心を支えてくれました．一番身近にいてくれた同期の存在が私の生活に大きなものであったことを改めて実感しました．本当にありがとうございます．最後に，東京という地元から遠い場所での生活を支援し，見守ってくださった母，祖母の支えがあったからこそ，充実した研究生活を送ることができました．本当にありがとうございました．全ての方のお名前をここに挙げることはできませんが，お世話になった全ての方に感謝の意を表します. 本当にありがとうございました．

参考文献

[1] Ray Kurzweil. The law of accelerating returns. In *Alan Turing: Life and legacy of a great thinker*, pp. 381–416. Springer, 2001.

[2] Ray Kurzweil. The singularity is near. In *Ethics and emerging technologies*, pp. 393–406. Springer, 2005.

[3] Mark Buchanan. The law of accelerating returns. *Nature Physics*, Vol. 4, No. 7, pp. 507–507, 2008.

[4] OpenAI. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[5] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Roziere, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[6] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 24824–24837, 2022.

[7] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchan. Chatgpt is not all you need. a state of the art review of large generative ai models. arXiv preprint arXiv:2301.04655, 2023.

[8] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, Vol. 21, No. 6, pp. 1087–1092, 1953.

[9] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, Vol. 2. Springer, 1999.

[10] Christelle Miqueu, Jose M Miguez, Manuel M Pineiro, Thomas Lafitte, and Bruno Mendiboure. Simultaneous application of the gradient theory and monte carlo molecular simulation for the investigation of methane/water interfacial properties. *The Journal of Physical Chemistry B*, Vol. 115, No. 31, pp. 9618–9625, 2011.

[11] George P Lithoxoos, Anastasios Labropoulos, Loukas D Peristeras, Nikolaos Kanellopoulos, Jannis Samios, and Ioannis G Economou. Adsorption of n2, ch4, co and co2 gases in single walled carbon nanotubes: A combined experimental and monte carlo molecular simulation study. *The Journal of Supercritical Fluids*, Vol. 55, No. 2, pp.

510–523, 2010.

[12] M de Lourdes Chávez, Liberto de Pablo, and Juan J de Pablo. Monte carlo molecular simulation of the hydration of k-montmorillonite at 353 k and 625 bar. *Langmuir*, Vol. 20, No. 24, pp. 10764–10770, 2004.

[13] Alfredo Palace Carvalho, José ANF Gomes, and M Natália DS Cordeiro. Parallel implementation of a monte carlo molecular simulation program. *Journal of Chemical Information and Computer Sciences*, Vol. 40, No. 3, pp. 588–592, 2000.

[14] Aleksandr Aleksandrovich Belov, Nikolai Nikolaevich Kalitkin, and Maksim Aleksandrovich Tintul. Unreliability of available pseudorandom number generators. *Computational Mathematics and Mathematical Physics*, Vol. 60, pp. 1747–1753, 2020.

[15] Berni Julian Alder and Thomas Everett Wainwright. Phase transition for a hard sphere system. *The Journal of chemical physics*, Vol. 27, No. 5, pp. 1208–1209, 1957.

[16] 岡崎進, 吉井範行. コンピュータシミュレーションの基礎（第 2 版）. 化学同人株式会社, 2011.

[17] P J Hoogerbrugge and J M V A Koelman. Simulating Microscopic Hydrodynamic Phenomena with Dissipative Particle Dynamics. *Europhysics Letters*, Vol. 19, No. 3, pp. 155–160, June 1992.

[18] Pep Español and Patrick Warren. Statistical Mechanics of Dissipative Particle Dynamics. *Europhysics Letters*, Vol. 30, No. 4, pp. 191–196, May 1995.

[19] Robert D Groot and Patrick B Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *The Journal of Chemical Physics*, Vol. 107, No. 11, pp. 4423–4435, June 1998.

[20] Carolyn L Phillips, Joshua A Anderson, and Sharon C Glotzer. Pseudo-random number generation for brownian dynamics and dissipative particle dynamics simulations on gpu devices. *Journal of Computational Physics*, Vol. 230, No. 19, pp. 7191–7201, 2011.

[21] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Vol. 8, No. 1, pp. 3–30, January 1998.

[22] David J Wheeler and Roger M Needham. TEA, a tiny encryption algorithm. In *Fast Software Encryption*, pp. 363–366. Springer, Berlin, Heidelberg, December 1994.

[23] VV Desai, VB Deshmukh, and DH Rao. Pseudo random number generator using elman neural network. In *2011 IEEE Recent Advances in Intelligent Computational Systems*, pp. 251–254. IEEE, 2011.

[24] VV Desai, Ravindra T Patil, VB Deshmukh, and DH Rao. Pseudo random number generator using time delay neural network. *World*, Vol. 2, No. 10, pp. 165–169, 2012.

[25] Veena Desai, Ravindra Patil, and Dandina Rao. Using layer recurrent neural network to generate pseudo random number sequences. *International Journal of Computer Science Issues*, Vol. 9, No. 2, pp. 324–334, 2012.

[26] Young-Seob Jeong, Kyojoong Oh, Chung-Ki Cho, and Ho-Jin Choi. Pseudo random

number generation using lstms and irrational numbers. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 541–544. IEEE, 2018.

[27] Luca Pasqualini and Maurizio Parton. Pseudo random number generation through reinforcement learning and recurrent neural networks. *Algorithms*, Vol. 13, No. 11, p. 307, 2020.

[28] Kayvan Tirdad and Alireza Sadeghian. Hopfield neural networks as pseudo random number generators. In *2010 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 1–6. IEEE, 2010.

[29] Sarab M Hameed and Layla M Mohammed Ali. Utilizing hopfield neural network for pseudo-random number generator. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–5. IEEE, 2018.

[30] Yu-Hua Wang, Zhi-Dong Shen, and Huan-Guo Zhang. Pseudo random number generator based on hopfield neural network. In *2006 International Conference on Machine Learning and Cybernetics*, pp. 2810–2813. IEEE, 2006.

[31] Marcello De Bernardi, MHR Khouzani, and Pasquale Malacaria. Pseudo-random number generation using generative adversarial networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 191–200. Springer, 2018.

[32] Rajvardhan Oak, Chaitanya Rahalkar, and Dhaval Gujar. Poster: Using generative adversarial networks for secure pseudorandom number generation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2597–2599, 2019.

[33] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

[34] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. arXiv preprint arXiv:1406.2661, 2014.

[35] Shiying Ma, Yi Hu, and Rong Wang. Self-Assembly of Polymer Tethered Molecular Nanoparticle Shape Amphiphiles in Selective Solvents. *Macromolecules*, Vol. 48, No. 9, pp. 3112–3120, April 2015.

[36] G Dorenbos. Competition between side chain length and side chain distribution: Searching for optimal polymeric architectures for application in fuel cell membranes. *Journal of Power Sources*, Vol. 276, pp. 328–339, February 2015.

[37] Wenhua Jiang, Jianhua Huang, Yongmei Wang, and Mohamed Laradji. Hydrodynamic interaction in polymer solutions simulated with dissipative particle dynamics. *The Journal of Chemical Physics*, Vol. 126, No. 4, p. 044901, January 2007.

[38] Sergey Litvinov, Marco Ellero, Xiangyu Hu, and Nikolaus A Adams. Smoothed dissipative particle dynamics model for polymer molecules in suspension. *Physical Review E*, Vol. 77, No. 6, p. 066703, June 2008.

[39] G Maurel, B Schnell, F Goujon, M Couty, and P Malfreyt. Multiscale Modeling Approach toward the Prediction of Viscoelastic Properties of Polymers. *Journal of Chemical Theory and Computation*, Vol. 8, No. 11, pp. 4570–4579, November 2012.

[40] Noriyoshi Arai, Takuma Akimoto, Eiji Yamamoto, Masato Yasui, and Kenji Yasuoka. Poisson property of the occurrence of flip-flops in a model membrane. *The Journal of Chemical Physics*, Vol. 140, No. 6, p. 064901, February 2014.

[41] Liuyang Zhang, Matthew Becton, and Xianqiao Wang. Designing nanoparticle translocation through cell membranes by varying amphiphilic polymer coatings. *The Journal of Physical Chemistry. B*, Vol. 119, No. 9, pp. 3786–3794, March 2015.

[42] Haizhen Zhang, Qiuju Ji, Changjin Huang, Sulin Zhang, Bing Yuan, Kai Yang, and Yu-qiang Ma. Cooperative Transmembrane Penetration of Nanoparticles. *Scientific Reports*, Vol. 5, No. 1, p. 543, May 2015.

[43] Gernot Guigas and Matthias Weiss. Membrane protein mobility depends on the length of extra-membrane domains and on the protein concentration. *Soft Matter*, Vol. 11, No. 1, pp. 33–37, December 2014.

[44] Zhigang Li and German Drazer. Hydrodynamic interactions in dissipative particle dynamics. *Physics of Fluids*, Vol. 20, No. 10, p. 103601, October 2008.

[45] Mohamed Laradji and Michael J A Hore. Nanospheres in phase-separating multicomponent fluids: a three-dimensional dissipative particle dynamics simulation. *The Journal of Chemical Physics*, Vol. 121, No. 21, pp. 10641–10647, December 2004.

[46] N Phan-Thien, N Mai-Duy, and B C Khoo. A spring model for suspended particles in dissipative particle dynamics. *Journal of Rheology*, Vol. 58, No. 4, pp. 839–867, May 2014.

[47] E S Boek, P V Coveney, H N W Lekkerkerker, and P van der Schoot. Simulating the rheology of dense colloidal suspensions using dissipative particle dynamics. *Physical Review E*, Vol. 55, No. 3, pp. 3124–3133, March 1997.

[48] Noriyoshi Arai, Kenji Yasuoka, and X C Zeng. Nanochannel with uniform and Janus surfaces: shear thinning and thickening in surfactant solution. *Langmuir*, Vol. 28, No. 5, pp. 2866–2872, February 2012.

[49] Jianchang Xu, Shuangqing Sun, Zhikun Wang, Shiyuan Peng, Songqing Hu, and Lijuan Zhang. pH-Induced evolution of surface patterns in micelles assembled from dirhamnolipids: dissipative particle dynamics simulation. *Physical Chemistry Chemical Physics*, Vol. 20, No. 14, pp. 9460–9470, April 2018.

[50] Ruslan Shupanov, Alexander Chertovich, and Pavel Kos. Micellar polymerization: Computer simulations by dissipative particle dynamics. *Journal of Computational Chemistry*, Vol. 39, No. 19, pp. 1275–1284, 2018.

[51] Richard L Anderson, David J Bray, Annalaura Del Regno, Michael A Seaton, Andrea S Ferrante, and Patrick B Warren. Micelle Formation in Alkyl Sulfate Surfactants Using Dissipative Particle Dynamics. *Journal of Chemical Theory and Computation*, Vol. 14,

No. 5, pp. 2633–2643, March 2018.

[52] I Pagonabarraga and D Frenkel. Dissipative particle dynamics for interacting systems. *The Journal of Chemical Physics*, Vol. 115, No. 11, pp. 5015–5026, September 2001.

[53] Patrick B Warren. Hydrodynamic Bubble Coarsening in Off-Critical Vapor-Liquid Phase Separation. *Physical Review Letters*, Vol. 87, No. 22, p. 225702, November 2001.

[54] S Y Trofimov, E L F Nies, and M A J Michels. Thermodynamic consistency in dissipative particle dynamics simulations of strongly nonideal liquids and liquid mixtures. *The Journal of Chemical Physics*, Vol. 117, No. 20, pp. 9383–9394, November 2002.

[55] J Bonet Avalos and A D Mackie. Dissipative particle dynamics with energy conservation. *Europhysics Letters*, Vol. 40, No. 2, pp. 141–146, October 1997.

[56] P Español. Dissipative particle dynamics with energy conservation. *Europhysics Letters*, Vol. 40, No. 6, pp. 631–636, December 1997.

[57] Pep Español and Mariano Revenga. Smoothed dissipative particle dynamics. *Physical Review E*, Vol. 67, No. 2, p. 026705, February 2003.

[58] Pep Español, Mar Serrano, and Hans Christian Öttinger. Thermodynamically Admissible Form for Discrete Hydrodynamics. *Physical Review Letters*, Vol. 83, No. 22, pp. 4542–4545, November 1999.

[59] Rakesh Vaiwala, Sameer Jadhav, and Rochish Thaokar. Electroporation Using Dissipative Particle Dynamics with a Novel Protocol for Applying Electric Field. *Journal of Chemical Theory and Computation*, Vol. 15, No. 1, pp. 603–612, January 2019.

[60] Abeer Khedr and Alberto Striolo. DPD Parameters Estimation for Simultaneously Simulating Water-Oil Interfaces and Aqueous Nonionic Surfactants. *Journal of Chemical Theory and Computation*, Vol. 14, No. 12, pp. 6460–6471, December 2018.

[61] Ming-Tsung Lee, Aleksey Vishnyakov, and Alexander V Neimark. Modeling Proton Dissociation and Transfer Using Dissipative Particle Dynamics Simulation. *Journal of Chemical Theory and Computation*, Vol. 11, No. 9, pp. 4395–4403, August 2015.

[62] N Mai-Duy, N Phan-Thien, and T Tran-Cong. Imposition of physical parameters in dissipative particle dynamics. *Computer Physics Communications*, Vol. 221, pp. 290–298, December 2017.

[63] Pep Español and Patrick B Warren. Perspective: Dissipative particle dynamics. *The Journal of Chemical Physics*, Vol. 146, No. 15, p. 150901, April 2017.

[64] Yu-Hang Tang and George Em Karniadakis. Accelerating dissipative particle dynamics simulations on GPUs: Algorithms, numerics and applications. *Computer Physics Communications*, Vol. 185, No. 11, pp. 2809–2822, November 2014.

[65] Jens Glaser, Trung Dac Nguyen, Joshua A Anderson, Pak Lui, Filippo Spiga, Jaime A Millan, David C Morse, and Sharon C Glotzer. Strong scaling of general-purpose molecular dynamics simulations on GPUs. *Computer Physics Communications*, Vol. 192, pp. 97–107, July 2015.

[66] Ansel L Blumers, Yu-Hang Tang, Zhen Li, Xuejin Li, and George E Karniadakis. GPU-

accelerated red blood cells simulations with transport dissipative particle dynamics. *Computer Physics Communications*, Vol. 217, pp. 171–179, August 2017.

[67] Brad Lee Holian, Ora E Percus, Tony T Warnock, and Paula A Whitlock. Pseudorandom number generator for massively parallel molecular-dynamics simulations. *Physical Review E*, Vol. 50, No. 2, pp. 1607–1615, August 1994.

[68] W B Langdon. A fast high quality pseudo random number generator for graphics processing units. In *2008 IEEE Congress on Evolutionary Computation (CEC)*, pp. 459–465. IEEE, 2008.

[69] A Zhmurov, K Rybnikov, Y Kholodov, and V Barsegov. Generation of Random Numbers on Graphics Processors: Forced Indentation In Silico of the Bacteriophage HK97. *The Journal of Physical Chemistry. B*, Vol. 115, No. 18, pp. 5278–5288, December 2010.

[70] Stanley Tzeng and Li-Yi Wei. *Parallel white noise generation on a GPU via cryptographic hash*. ACM, New York, New York, USA, February 2008.

[71] Fahad Zafar, Marc Olano, and Aaron Curtis. Gpu random numbers via the tiny encryption algorithm. In *Proceedings of the Conference on High Performance Graphics*, pp. 133–141. Eurographics Association, 2010.

[72] Yaser Afshar, Friederike Schmid, Ahmadreza Pishevar, and S Worley. Exploiting seeding of random number generators for efficient domain decomposition parallelization of dissipative particle dynamics. *Computer Physics Communications*, Vol. 184, No. 4, pp. 1119–1128, 2013.

[73] Carine Malheiro, Bruno Mendiboure, Jose-Manuel Miguez, Manuel M Pineiro, and Christelle Miqueu. Nonlocal density functional theory and grand canonical monte carlo molecular simulations of water adsorption in confined media. *The Journal of Physical Chemistry C*, Vol. 118, No. 43, pp. 24905–24914, 2014.

[74] PJ Hoogerbrugge and JMVA Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *Europhysics Letters*, Vol. 19, No. 3, p. 155, 1992.

[75] Pep Español and Patrick Warren. Statistical mechanics of dissipative particle dynamics. *Europhysics Letters*, Vol. 30, No. 4, p. 191, 1995.

[76] Robert D Groot and Patrick B Warren. Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation. *The Journal of Chemical Physics*, Vol. 107, No. 11, pp. 4423–4435, 1997.

[77] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

[78] 岩下武史, 片桐孝洋, 高橋大介. スパコンを知る　その基礎から最新の動向まで. 東京大学出版会, 2015.

[79] Masaki Iwasawa, Ataru Tanikawa, Natsuki Hosono, Keigo Nitadori, Takayuki Muranushi, and Junichiro Makino. Implementation and performance of FDPS: a framework for developing parallel particle simulation codes. *Publications of the Astronomical*

Society of Japan, Vol. 68, No. 4, p. 54, August 2016.

[80] Yu-Hang Tang and George Em Karniadakis. Accelerating dissipative particle dynamics simulations on gpus: Algorithms, numerics and applications. *Computer Physics Communications*, Vol. 185, No. 11, pp. 2809–2822, 2014.

[81] 脇本和昌. 乱数の知識. 森北出版株式会社, 1981.

[82] 結城浩. 暗号技術入門. SB クリエイティブ株式会社, 2015.

[83] Herbert A Sturges. The choice of a class interval. *Journal of the American Statistical Association*, Vol. 21, No. 153, pp. 65–66, 1926.

[84] 藤井光章. 統計学 One Point 7 暗号と乱数ー乱数の統計的検定ー. 共立出版, 2018.

[85] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va, 2001.

[86] Siddhant Jape, Vinod Kumar Joshi, and Prashanth Barla. Design of a novel non-volatile hybrid spintronic true random number generator. *International Journal of Circuit Theory and Applications*, Vol. 50, No. 5, pp. 1487–1501, 2022.

[87] Kyungduk Kim, Stefan Bittner, Yongquan Zeng, Stefano Guazzotti, Ortwin Hess, Qi Jie Wang, and Hui Cao. Massively parallel ultrafast random bit generation with a chip-scale laser. *Science*, Vol. 371, No. 6532, pp. 948–952, 2021.

[88] David J Wheeler and Roger M Needham. Tea, a tiny encryption algorithm. In *International Workshop on Fast Software Encryption*, pp. 363–366. Springer, 1994.

[89] Vikram Reddy Andem. *A cryptanalysis of the tiny encryption algorithm*. PhD thesis, University of Alabama, 2003.

[90] George EP Box, Mervin E Muller, et al. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, Vol. 29, No. 2, pp. 610–611, 1958.

[91] Hideo Doi, Koji Okuwaki, Yuji Mochizuki, Taku Ozawa, and Kenji Yasuoka. Dissipative particle dynamics (dpd) simulations with fragment molecular orbital (fmo) based effective parameters for 1-palmitoyl-2-oleoyl phosphatidyl choline (popc) membrane. *Chemical Physics Letters*, Vol. 684, pp. 427–432, 2017.

[92] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

[93] Hui Xu, Xiaojun Tong, and Xianwen Meng. An efficient chaos pseudo-random number generator applied to video encryption. *Optik*, Vol. 127, No. 20, pp. 9305–9319, 2016.

[94] Bo Li, Xiaofeng Liao, and Yan Jiang. A novel image encryption scheme based on improved random number generator and its implementation. *Nonlinear Dynamics*, Vol. 95, No. 3, pp. 1781–1805, 2019.

[95] Yukiko Sato, Stefan Brückner, Shuichi Kurabayashi, and Ikumi Waragai. An empirical taxonomy of monetized random reward mechanisms in games. *Proceedings of DiGRA 2020*, Vol. 21, , 2020.

[96] Saihui Hou and Zilei Wang. Weighted channel dropout for regularization of deep con-

volutional neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, pp. 8425–8432, 2019.

[97] Jordan J Bird, Anikó Ekárt, and Diego R Faria. On the effects of pseudorandom and quantum-random number generators in soft computing. *Soft Computing*, Vol. 24, No. 12, pp. 9243–9256, 2020.

[98] Ryo Kobayashi. Modeling and numerical simulations of dendritic crystal growth. *Physica D: Nonlinear Phenomena*, Vol. 63, No. 3-4, pp. 410–423, 1993.

[99] Ryo Kobayashi. A numerical approach to three-dimensional dendritic solidification. *Experimental Mathematics*, Vol. 3, No. 1, pp. 59–81, 1994.

[100] Atsushi Uchida, Kazuya Amano, Masaki Inoue, Kunihito Hirano, Sunao Naito, Hiroyuki Someya, Isao Oowada, Takayuki Kurashige, Masaru Shiki, Shigeru Yoshimori, et al. Fast physical random bit generation with chaotic semiconductor lasers. *Nature Photonics*, Vol. 2, No. 12, pp. 728–732, 2008.

[101] Igor Reidler, Yaara Aviad, Michael Rosenbluh, and Ido Kanter. Ultrahigh-speed random number generation based on a chaotic semiconductor laser. *Physical Review Letters*, Vol. 103, No. 2, p. 024102, 2009.

[102] Ido Kanter, Yaara Aviad, Igor Reidler, Elad Cohen, and Michael Rosenbluh. An optical ultrafast random bit generator. *Nature Photonics*, Vol. 4, No. 1, pp. 58–61, 2010.

[103] Apostolos Argyris, Stavros Deligiannidis, Evangelos Pikasis, Adonis Bogris, and Dimitris Syvridis. Implementation of 140 gb/s true random bit generator based on a chaotic photonic integrated circuit. *Optics Express*, Vol. 18, No. 18, pp. 18763–18768, 2010.

[104] Kunihito Hirano, Taiki Yamazaki, Shinichiro Morikatsu, Haruka Okumura, Hiroki Aida, Atsushi Uchida, Shigeru Yoshimori, Kazuyuki Yoshimura, Takahisa Harayama, and Peter Davis. Fast random bit generation with bandwidth-enhanced chaos in semiconductor lasers. *Optics Express*, Vol. 18, No. 6, pp. 5512–5524, 2010.

[105] Jianzhong Zhang, Yuncai Wang, Ming Liu, Lugang Xue, Pu Li, Anbang Wang, and Mingjiang Zhang. A robust random number generator based on differential comparison of chaotic laser signals. *Optics Express*, Vol. 20, No. 7, pp. 7496–7506, 2012.

[106] Xiao-Zhou Li and Sze-Chun Chan. Heterodyne random bit generation using an optically injected semiconductor laser in chaos. *IEEE Journal of Quantum Electronics*, Vol. 49, No. 10, pp. 829–838, 2013.

[107] Neus Oliver, Miguel Cornelles Soriano, David W Sukow, and Ingo Fischer. Fast random bit generation using a chaotic laser: approaching the information theoretic limit. *IEEE Journal of Quantum Electronics*, Vol. 49, No. 11, pp. 910–918, 2013.

[108] Martin Virte, Emeric Mercier, Hugo Thienpont, Krassimir Panajotov, and Marc Sciamanna. Physical random bit generation from chaotic solitary laser diode. *Optics Express*, Vol. 22, No. 14, pp. 17271–17280, 2014.

[109] Ryohsuke Sakuraba, Kento Iwakawa, Kazutaka Kanno, and Atsushi Uchida. Tb/s physical random bit generation with bandwidth-enhanced chaos in three-cascaded semicon-

ductor lasers. *Optics Express*, Vol. 23, No. 2, pp. 1470–1490, 2015.

[110] Xi Tang, Zheng-Mao Wu, Jia-Gui Wu, Tao Deng, Jian-Jun Chen, Li Fan, Zhu-Qiang Zhong, and Guang-Qiong Xia. Tbits/s physical random bit generation based on mutually coupled semiconductor laser chaotic entropy source. *Optics Express*, Vol. 23, No. 26, pp. 33130–33141, 2015.

[111] T Butler, C Durkan, D Goulding, S Slepneva, B Kelleher, SP Hegarty, and G Huyet. Optical ultrafast random number generation at 1 tb/s using a turbulent semiconductor ring cavity laser. *Optics Letters*, Vol. 41, No. 2, pp. 388–391, 2016.

[112] Susumu Shinohara, Kenichi Arai, Peter Davis, Satoshi Sunada, and Takahisa Harayama. Chaotic laser based physical random bit streaming system with a computer application interface. *Optics Express*, Vol. 25, No. 6, pp. 6461–6474, 2017.

[113] Kazusa Ugajin, Yuta Terashima, Kento Iwakawa, Atsushi Uchida, Takahisa Harayama, Kazuyuki Yoshimura, and Masanobu Inubushi. Real-time fast physical random number generator with a photonic integrated circuit. *Optics Express*, Vol. 25, No. 6, pp. 6511–6523, 2017.

[114] Shuiying Xiang, Bo Wang, Yang Wang, Yanan Han, Aijun Wen, and Yue Hao. 2.24-tb/s physical random bit generation with minimal post-processing based on chaotic semiconductor lasers network. *Journal of Lightwave Technology*, Vol. 37, No. 16, pp. 3987–3993, 2019.

[115] Lawrence E Bassham III, Andrew L Rukhin, Juan Soto, James R Nechvatal, Miles E Smid, Elaine B Barker, Stefan D Leigh, Mark Levenson, Mark Vangel, David L Banks, et al. Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010.

[116] George Marsaglia, et al. Xorshift rngs. *Journal of Statistical Software*, Vol. 8, No. 14, pp. 1–6, 2003.

[117] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, Vol. 521, No. 7553, pp. 436–444, May 2015.

[118] Luca Pasqualini and Maurizio Parton. Pseudo random number generation: A reinforcement learning approach. *Procedia Computer Science*, Vol. 170, pp. 1122–1127, 2020.

[119] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028, 2017.

[120] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957, 2018.

[121] Tabea Kossen, Pooja Subramaniam, Vince I Madai, Anja Hennemuth, Kristian Hildebrand, Adam Hilbert, Jan Sobesky, Michelle Livne, Ivana Galinovic, Ahmed A Khalil, et al. Synthesizing anonymized and labeled tof-mra patches for brain vessel segmentation using generative adversarial networks. *Computers in biology and medicine*, Vol.

131, p. 104254, 2021.

[122] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, Vol. 9, No. 11, 2008.

[123] Tilo Fischer. Testing cryptographically secure pseudo random number generators with artificial neural networks. In *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 1214–1223. IEEE, 2018.

[124] Igor Tomičić, Petra Grd, and Markus Schatten. Reverse engineering of the mmorpg client protocol. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1099–1104. IEEE, 2019.

[125] John K Haas. A history of the unity game engine. *Diss. Worcester Polytechnic Institute*, Vol. 483, No. 2014, p. 484, 2014.

[126] George Marsaglia. Random number generators. *Journal of Modern Applied Statistical Methods*, Vol. 2, No. 1, p. 2, 2003.

[127] Shin Harase and Takamitsu Kimoto. Implementing 64-bit maximally equidistributed f2-linear generators with mersenne prime period. *ACM Transactions on Mathematical Software (TOMS)*, Vol. 44, No. 3, pp. 1–11, 2018.

[128] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. *GitHub repository*, p. 54, 2019.

[129] Onnx runtime developers: Onnx runtime, accessed 2023-02-21. `https://onnxruntime.ai/`.

[130] Tian Jin, Gheorghe-Teodor Bercea, Tung D Le, Tong Chen, Gong Su, Haruki Imai, Yasushi Negishi, Anh Leu, Kevin O'Brien, Kiyokuni Kawachiya, et al. Compiling onnx neural network models using mlir. arXiv preprint arXiv:2008.08272, 2020.