

## A-20 CUDAを用いた超伝導体動力学の並列高速シミュレーション

杉本陸\*, 有田拳\*\*, 小田部荘司\*\*

(\*九州工業大学物理情報工学科, \*\*九州工業大学院情報工学研究院)

## 1. はじめに

超伝導体における渦糸の生成・消滅や相転移といった動的現象の解明は、基礎物理学のみならず、次世代超伝導デバイスの設計においても極めて重要である。これらの非平衡・非線形現象を記述する有効な理論として時間依存ギンツブルグ・ランダウ (Time Depended Ginzburg Landu: TDGL) 方程式が知られているが、その数値シミュレーションは膨大な計算コストを伴う。特に、2次元以上の大規模格子系を扱う場合、従来のCPUによる逐次計算ではシミュレーションの時間的コストが高いことが問題として知られている。

本研究の目的は、この計算ボトルネックを抜本的に解消することにある。そのために、GPUが持つ大規模な並列計算能力に着目し、CUDA (Compute Unified Device Architecture) を用いたTDGL方程式の高速並列シミュレーションプログラムを開発した。本稿では、JavaとJCudaフレームワーク上で、計算集約的な秩序パラメータとベクトルポテンシャルの更新処理をCUDAカーネルとして実装し、CPU実装から約87倍の高速化を実現した手法について報告する。

## 2. GPUによる並列実装と高速化

本シミュレーションの高速化は、計算集約的な処理のGPUオフロードと、データ転送を最小化するメモリ管理の最適化によって実現した。開発にはJavaとCUDA C++をJCudaで連携させ、全体の制御をCPU、TDGL方程式の時間発展計算をGPUに分離した。格子点ごとの秩序パラメータやリンク変数の更新といった並列性の高い処理は、複数のCUDAカーネルとして実装し、GPU上で並列実行する。

CPU-GPU間のデータ転送オーバーヘッドを削減するため、大規模な配列データはシミュレーション開始時に一度だけGPUへ転送し、以降の計算はGPU内部で完結させる。結果の描画や保存など、必要な場合にのみデータをCPUへコピーバックする設計である。なお、このGPU化の過程では、開発効率向上のため生成AIを補助的に活用した。コードの雛形作成に利用したが、出力されたコードの構文エラー等は手動で修正し、実装を完成させた。

この並列実装とデータ転送回数を最小限にする実装を組み合わせる設計により、計算時間の大部分をGPUの高速な演算に費やすことが可能となり、1秒あたりに計算されるフレーム数を大幅に増加させることに成功した。

## 3. シミュレーション結果と考察

計算速度の向上を評価するため、CPU (AMD Ryzen 7 7800X3D) による逐次計算実装と、GPU (NVIDIA GeForce RTX 4070Ti SUPER) による並列計算実装の性能を、同一条件下で比較した。なお、本稿で示す性能値はこれらの特定デバイスで計測したものであり、特にGPUを用いたシミュレーション性能はデバイスによって大きく変動する可能性がある。計算速度は1秒あたりのフレーム更新数 (FPS) で測定した。その結果と比較を図1に示す。また、シミュレーションによって得られた画像の例を図2に示す。濃淡は秩序パラメータの大きさを示し、色は秩序パラメータの位相情報を示している。黒い点に見えるところは渦糸である。

この性能向上により、従来は困難であった長時間のシミュレーションや、多数の物理パラメータの組み合わせを試行する大規模なパラメータ探索が、現実的な時間で実行可能となった。加えて、パラメータを変更しながら、その影響が系のダイナミクスに与える効果を最小限の時間的コストで確認できる。このリアルタイム性と対話性の獲得は、研究の効率化だけでなく、超伝導現象の直感的理解を促す教育的観点からも極めて有益である。

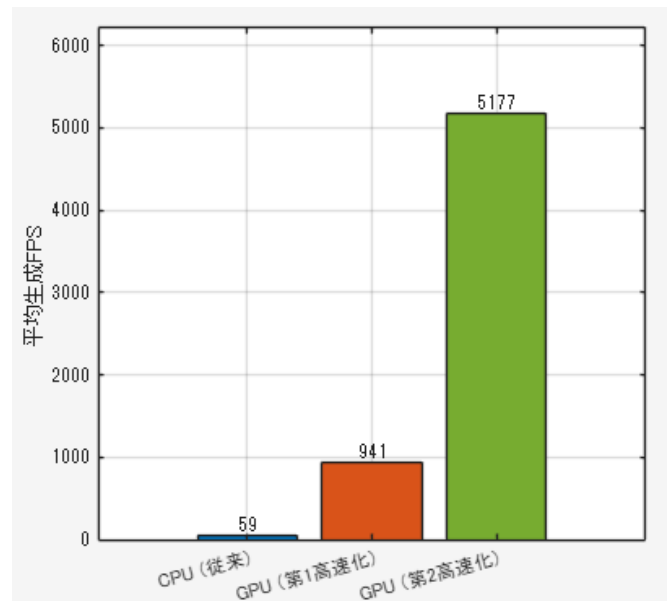


図1. 生成フレーム数の推移をそれぞれ比較したグラフ

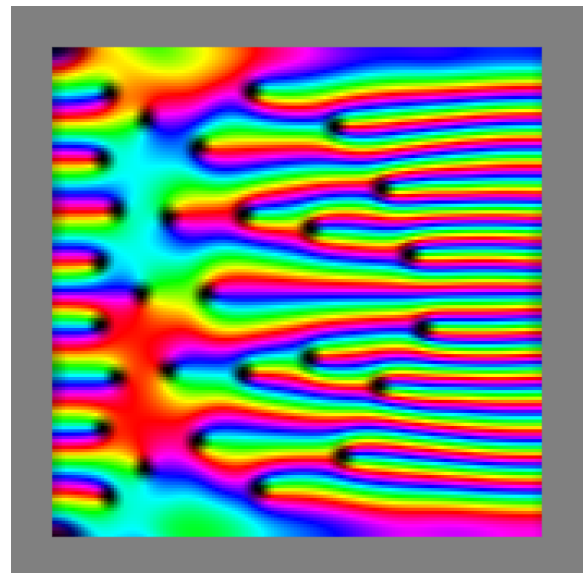


図2. シミュレーションによって得られた画像

## 参考文献

- 1) K. Arita et al., *Physica C* **662** (2024) 1354522.
- 2) T. Matsuno, E.S. Otabe, Y. Mawatari, *Journal of the Physical Society of Japan* **89** (2020) 054006