高次元微分演算子を計算するための確率的テンソル縮約ネットワーク（特願2025-046150、出願人：New York General Group, Inc.、発明者：村上 由宇）

New York General Group
2025

1. 技術分野

[0001]
本発明は、高次元微分演算子を効率的に計算するためのシステム及び方法に関し、特に物理情報ニューラルネットワーク（PINN）や高次元偏微分方程式（PDE）の解法に適用される確率的テンソル縮約ネットワークに関する。本発明は、計算科学、機械学習、科学計算、量子力学、流体力学、金融工学、分子動力学など、高次元微分演算子を含む様々な分野で利用可能である。

2. 背景技術

[0002]
高次元微分演算子を含む最適化問題や偏微分方程式の解法は、科学計算、機械学習、物理シミュレーションなど多くの分野で重要な役割を果たしている。しかし、従来の方法では、入力次元dと微分の次数kが増加するにつれて計算量とメモリ要件が指数関数的に増大する問題がある。

[0003]
具体的には、d次元の関数に対するk次の微分テンソルのサイズはO(d^k)であり、計算グラフのサイズはO(2^(k-1)L)となる（Lは前方計算グラフの演算数）。この指数関数的なスケーリングは、高次元問題や高次微分を含む問題の解決を実質的に不可能にしている。

[0004]
従来技術では、確率的次元勾配降下法（SDGD）やハチンソントレース推定器（HTE）などの手法が提案されているが、これらは特定の微分演算子（主にラプラシアン）に限定されており、任意の高次微分演算子に対応できない。また、これらの手法でも微分の次数kに関する指数関数的なスケーリングの問題は解決されていない。

[0005]
例えば、SDGDは高次元微分演算子を加算項のランダムサブセットで近似することで計算を分散化する。しかし、この方法は各サンプリングされた次元に対して後方モード自動微分を繰り返し適用する必要があり、微分の次数kに関して依然としてO(2^(k-1))のスケーリングを示す。具体的には、SDGDは微分演算子Dを以下のようにランダムにサンプリングされた加算項のサブセットで近似する：

D := ∑^(N_D)_(j=1) D_j ≈ (N_D)/|J| ∑_(j∈J) D_j := D̃^J

ここで、D̃^Jは真の演算子Dを近似するSDGD演算子、Jはサンプリングされたインデックス集合、|J|はバッチサイズである。例えば、d次元ポアソン方程式の場合、N_D = d、D = ∑^d_(j=1) ∂^2/∂x^2_j、加算項はD_j = ∂^2/∂x^2_jである。

[0006]
HTEはヘッセ行列やヤコビ行列のトレースを推定するための古典的な手法であるが、2次以上の微分演算子に対する一般化が困難である。HTEは行列Aのトレースを以下のように確率的に推定する：

tr(A) = E_(v〜p(v))[v^T A v]

ここで、p(v)は等方的な分布、つまりE_(v〜p(v))[vv^T] = Iを満たす分布である。ラプラシアンのような対角的な演算子には効果的であるが、混合偏微分を含む非対角的な演算子には適用が難しい。

[0007]
有限差分法や確率的平滑化などの自動微分を完全に回避する手法も提案されているが、これらの方法は自動微分と比較して精度が離散化の選択に大きく依存するという欠点がある。有限差分法は、微分を差分近似で置き換えるため、離散化誤差が生じる。確率的平滑化は、ガウスランダム変数に対する期待値をアンザッツとして使用し、その導関数はシュタイン恒等式を通じて別のガウスランダム変数の期待値として表現できるが、やはり近似誤差が問題となる。

[0008]
テイラーモード自動微分は高次の自動微分手法として知られているが、従来はスカラー関数に対してのみ定義されており、多変数関数の高次微分テンソルの計算には直接適用できなかった。テイラーモード自動微分は、関数のテイラー展開がプリミティブによってどのように変化するかを決定することで高次の自動微分を定義する。具体的には、k次のプッシュフォワードd^kFは以下のように定義される：

d^k F(J^k_g(t)) = J^k_(F∘g)(t) = ([F∘g](t), ∂/∂t[F∘g](t), ..., ∂^k/∂t^k[F∘g](t))
= (F(a), ∂F(a)(v^(1)), ∂^2F(a)(v^(1), v^(2)), ..., ∂^kF(a)(v^(1), ..., v^(k)))

ここで、J^k_g(t) := (g(t), g'(t), g''(t), ..., g^(k)(t)) = (a, v^(1), v^(2), ..., v^(k))はg(t)のk-ジェットである。しかし、この方法は直接的には多変数関数の高次微分テンソルの特定の要素や縮約を計算するためには使用できなかった。

[0009]
また、高次元問題に対応するためのメモリ最適化技術も限られていた。例えば、ニューラルネットワークの第一層のパラメータ数は入力次元に比例して増加するため、超高次元問題ではメモリのボトルネックとなる。従来の方法では、このようなメモリの問題に対する効果的な解決策は提供されていなかった。

[0010]
さらに、物理情報ニューラルネットワーク（PINN）の訓練において、勾配正則化（gradient-enhanced PINN、gPINN）は解の精度を向上させる効果的な手法として知られているが、高次元問題に対しては計算コストが非常に高くなるため、実用的ではなかった。gPINNは残差の勾配がゼロベクトルに近くなるように正則化するが、残差の勾配の計算には高次の微分が必要となり、従来の方法では効率的に計算できなかった。

3. 発明が解決しようとする課題

[0011]
本発明は、以下の課題を解決することを目的とする：

[0012]
1. 入力次元dと微分の次数kの両方に関するスケーリング問題を同時に解決すること：従来の方法では、入力次元dに関してO(d^k)、微分の次数kに関してO(2^(k-1)L)のスケーリングを示していたが、本発明ではこれらの指数関数的なスケーリングを多項式的なスケーリングに削減することを目指す。

[0013]
2. 任意の高次微分演算子に対応可能な汎用的な計算フレームワークを提供すること：従来の方法（SDGDやHTE）は主にラプラシアンなどの特定の演算子に限定されていたが、本発明では任意の次数の任意の微分演算子（混合偏微分を含む）に対応できるフレームワークを提供する。

[0014]
3. 計算効率とメモリ効率を大幅に向上させること：従来の方法では、高次元高次微分を含む問題の計算には膨大な計算リソースとメモリが必要だったが、本発明では計算量とメモリ要件を大幅に削減する。

[0015]
4. 従来は計算が不可能だった超高次元問題（100万次元以上）を解決可能にすること：従来の方法では、数千次元を超える問題は実質的に解くことができなかったが、本発明では100万次元以上の問題を現実的な時間とリソースで解決できるようにする。

[0016]
5. 物理情報ニューラルネットワーク（PINN）や高次元偏微分方程式（PDE）の解法を高速化すること：従来のPINNは高次元問題に対して計算コストが高く、実用的ではなかったが、本発明ではPINNの訓練を大幅に高速化し、より複雑な問題に適用可能にする。

[0017]
6. 任意の混合偏微分を含む微分演算子に対しても効率的な計算を可能にすること：従来の方法では混合偏微分を含む演算子の計算は特に困難だったが、本発明では任意の混合偏微分を効率的に計算できるようにする。

[0018]
7. 並列計算やハードウェア加速を最大限に活用できる計算フレームワークを提供すること：従来の方法では並列化が困難だったが、本発明では計算を並列化し、GPUなどのハードウェアアクセラレータを最大限に活用できるようにする。

[0019]
8. 勾配正則化を含む高度なPINN手法を高次元問題に適用可能にすること：従来のgPINNは高次元問題に対して計算コストが非常に高く、実用的ではなかったが、本発明ではgPINNを高次元問題に効率的に適用できるようにする。

[0020]
9. 適応的なサンプリング戦略により、確率的推定器の分散を最小化すること：従来の確率的手法では推定の分散が大きいことが問題だったが、本発明では分散を最小化するサンプリング戦略を提供する。

[0021]
10. 様々な科学・工学分野における大規模シミュレーションを可能にすること：従来は計算の制約により限られていた大規模シミュレーションを、本発明により実現可能にする。

4. 課題を解決するための手段

[0022]
本発明は、確率的テンソル縮約ネットワーク（STCN）と呼ばれる新しい計算フレームワークを提供する。STCNは、任意の微分演算子をテンソル縮約として定式化し、テイラーモード自動微分を用いてランダムジェットの前方伝播を効率的に計算することで、高次元微分演算子の確率的推定を行う。

[0023]
具体的には、k次の微分演算子Lは以下のようにテンソル内積として表現できる：

$$Lu(x) = \sum_{(\alpha \in I(L))} C_\alpha D^\alpha u(x) = \sum_{(d_1,...,d_k)} D^k_u(a)_{(d_1,...,d_k)} C_{(d_1,...,d_k)}(L) = D^k_u(a) \cdot C(L)$$

ここで、$d_i \in [1, d]$、$i \in [1, k]$はi番目の軸上のテンソルインデックス、C(L)はLの係数テンソルである。例えば、ラプラシアン演算子∇^2の係数テンソルはd次元の単位行列Iである。

[0024]
STCNは、以下の条件を満たす入力l-ジェットJ^l_g上の分布pを構築する：

$$E[v^{(v_1)}_{(d_1)} \cdots v^{(v_k)}_{(d_k)}] = C_{(d_1,...,d_k)}(L)$$

この条件が満たされると、ランダムジェットの前方伝播の期待値は微分演算子の値と一致する：

$$E[\partial^l u(J^l_g)] = D^k u(x) \cdot C(L) = Lu(x)$$

[0025]
STCNは以下のモジュールから構成される：

[0026]
1. 演算子分析モジュール：微分演算子の構造を分析し、係数テンソルを構築する。このモジュールは、微分演算子の数学的表現を解析し、対応する係数テンソルC(L)を構築する。また、演算子の特性（対称性、スパース性など）を分析し、後続のモジュールのための情報を提供する。さらに、演算子の加法的分解を実行し、各項を個別に処理することもできる。

[0027]
2. ジェット構造最適化モジュール：効率的な計算のための最適なジェット構造を決定する。このモジュールは、微分演算子の次数と非対角性（混合偏微分の有無と構造）に基づいて、必要な最小のジェット次数lと疎密パターンを決定する。また、疎なジェット構造と密なジェット構造の間で自動的に選択する機能も提供する。

[0028]
3. サンプリング戦略生成モジュール：確率的推定のための最適なサンプリング戦略を構築する。このモジュールは、係数テンソルのスペクトル特性を分析することにより、確率的推定器の分散を最小化するサンプリング分布を生成する。また、ラデマッハー分布、ガウス分布、均一分布などの様々な分布から最適なものを選択する機能も提供する。さらに、訓練過程で確率的推定器の分散を監視し、サンプリング戦略を動的に調整することもできる。

[0029]
4. テンソル縮約エンジン：ハードウェア加速を用いてテンソル縮約を効率的に計算する。このエンジンは、サンプリング戦略に従ってランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる。また、複数のランダムジェットの前方伝播を並列に計算する機能や、ニューラルネットワークの第一層において重み共有技術を適用することによりパラメータ数を削減する機能も提供する。さらに、勾配正則化を含む物理情報ニューラルネットワークの訓練を加速するために、勾配残差項の確率的推定を計算する機能も備える。

[0030]
5. 階層的計算スキーム：超高次元問題に対応するため、高次元領域を管理可能な次元の部分領域に分解する。このスキームは、高次元領域をm個の部分領域（各部分領域はd/m次元）に分解し、各部分領域でSTCNを適用する。これにより、メモリ要件をさらに削減し、より大規模な問題に対応できる。

[0031]
本発明の主な特徴の一つは、スカラー関数に対するテイラーモード自動微分を多変数関数の任意の微分テンソル縮約として解釈し、これを利用して高次元微分演算子の効率的な計算を実現することである。具体的には、スカラー関数に対するk次のフレシェ微分∂^k Fは、適切に構成された入力ジェットJ^k_gを用いることで、多変数関数の微分テンソルD^k_Fの任意の縮約を計算できることを示す：

$$\partial^k F(a)(v^{(1)}, \ldots, v^{(k)}) = \partial^k/\partial t^k [F \circ g](t) = D^k_F(a) \cdot \otimes^k_{i=1} v^{(v_i)}$$

ここで、v^(v_i)は入力ジェットの接線ベクトルであり、v_i ∈ [1, k]である。

[0032]
また、本発明は微分演算子の構造に基づいて最適なジェット構造（次数と疎密パターン）を自動的に決定する方法を提供する。対角的な演算子（混合偏微分を含まない）の場合、l = kで十分である。最大限に非対角的な演算子の場合、必要な最小のlは(1+k)k/2となる。

[0033]
さらに、本発明は係数テンソルのスペクトル特性を分析することにより、確率的推定器の分散を最小化するサンプリング戦略を構築する方法を提供する。疎なジェットの場合、分布pは標準基底と零ベクトルのみからなるジェットに対する離散分布として定義される：

$$p(\otimes^k_{i=1} e_{d_i}) = C_{d_1,\ldots,d_k}/Z,\ d_1,\ldots,d_k \in I(L)$$

ここで、Zは正規化因子である。密なジェットの場合、係数テンソルの固有値を共分散とする多変量ガウス分布などが使用される。

[0034]
本発明は、テイラーモード自動微分の計算複雑性O(k^2dL)とメモリ要件O(kd)を活用し、従来の繰り返し適用される後方モード自動微分の計算複雑性O(2^k(dh + (L-1)h^2))とメモリ要件O(2^(k-1)(d + (L-1)h))と比較して大幅に効率的な計算を実現する。

[0035]
また、本発明は入力次元が非常に高い場合のメモリ最適化のために、ニューラルネットワークの第一層での重み共有技術を提供する。具体的には、入力ベクトルに対して1次元畳み込みを適用し、その出力をMLPに渡す。畳み込みのフィルタサイズをBとすると、パラメータ数は約1/B倍に削減される。

[0036]
さらに、本発明は勾配正則化を含む物理情報ニューラルネットワーク（gPINN）の効率的な実装を提供する。gPINNは残差の勾配がゼロベクトルに近くなるように正則化するが、本発明では残差の勾配の計算を効率的に行うことができる。

5. 発明の効果

[0037]
本発明の主な効果は以下の通りである：

[0038]
1. 計算効率：従来の方法と比較して最大1000倍の高速化と30倍のメモリ削減を実現する。これは、テイラーモード自動微分の効率的な利用と確率的サンプリングの組み合わせによるものである。従来の方法では、微分の次数kに関して計算複雑性がO(2^(k-1)L)、メモリ要件がO(2^(k-1)(d + (L-1)h))であったのに対し、本発明ではそれぞれO(k^2dL)、O(kd)に削減される。これにより、従来は計算が不可能だった高次微分を含む問題も効率的に解くことができる。

[0039]

2. スケーラビリティ：100万次元以上の問題を扱うことが可能となる。従来の方法では、入力次元dに関してO(d^k)のスケーリングを示していたが、本発明では確率的サンプリングにより次元に関する多項式的なスケーリングを除去する。また、階層的計算スキームにより、さらに大規模な問題（1000万次元以上）にも対応できる。これにより、従来は計算リソースの制約により解くことができなかった超高次元問題も解決可能となる。

[0040]
3. 汎用性：任意の次数の任意の微分演算子に適用可能である。従来の方法（SDGDやHTE）は主にラプラシアンなどの特定の演算子に限定されていたが、本発明は任意の混合偏微分を含む一般的な微分演算子に対応できる。これにより、より複雑な物理現象を記述する偏微分方程式や最適化問題に対応できる。

[0041]
4. 使いやすさ：最適な計算戦略の自動決定により、ユーザーは微分演算子の詳細な構造を理解していなくても効率的な計算が可能となる。ジェット構造最適化モジュールは、微分演算子の構造に基づいて最適なジェット次数と疎密パターンを自動的に決定し、サンプリング戦略生成モジュールは確率的推定器の分散を最小化するサンプリング戦略を自動的に構築する。これにより、専門知識がなくても高度な計算を実行できる。

[0042]
5. 並列性：計算が並列化可能であり、GPUなどのハードウェア加速を最大限に活用できる。従来の方法では、微分の次数kが増加すると順次計算の数が増加していたが、本発明では順次計算の数は増加しない。また、複数のランダムジェットの前方伝播を並列に計算できるため、マルチコアCPUやGPUなどのハードウェアアクセラレータを効率的に活用できる。さらに、分散計算機構により、複数のハードウェアアクセラレータ間で計算を分散させることも可能である。

[0043]
6. メモリ効率：微分テンソル全体を計算・保存する必要がなく、必要な縮約のみを効率的に計算できる。これにより、大規模な問題でもメモリ制約内で計算が可能となる。また、ニューラルネットワークの第一層での重み共有技術により、パラメータ数を大幅に削減し、さらにメモリ効率を向上させることができる。

[0044]
7. 精度：自動微分に基づいているため、有限差分法や確率的平滑化などの近似手法と比較して高精度な計算が可能である。テイラーモード自動微分は、解析的な導関数に基づいているため、離散化誤差や近似誤差が生じない。また、サンプリング戦略生成モジュールにより、確率的推定器の分散を最小化することで、少ないサンプル数でも高精度な推定が可能となる。

[0045]
8. 適応性：訓練過程で確率的推定器の分散を監視し、サンプリング戦略を動的に調整することができる。これにより、訓練の初期段階では計算効率を優先し、後期段階では精度を優先するなど、状況に応じた最適な戦略を採用できる。また、適応的バッチサイズ調整機構により、計算効率と推定精度のバランスを最適化することができる。

[0046]
9. 拡張性：様々な微分演算子や問題設定に対応できる柔軟なフレームワークを提供する。演算子分析モジュールは、微分演算子の加法的分解を実行し、各項を個別に処理することができるため、複雑な演算子にも対応できる。また、混合精度計算や分散計算などの最適化技術を組み合わせることで、さらなる性能向上が可能である。

[0047]
10. 応用範囲：科学計算、機械学習、物理シミュレーションなど多くの分野で広く応用できる。特に、量子多体系シミュレーション、計算流体力学、金融モデリング、分子動力学シミュレーションなど、高次元微分演算子を含む問題に対して大きな効果を発揮する。これにより、従来は計算の制約により限られていた大規模シミュレーションや複雑な現象のモデリングが可能となる。

6. 発明を実施するための形態

[0048]
以下、本発明の実施形態について詳細に説明する。

1. 確率的テンソル縮約ネットワークの基本構造

[0049]
確率的テンソル縮約ネットワーク（STCN）は、微分演算子をテンソル縮約として表現し、テイラーモード自動微分を用いて効率的に計算するためのフレームワークである。STCNの基本的なワークフローは以下の通りである：
(1) 微分演算子を係数テンソルとして表現する
(2) 係数テンソルに基づいて最適なジェット構造を決定する
(3) ジェット構造と係数テンソルに基づいて確率的サンプリング戦略を構築する
(4) サンプリング戦略に従ってランダムジェットをサンプリングする
(5) テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる
(6) 前方伝播の結果を組み合わせて微分演算子の推定値を得る

[0050]
本発明の核心は、スカラー関数に対するテイラーモード自動微分が多変数関数の微分テンソルの任意の縮約を計算できるという洞察にある。具体的には、k次のフレシェ微分∂^k Fは、適切に構成された入力ジェットJ^k_gを用いることで、以下のように表現できる：

$$\partial^k F(a)(v^{(1)}, \ldots, v^{(k)}) = \partial^k/\partial t^k [F \circ g](t) = D^k_F(a) \cdot \otimes^k_{i=1} v^{(v_i)}$$

ここで、v^(v_i)は入力ジェットの接線ベクトルであり、v_i ∈ [1, k]である。この式は、適切な入力ジェットを構成することで、微分テンソルD^k_Fの任意の縮約を計算できることを示している。

[0051]

例えば、ヘッセ行列の二次形式は、2次のフレシェ微分∂^2 Fを用いて以下のように計算できる：

∂^2 F(a)(v^(1), v^(2)) = ∂^2/∂t^2 [F ∘ g](t) = D F(a) v^(2) + D^2_F(a)_(d',d_1,d_2) v^(1)_(d_1) v^(1)_(d_2)

ここで、v^(2) = 0と設定することで、ヘッセ行列の二次形式D^2_F(a)_(d',d_1,d_2) v^(1)_(d_1) v^(1)_(d_2)のみを計算できる。

[0052]
一般に、十分に大きなl ≥ kに対して、k ≤ l' ≤ lとなるl'が存在し、∂^(l') F(J^(l')_g) = D^k_F(a) · ⊗^k_(i=1) v^(v_i)となるようなJ^l_gを見つけることができる。ここで、v_i ∈ [1, k]であり、一部の接線v^(v_i)を零ベクトルに設定することで実現できる。つまり、任意の微分テンソル縮約は、高次のフレシェ微分に含まれており、テイラーモード自動微分を通じて効率的に評価できる。

[0053]
必要なlの大きさは、演算子の非対角性に依存する。演算子が対角的（混合偏微分を含まない）な場合、l = kで十分である。演算子が最大限に非対角的、つまり微分される次元がすべて異なる偏微分の場合、必要な最小のlは(1+k)k/2となる。

2. 演算子分析モジュール

[0054]
演算子分析モジュールは、微分演算子の構造を分析し、係数テンソルを構築する。微分演算子Lは以下のように表現される：

L = ∑_(α∈I(L)) C_α D^α

ここで、I(L)は演算子Lに含まれる項のインデックス集合、C_αは各項の係数、D^αは偏微分演算子である。D^α = ∂^|α|/(∂x^(α_1)_1 ... ∂x^(α_d)_d)であり、α = (α_1, α_2, ..., α_d)は多重インデックスである。

[0055]
スカラー関数u: R^d → Rに対して、k次の微分演算子Lは以下のようにテンソル内積として表現できる：

Lu(a) = ∑_(α∈I(L)) C_α D^α u(a) = ∑_(d_1,...,d_k) D^k_u(a)_(d_1,...,d_k) C_(d_1,...,d_k)(L) = D^k_u(a) · C(L)

ここで、d_i ∈ [1, d]、i ∈ [1, k]はi番目の軸上のテンソルインデックスであり、C(L)はLの係数テンソルである。

[0056]
例えば、ラプラシアン演算子∇^2の係数テンソルはd次元の単位行列Iである。より複雑な演算子は、f(x, u, D^(k_1) u, ..., D^(k_n) u)の形式で構築できる。ここで、fは任意の関数である。

[0057]
演算子分析モジュールは、微分演算子の数学的表現を解析し、対応する係数テンソルC(L)を構築する。このモジュールは以下の機能を提供する：

(1) 微分演算子の次数kを特定する：演算子に含まれる最高次の微分の次数を特定する。例えば、ラプラシアン演算子∇^2は2次の微分演算子である。

(2) 演算子に含まれる各項の係数C_αを抽出する：演算子の数学的表現から各項の係数を抽出する。例えば、ラプラシアン演算子∇^2 = ∑^d_(i=1) ∂^2/∂x^2_iの場合、すべての対角項の係数は1である。

(3) 係数テンソルC(L)を構築する：抽出した係数からk次の係数テンソルを構築する。例えば、ラプラシアン演算子の場合、係数テンソルはd次元の単位行列Iとなる。

(4) 演算子の特性（対称性、スパース性など）を分析する：係数テンソルの特性を分析し、後続のモジュールのための情報を提供する。例えば、対称性を利用して計算を効率化したり、スパース性を利用してメモリ使用量を削減したりできる。

(5) 演算子の加法的分解を実行する：複雑な演算子を加法的に分解し、各項を個別に処理することで計算を効率化する。例えば、複合演算子L = L_1 + L_2 + ... + L_nの場合、各項L_iを個別に処理し、結果を合算することができる。

[0058]
例えば、2次の放物型偏微分方程式の場合：

∂u(x,t)/∂t + 1/2 tr(σσ^T(x,t) ∂^2u(x,t)/∂x^2) + ∇u(x,t)·μ(x,t) + f(t,x,u(x,t),σ^T(x,t)∇u(x,t)) = 0

この方程式に含まれる2次微分項の係数テンソルは1/2 σσ^Tである。演算子分析モジュールは、この項を特定し、係数テンソルを構築する。

[0059]
また、演算子分析モジュールは、非線形演算子を線形化して処理することもできる。例えば、非線形項u(x)^3を含むアレン・カーン方程式の場合、この項を線形項と見なして処理し、係数を適宜調整することができる。

[0060]
さらに、演算子分析モジュールは、微分演算子の構造に基づいて最適な処理戦略を決定するためのヒントを提供する。例えば、演算子が疎である場合、疎なジェット構造を使用することが推奨される。また、演算子が密である場合、密なジェット構造が適している可能性がある。

3. ジェット構造最適化モジュール

[0061]
ジェット構造最適化モジュールは、微分演算子の次数と非対角性に基づいて最適なジェット構造を決定する。具体的には、以下のステップを実行する：

(1) 微分演算子の次数kを特定する：演算子に含まれる最高次の微分の次数を特定する。

(2) 演算子の非対角性（混合偏微分の有無と構造）を分析する：演算子に含まれる混合偏微分の構造を分析する。例えば、演算子が対角的（混合偏微分を含まない）か、部分的に非対角的か、完全に非対角的かを判断する。

(3) 必要な最小のジェット次数lを決定する：演算子の次数と非対角性に基づいて、必要な最小のジェット次数lを決定する。対角的な演算子の場合、l = kで十分である。最大限に非対角的な演算子の場合、必要な最小のlは(1+k)k/2となる。

(4) ジェットの疎密パターンを決定する：演算子の構造に基づいて、ジェットの疎密パターン（どの接線ベクトルが非ゼロか）を決定する。

[0062]
ジェット構造は、入力l-ジェットJ^l_g(t) = (g(t), g'(t), g''(t), ..., g^(l)(t)) = (a, v^(1), v^(2), ..., v^(l))によって特徴付けられる。ここで、v^(j)はg(t)のj次の接線ベクトルである。

[0063]
例えば、混合偏微分∂^3u/(∂x^2_i ∂x_j)を計算する場合、以下のようなジェット構造が考えられる：

(1) l = 4の場合：∂^4u(x)(e_i, e_j, 0, 0) - ∂^4u(x)(e_i, 0, 0, 0) - 3∂^2u(x)(e_j, 0))/6
(2) l = 5の場合：(∂^5u(x)(e_i, 0, e_j, 0, 0) - ∂^5u(x)(e_i, 0, 0, 0, 0))/10
(3) l = 7の場合：∂^7u(x)(0, e_i, e_j, 0, 0, 0, 0)/105

[0064]
一般的に、より高次のジェットを使用すると、より少ない前方伝播で混合偏微分を計算できるが、計算コストが増加する可能性がある。ジェット構造最適化モジュールは、計算効率とメモリ効率のバランスを考慮して最適なジェット構造を決定する。

[0065]
具体的には、任意の混合偏微分∂^(∑^T_(j) q_(i_j))u/(∂x^(q_(i_1))_(i_1) ... ∂x^(q_(i_T))_(i_T))を計算するために、以下の条件を満たすジェット次数kと疎密パターンJ = (j_1, ..., j_T)を見つける：

(1) ジェット次数k ∈ N
(2) 疎密パターンJ = (j_1, ..., j_T)（v^(j) = 0, j ∉ J、j_t < k, ∀t ∈ [1, T]）

[0066]
p_j = 0（j ∉ J）またはp_j = q_(i_t)（j = j_t）と設定すると、(p_1, p_2, ..., p_k) ∈ N^kはkの分割となる。ここで、kの分割とは、∑^k_(i=1) i·p_i = kを満たすタプル(p_1, ..., p_k) ∈ N^kである。

[0067]
さらに、他の分割の疎密パターンが考慮中の分割の疎密パターンのサブセットでない場合、余分な項を除去する必要がない。これにより、一回の前方伝播で目的の混合偏微分を計算できる。

[0068]
ジェット構造最適化モジュールは、微分演算子の構造に基づいて最適なジェット次数と疎密パターンを自動的に決定するヒューリスティックアルゴリズムを実装する。このアルゴリズムは、以下のステップで構成される：

(1) 微分演算子の次数kと非対角性を分析する
(2) 必要な最小のジェット次数lの初期推定値を設定する
(3) lに対して可能なすべての疎密パターンを列挙する
(4) 各疎密パターンに対して、目的の微分テンソル縮約を計算できるかどうかを確認する
(5) 目的の縮約を計算できる疎密パターンの中から、計算効率とメモリ効率のバランスが最適なものを選択する
(6) 適切な疎密パターンが見つからない場合、lを増加させて再試行する

[0069]
また、ジェット構造最適化モジュールは、疎なジェット構造と密なジェット構造の間で自動的に選択する機能も提供する。疎なジェット構造は標準基底と零ベクトルのみからなるジェットを使用し、任意の微分演算子に適用可能である。一方、密なジェット構造はガウス分布などの連続分布からサンプリングされたベクトルを使用し、特定の演算子（主に2次の演算子）に対してより効率的である場合がある。

[0070]
例えば、ラプラシアン演算子∇^2の場合、HTEに基づく密なジェット構造を使用することで、分散を最小化できる。一方、4次の対角演算子∑^d_(i=1) ∂^4u/∂x^4_iの場合、密なジェット構造を構築できないことが証明されているため、疎なジェット構造を使用する必要がある。

[0071]
ジェット構造最適化モジュールは、演算子の構造と計算リソースの制約に基づいて、最適なジェット構造を自動的に決定する。これにより、ユーザーは微分演算子の詳細な構造を理解していなくても、効率的な計算が可能となる。

4. サンプリング戦略生成モジュール

[0072]
サンプリング戦略生成モジュールは、ジェット構造と係数テンソルに基づいて確率的サンプリング戦略を構築する。このモジュールは以下の機能を提供する：

(1) 係数テンソルのスペクトル特性を分析する：係数テンソルの固有値や特異値を分析し、その構造を理解する。

(2) 確率的推定器の分散を最小化するサンプリング分布を設計する：係数テンソルの構造に基づいて、確率的推定器の分散を最小化するサンプリング分布を設計する。

(3) 疎なジェットと密なジェットのどちらが適切かを決定する：演算子の構造と計算リソースの制約に基づいて、疎なジェットと密なジェットのどちらが適切かを決定する。

[0073]
任意の微分テンソル縮約D^k_u(a)·C(L)は、適切な分布からのランダム縮約によって推定できる：

E[D^k_u(a)_(d_1,...,d_k) v^(v_1)_(d_1) ... v^(v_k)_(d_k)] = D^k_u(a)_(d_1,...,d_k) E[v^(v_1)_(d_1) ... v^(v_k)_(d_k)] = D^k_u(a) · E[⊗^k_(i=1) v^(v_i)]

[0074]
この式は、E[v^(v_1)_(d_1) ... v^(v_k)_(d_k)] = C_(d_1,...,d_k)(L)を満たす分布pからのランダムジェットJ^l_gを用いることで、Lu(a) = D^k_u(a) · C(L)の不偏推定量を得られることを示している。

[0075]
疎なジェットの場合、分布pは標準基底と零ベクトルのみからなるジェットに対する離散分布として定義される：

p(⊗^k_(i=1) e_(d_i)) = C_(d_1,...,d_k)/Z, d_1,...,d_k ∈ I(L)

ここで、Zは正規化因子である。

[0076]
例えば、ラプラシアン演算子∇^2の場合、STCNによる確率的推定は以下のように行われる：

∇^2_J u_θ(a) = d/|J| ∑_(j∈J) ∂^2 u_θ(a)/∂x^2_j = d/|J| ∑_(j∈J) ∂^2 u_θ(a)(e_j, 0) = d/|J| ∑_(j∈J) d^2 u_θ(a, e_j, 0)[2]

ここで、Jはサンプリングされたインデックス集合、|J|はバッチサイズ、e_jは標準基底ベクトル、0は零ベクトル、添字[2]は出力ジェットから2次の接線を取ることを意味する。

[0077]
密なジェットの場合、係数テンソルの固有値を共分散とする多変量ガウス分布などが使用される。例えば、2次の微分演算子Dの場合、係数テンソルCの固有分解C'' = 1/2(C + C^T) + λI = UΣU^Tを用いて、以下のようなSTCNを構築できる：

E_(v〜N(0,Σ))[∂^2u(a)(Uv, 0)] - λE_(v〜N(0,I))[∂^2u(a)(v, 0)] = D^2_u(a)·[C'' - λI] = D^2_u(a)·C

[0078]
ただし、密なジェットを用いたSTCNは2次以上の微分演算子に対して常に構築できるわけではない。例えば、4次の対角演算子∑^d_(i=1) ∂^4u/∂x^4_iに対しては、密なジェットを用いたSTCNを構築できないことが証明されている。

[0079]
サンプリング戦略生成モジュールは、演算子の構造に基づいて最適なサンプリング戦略（疎なジェットまたは密なジェット）を自動的に選択する。一般的には、疎なジェットが普遍的に適用可能であるが、特定の演算子に対しては密なジェットがより低い分散を示す場合がある。

[0080]
また、サンプリング戦略生成モジュールは、ラデマッハー分布、ガウス分布、均一分布などの様々な分布からサンプリング分布を選択する機能も提供する。例えば、HTEの場合、等方的な分布（E[vv^T] = I）が必要であり、ラデマッハー分布（各要素が±1のベクトル）がHTEの分散を最小化することが知られている。

[0081]
さらに、サンプリング戦略生成モジュールは、訓練過程で確率的推定器の分散を監視し、サンプリング戦略を動的に調整することもできる。例えば、訓練の初期段階では計算効率を優先して小さなバッチサイズを使用し、後期段階では精度を優先して大きなバッチサイズを使用するなど、状況に応じた最適な戦略を採用できる。

[0082]
サンプリング戦略生成モジュールは、以下のアルゴリズムを実装する：

(1) 係数テンソルC(L)のスペクトル分解を計算する
(2) 係数テンソルの構造（対称性、スパース性など）を分析する
(3) 演算子の次数kに基づいて、密なジェットが構築可能かどうかを判断する
(4) 密なジェットが構築可能な場合、係数テンソルの固有値に基づいてサンプリング分布を設計する
(5) 密なジェットが構築できない場合、または疎なジェットの方が効率的な場合、疎なジェットのサンプリング分布を設計する
(6) サンプリング分布の分散を分析し、必要に応じて調整する
(7) 訓練過程で分散を監視し、サンプリング戦略を動的に調整するメカニズムを設定する

5. テンソル縮約エンジン

[0083]
テンソル縮約エンジンは、サンプリング戦略に従ってランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる。このエンジンは以下の特徴を持つ：

(1) 複数のランダムジェットの前方伝播を並列に計算する
(2) ハードウェア加速（GPU等）を活用して計算を高速化する
(3) メモリ使用量を最適化する
(4) ニューラルネットワークの第一層において重み共有技術を適用することによりパラメータ数を削減する
(5) 勾配正則化を含む物理情報ニューラルネットワークの訓練を加速するために、勾配残差項の確率的推定を計算する
(6) 適応的バッチサイズ調整機構を含み、計算効率と推定精度のバランスを最適化する
(7) 混合精度計算を実行し、前方伝播には低精度演算を、勾配蓄積には高精度演算を使用する
(8) 複数のハードウェアアクセラレータ間で計算を分散させる分散計算機構を含む

[0084]

テイラーモード自動微分は、高次の自動微分手法であり、関数の高次導関数を効率的に計算できる。具体的には、k次のプッシュフォワードd^kFは以下のように定義される：

d^k F(J^k_g(t)) = J^k_(F∘g)(t) = ([F∘g](t), ∂/∂t[F∘g](t), ..., ∂^k/∂t^k[F∘g](t))
= (F(a), ∂F(a)(v^(1)), ∂^2F(a)(v^(1), v^(2)), ..., ∂^kF(a)(v^(1), ..., v^(k)))

[0085]
このプッシュフォワードd^kFは、任意の合成関数F = F_L ∘ F_(L-1) ∘ ... ∘ F_1に対して、プリミティブのk次プッシュフォワードd^kF_iから計算できる：

d^k[F_2 ∘ F_1](J^k_g(t)) = J^k_(F_2∘F_1∘g)(t) = d^kF_2(J^k_(F_1∘g)(t)) = [d^kF_2 ∘ d^kF_1](J^k_g(t))

[0086]
テイラーモード自動微分の計算複雑性は、k次のプッシュフォワードに対してO(k^2dL)であり、メモリ要件はO(kd)である。これは、従来の繰り返し適用される後方モード自動微分の計算複雑性O(2^k(dh + (L-1)h^2))とメモリ要件O(2^(k-1)(d + (L-1)h))と比較して大幅に効率的である。

[0087]
テンソル縮約エンジンは、サンプリングされたランダムジェットの前方伝播を並列に計算し、結果を組み合わせて微分演算子の推定値を得る。この並列計算により、GPUなどのハードウェア加速を最大限に活用できる。

[0088]
具体的には、テンソル縮約エンジンは以下のステップで動作する：

(1) サンプリング戦略に従ってランダムジェットJ^l_g(t) = (a, v^(1), v^(2), ..., v^(l))をサンプリングする
(2) テイラーモード自動微分を用いて、関数uを通じてランダムジェットを前方に伝播させ、出力ジェットJ^l_(u∘g)(t)を得る
(3) 出力ジェットから必要な成分（通常はl次の接線）を抽出し、微分演算子の推定値を計算する
(4) 複数のランダムジェットからの推定値を平均して、最終的な推定値を得る

[0089]
テンソル縮約エンジンは、複数のランダムジェットの前方伝播を並列に計算することで、計算効率を向上させる。これは、各ランダムジェットの前方伝播が独立しているため、並列化が容易であるという利点を活かしている。

[0090]
また、テンソル縮約エンジンは、ニューラルネットワークの第一層において重み共有技術を適用することにより、パラメータ数を削減する。具体的には、入力ベクトルに対して1次元畳み込みを適用し、その出力をMLPに渡す。畳み込みのフィルタサイズをBとすると、パラメータ数は約1/B倍に削減される。

[0091]
さらに、テンソル縮約エンジンは、勾配正則化を含む物理情報ニューラルネットワーク（gPINN）の訓練を加速するために、勾配残差項の確率的推定を計算する機能も提供する。gPINNは残差の勾配がゼロベクトルに近くなるように正則化するが、残差の勾配の計算には高次の微分が必要となる。テンソル縮約エンジンは、STCNを用いて残差の勾配を効率的に計算することができる。

[0092]
テンソル縮約エンジンは、適応的バッチサイズ調整機構も含んでおり、計算効率と推定精度のバランスを最適化することができる。具体的には、訓練の初期段階では計算効率を優先して小さなバッチサイズを使用し、後期段階では精度を優先して大きなバッチサイズを使用するなど、状況に応じた最適な戦略を採用できる。

[0093]
また、テンソル縮約エンジンは、混合精度計算を実行し、前方伝播には低精度演算（例：float16）を、勾配蓄積には高精度演算（例：float32）を使用することで、計算効率をさらに向上させることができる。

[0094]
さらに、テンソル縮約エンジンは、複数のハードウェアアクセラレータ間で計算を分散させる分散計算機構も含んでいる。これにより、より大規模な問題に対応することができる。

6. 階層的計算スキーム

[0095]
超高次元問題（100万次元以上）に対応するため、STCNは階層的計算スキームを提供する。このスキームは以下のステップで構成される：

(1) 高次元領域を管理可能な次元の部分領域に分解する
(2) 各部分領域で確率的テンソル縮約を計算する
(3) 部分領域の結果を組み合わせて全体の解を構築する

[0096]
具体的には、d次元の領域をm個の部分領域（各部分領域はd/m次元）に分解し、各部分領域でSTCNを適用する。これにより、メモリ要件をさらに削減し、より大規模な問題に対応できる。

[0097]
例えば、1000万次元の問題を解く場合、領域を10個の100万次元部分領域に分解し、各部分領域でSTCNを適用することで、全体の解を効率的に構築できる。

[0098]
階層的計算スキームは、以下のアルゴリズムを実装する：

(1) 入力次元dと利用可能なメモリに基づいて、適切な分割数mを決定する
(2) 入力領域を次元に沿ってm個の部分領域に分割する
(3) 各部分領域に対してSTCNを適用し、部分的な結果を得る

(4) 部分的な結果を組み合わせて全体の解を構築する

[0099]
この階層的アプローチにより、従来は計算が不可能だった超高次元問題も解決可能となる。

7. 重み共有技術

[0100]
入力次元が非常に高い場合、ニューラルネットワークの第一層のパラメータ数が入力次元に比例して増加し、メモリのボトルネックとなる。STCNは、第一層での重み共有技術を提供し、パラメータ数を大幅に削減する。

[0101]
具体的には、入力次元をdとし、MLPの隠れ層の次元をhとする。d >> hの場合、第一層の重みはd × h次元の行列となり、後続の層の重み行列はh × h次元となる。

[0102]
重み共有スキームを導入することで、第一層のパラメータの冗長性を削減できる。具体的には、MLPに入力する前に、入力ベクトルx_iに対して追加の1次元畳み込みを実行する。この1次元畳み込みのフィルタサイズをBとし（BはDを割り切れる値）、ストライドサイズをBとすると、畳み込み出力は重複せず、チャネル数は1に設定される。

[0103]
この重み共有スキームにより、パラメータ数は約1/B倍に削減される。フィルタのパラメータ数はB × 1であり、後続の全結合層の重み行列のサイズはd/B × Hとなる。したがって、第一層の総パラメータ数はd × hからd/B × h + Bに削減される。

[0104]
例えば、d = 10^6、h = 100の場合、第一層のパラメータ数はd × h = 100 × 10^6となる。ブロックサイズB = 100を使用すると、パラメータ数はd/B × h + B = 10^6 + 100に削減される。ブロックサイズB = 10の場合、パラメータ数はd/B × h + B = 10 × 10^6 + 10となる。つまり、より大きなブロックサイズBを使用することで、モデルパラメータ数を大幅に削減できる。

[0105]
重み共有技術は、以下のアルゴリズムを実装する：

(1) 入力次元dと隠れ層の次元hに基づいて、適切なブロックサイズBを決定する
(2) 1次元畳み込み層を構築し、フィルタサイズをB、ストライドサイズをB、チャネル数を1に設定する
(3) 畳み込み層の出力をMLPに渡す
(4) 順伝播時には、入力ベクトルを畳み込み層に通してから、MLPに渡す
(5) 逆伝播時には、MLPの勾配を畳み込み層に伝播させ、パラメータを更新する

[0106]
この重み共有技術により、超高次元問題に対してもメモリ効率の良いニューラルネットワークを構築することができる。

8. 勾配正則化を含む物理情報ニューラルネットワーク

[0107]
物理情報ニューラルネットワーク（PINN）は、ニューラルネットワークでパラメータ化されたアンザッツu_θ(x)を用いて偏微分方程式を解くための手法である。PINNは、以下の最適化問題の典型的な例である：

arg min_θ f(x, u_θ(x), D^(α_(1)) u_θ(x), ..., D^(α_(n)) u_θ(x))

[0108]
勾配正則化を含むPINN（gPINN）は、残差の勾配がゼロベクトルに近くなるようにPINNを正則化する手法である。これにより、解の精度が向上する。具体的には、PINN損失は以下の項で拡張される：

ℓ_(gPINN) ({x^(i)}^(N_r)_(i=1)) = 1/N_r ∑_i ∑^d_j (∂R(x^(i))/∂x_j)^2

ここで、R(x)はPDE残差である。

[0109]
高次元PDEに勾配正則化を適用することは計算コストが高いが、STCNを用いることで効率的に計算できる。例えば、分離不可能なAllen-Cahn方程式の残差の勾配は以下のように与えられる：

∂R(x)/∂x_j = ∂/∂x_j [∑_i ∂^2u(x)/∂x^2_i + u(x) - u^3(x) - f(x)]
= ∑^d_i=1 ∂^3u(x)/(∂x_j∂x^2_i) + ∂u(x)/∂x_j - 3u^2(x)∂u(x)/∂x_j - ∂f(x)/∂x_j

[0110]
STCNランダム化により、インデックスiを[1, d]からサンプリングして2次項∂^2u/∂x^2_iをランダム化できる。また、gPINNペナルティ項もサンプリングできる。具体的には、以下のジェットのプッシュフォワードを用いる：

J = d^7u(x, 0, e_i, e_j, 0, 0, 0, 0),
∂^3u(x)/(∂x_j∂x^2_i) = J[7]/105,
∂^2u(x)/∂x^2_i = J[4]/3

[0111]
このプッシュフォワードにより、ラプラシアン演算子のSTCNと混合偏微分を効率的に計算できる。これにより、元のgPINN損失のランダム化されたラプラシアンを持つ以下の上限を最小化することで、gPINN正則化損失を効率的に分散化できる：

ℓ̃_(gPINN) ({x^(i)}^(N_r)_(i=1), I, J)
= 1/N_r ∑_(j∈J) ∑_(i∈I) (∂^3u(x)/(∂x_j∂x^2_i) + ∂u(x)/∂x_j - 3u^2(x)∂u(x)/∂x_j - ∂f(x)/∂x_j)^2

≥ 1/N_r ∑_(j∈J) (∑_(i∈I) ∂^3u(x)/(∂x_j∂x^2_i) + ∂u(x)/∂x_j - 3u^2(x)∂u(x)/∂x_j - ∂f(x)/∂x_j)^2

ここで、Jはg-PINN項をサンプリングするための独立にサンプリングされたインデックス集合である。

[0112]
この手法を「分散化gPINN」と呼ぶ。分散化gPINNは、従来のgPINNと比較して計算コストを大幅に削減しつつ、同等の正則化効果を提供する。これにより、高次元問題に対してもgPINNを効率的に適用することができる。

9. STCNの実装と最適化

[0113]
STCNは、効率的な実装と最適化のための様々な技術を提供する。以下に、主要な実装の詳細を示す：

[0114]
1. ジェット構造の効率的な表現：
ジェット構造は、プライマルaと接線ベクトルv^(1), ..., v^(l)のタプルとして表現される。疎なジェットの場合、ほとんどの接線ベクトルが零ベクトルであるため、非ゼロの接線ベクトルとそのインデックスのみを保存することでメモリ使用量を削減できる。

[0115]
2. テイラーモード自動微分の効率的な実装：
テイラーモード自動微分は、プリミティブのk次プッシュフォワードd^kF_iを合成することで実装される。各プリミティブに対して、高次導関数の解析式を導出し、一般化された連鎖律（ファア・ディ・ブルーノの公式）を用いて合成する。

[0116]
3. バッチ処理と並列計算：
複数のランダムジェットの前方伝播を並列に計算することで、計算効率を向上させる。これは、GPUなどのハードウェア加速を最大限に活用するために重要である。

[0117]
4. メモリ最適化：
微分テンソル全体を計算・保存する代わりに、必要な縮約のみを計算することでメモリ使用量を削減する。また、重み共有技術を用いてニューラルネットワークのパラメータ数を削減する。

[0118]
5. サンプリング戦略の最適化：
確率的推定器の分散を最小化するサンプリング戦略を設計する。これには、係数テンソルのスペクトル特性の分析や、最適なサンプリング分布の選択が含まれる。

[0119]
6. 適応的バッチサイズ：
計算効率と推定精度のバランスを取るために、適応的なバッチサイズを使用する。具体的には、初期の訓練段階では小さなバッチサイズを使用し、訓練が進むにつれてバッチサイズを増加させる。

[0120]
7. 混合精度訓練：
計算効率をさらに向上させるために、混合精度訓練を適用する。具体的には、前方伝播には低精度（例：float16）を使用し、勾配の蓄積には高精度（例：float32）を使用する。

[0121]
8. 分散計算：
大規模な問題に対応するために、複数のハードウェアアクセラレータ間で計算を分散させる。これには、データ並列性やモデル並列性などの技術が含まれる。

[0122]
9. 動的計算グラフ最適化：
計算グラフを動的に最適化することで、不要な計算を削減し、計算効率を向上させる。これには、不要な演算の削除や、演算の融合などの技術が含まれる。

[0123]
10. キャッシュ最適化：
頻繁に使用される中間結果をキャッシュすることで、計算の重複を避け、計算効率を向上させる。これには、プッシュフォワードの結果や、サンプリングされたジェットなどが含まれる。

7. 実施例

[0124]
本発明の実施例として、以下の微分演算子に対するSTCNの適用方法を説明する。

実施例1：ラプラシアン演算子

[0125]
ラプラシアン演算子∇^2は、2次の微分演算子であり、以下のように定義される：

∇^2 u(x) = ∑^d_(i=1) ∂^2 u(x)/∂x^2_i

[0126]

STCNによるラプラシアン演算子の確率的推定は以下のように行われる：

∇^2_J u_θ(a) = d/|J| ∑_(j∈J) ∂^2 u_θ(a)/∂x^2_j = d/|J| ∑_(j∈J) ∂^2 u_θ(a)(e_j, 0) = d/|J| ∑_(j∈J) d^2 u_θ(a, e_j, 0)[2]

ここで、Jはサンプリングされたインデックス集合、e_jは標準基底ベクトル、0は零ベクトル、添字[2]は出力ジェットから2次の接線を取ることを意味する。

[0127]
この実装は、以下の式から導かれる：

∂^2 F(a)(v^(1), v^(2)) = ∂^2/∂t^2 [F ∘ g](t) = D F(a) v^(2) + D^2_F(a)_(d',d_1,d_2) v^(1)_(d_1) v^(1)_(d_2)

この式によれば、ヘッセ行列の二次形式は、v^(2) = 0、v^(1) = e_jと設定することで∂^2を通じて計算できる。

[0128]
ラプラシアン演算子に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールがラプラシアン演算子を解析し、係数テンソルが単位行列Iであることを特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、l = 2、v^(1) = e_j、v^(2) = 0）
(3) サンプリング戦略生成モジュールが確率的サンプリング戦略を構築する（この場合、インデックスjを[1, d]から均一にサンプリングする）
(4) テンソル縮約エンジンがランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる
(5) 前方伝播の結果から2次の接線を抽出し、スケーリング係数d/|J|を乗じてラプラシアン演算子の推定値を得る

[0129]
この方法により、従来の方法と比較して大幅な計算効率の向上とメモリ使用量の削減を実現できる。

実施例2：高次対角微分演算子

[0130]
k次の対角微分演算子L = ∑^d_(j=1) ∂^k/∂x^k_j に対するSTCNの適用は以下のように行われる：

L_J u_θ(a) = d/|J| ∑_(j∈J) ∂^k u_θ(a)/∂x^k_j = d/|J| ∑_(j∈J) ∂^k u_θ(a)(e_j, 0, ...)

[0131]
この実装は、ファア・ディ・ブルーノの公式から導かれる。ファア・ディ・ブルーノの公式によれば、1次の接線v^(1)を標準基底e_jに設定し、他のすべての接線v^(i)を零ベクトルに設定することで、目的の高次対角要素が得られる。

[0132]
高次対角微分演算子に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールが高次対角微分演算子を解析し、係数テンソルを特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、l = k、v^(1) = e_j、v^(i) = 0 for i > 1）
(3) サンプリング戦略生成モジュールが確率的サンプリング戦略を構築する（この場合、インデックスjを[1, d]から均一にサンプリングする）
(4) テンソル縮約エンジンがランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる
(5) 前方伝播の結果からk次の接線を抽出し、スケーリング係数d/|J|を乗じて高次対角微分演算子の推定値を得る

[0133]
この方法により、従来の方法では計算が困難だったk次の対角微分演算子も効率的に計算できる。

実施例3：2次放物型偏微分方程式

[0134]
2次放物型偏微分方程式は、以下の形式で表される：

∂u(x,t)/∂t + 1/2 tr(σσ^T(x,t) ∂^2 u(x,t)/∂x^2) + ∇u(x,t)·μ(x,t) + f(t,x,u(x,t),σ^T(x,t)∇u(x,t)) = 0

[0135]
この方程式に含まれる2次微分項は、変数変換により以下のように書き換えられる：

1/2 tr(σ(x,t)σ(x,t)^T ∂^2 u(x,t)/∂x^2) = 1/2 ∑^d_(i=1) ∂^2 u(x,t)(σ(x,t)e_i, 0)

[0136]
STCNは、上記の式に基づいてd項の和をサンプリングする：

1/2 tr(σ(x,t)σ(x,t)^T ∂^2 u(x,t)/∂x^2) ≈ d/(2|J|) ∑_(j∈J) ∂^2 u(x,t)(σ(x,t)e_j, 0)

[0137]
この変換は、以下の等式に基づいている：

1/2 tr(σ(x,t)σ(x,t)^T ∂^2 u(x,t)/∂x^2) = 1/2 tr(σ(x,t)^T ∂^2 u(x,t)/∂x^2 σ(x,t)) = 1/2 ∑^d_(i=0) (σ(x,t)^T ∂^2 u(x,t)/∂x^2 σ(x,t))_(i,i) = 1/2 ∑^d_(i=0) e^T_i σ(x,t)^T ∂^2 u(x,t)/∂x^2 σ(x,t) e_i = 1/2 ∑^d_(i=0) ∂^2 u((x,t), σ(x,t)e_i, 0^T)[3]

[0138]
2次放物型偏微分方程式に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールが2次放物型偏微分方程式を解析し、2次微分項の係数テンソルが1/2 σσ^Tであることを特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、l = 2、v^(1) = σ(x,t)e_j、v^(2) = 0）
(3) サンプリング戦略生成モジュールが確率的サンプリング戦略を構築する（この場合、インデックスjを[1, d]から均一にサンプリングする）
(4) テンソル縮約エンジンがランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる

(5) 前方伝播の結果から2次の接線を抽出し、スケーリング係数d/(2|J|)を乗じて2次微分項の推定値を得る

[0139]
この方法により、高次元の2次放物型偏微分方程式も効率的に解くことができる。

実施例4：2次元コルテベーグ・ド・フリース（KdV）方程式

[0140]
2次元KdV方程式は以下のように定義される：

u_(ty) + u_(xxxy) + 3(u_y u_x)_x - u_(xx) + 2u_(yy) = 0

[0141]
この方程式に含まれるすべての微分項は、以下のジェットのプッシュフォワードから計算できる：

J = d^13 u(x, v^(1), ..., v^(13)), v^(3) = e_x, v^(4) = e_y, v^(7) = e_t, v^(i) = 0, ∀i ∉ {3, 4, 7},
u_x = J[1], u_y = J[2], u_(xx) = J[4], u_(xy) = J[5]/35,
u_(yy) = J[6]/35, u_(ty) = J[9]/330, u_(xxxy) = J[11]/200200

[0142]
ここで、添字[i]はジェットからi次の接線を選択することを意味し、前置因子はファア・ディ・ブルーノの公式によって決定される。

[0143]
この場合、すべての項が1つのプッシュフォワードで計算できるため、ランダム化は必要ない。あるいは、これらの項は異なるジェットの低次のプッシュフォワードで計算することもできる：

J^(1) = d^9 u(x, 0, e_x, e_y, 0, ...), J^(2) = d^3 u(x, 0, e_y, e_t), J^(3) = d^3 u(x, 0, e_y, 0)

u_x = J^(1)[2], u_y = J^(1)[3], u_(xx) = J^(1)[4]/3, u_(xy) = J^(1)[5]/10, u_(yy) = J^(3)[2],
u_(yyy) = J^(3)[3], u_(xxxy) = (J^(1)[9] - 280u_(yyy))/840, u_(ty) = (J^(2)[3] - u_(yyy))/3

[0144]
入力次元dが高い場合、STCNによるランダム化は大幅な高速化をもたらす。

[0145]
2次元KdV方程式に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールが2次元KdV方程式を解析し、各項の構造を特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、l = 13、特定の接線ベクトルを標準基底に設定し、他を零ベクトルに設定する）
(3) テンソル縮約エンジンがジェットを構築し、テイラーモード自動微分を用いて関数を通じてジェットを前方に伝播させる
(4) 前方伝播の結果から必要な成分を抽出し、適切なスケーリング係数を適用して各項の値を得る

[0146]
この方法により、複雑な混合偏微分を含む方程式も効率的に解くことができる。

実施例5：2次元カドムツェフ・ペトビアシビリ（KP）方程式

[0147]
2次元KP方程式は以下のように定義される：

(u_t + 6u u_x + u_(xxx))_x + 3σ^2 u_(yy) = 0

[0148]
これは以下のように展開できる：

u_(tx) + 6u_x u_x + 6u u_(xx) + u_(xxxx) + 3σ^2 u_(yy) = 0

[0149]
すべての微分項は、5-ジェット、4-ジェット、および2-ジェットのプッシュフォワードで計算できる：

J^(1) := d^5 u(x, 0, e_t, e_x, 0, 0)
J^(2) := d^4 u(x, e_x, 0, 0, 0)
J^(3) := d^2 u(x, e_y, 0)

u_(tx) = J^(1)[5]/10,
u_x = J^(2)[1], u_(xx) = J^(2)[2], u_(xxxx) = J^(2)[4],
u_(yy) = J^(3)[2]

[0150]
2次元KP方程式に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールが2次元KP方程式を解析し、各項の構造を特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、複数のジェット構造を組み合わせる）
(3) テンソル縮約エンジンが各ジェットを構築し、テイラーモード自動微分を用いて関数を通じてジェットを前方に伝播させる
(4) 前方伝播の結果から必要な成分を抽出し、適切なスケーリング係数を適用して各項の値を得る

[0151]
この方法により、複雑な高次微分を含む方程式も効率的に解くことができる。

実施例6：勾配正則化を含む1次元コルテベーグ・ド・フリース（g-KdV）方程式

[0152]
1次元KdV方程式は以下のように定義される：

u_t + u u_x + α u_(xxx) = 0

[0153]
勾配正則化を含むPINN（gPINN）は、残差の勾配が零ベクトルに近くなるようにPINNを正則化する。具体的には、PINN損失は以下の項で拡張される：

ℓ_(gPINN) ({x^(i)}^(N_r)_(i=1)) = 1/N_r ∑_i ∑^d_j (∂R(x^(i))/∂x_j)^2

[0154]
残差の勾配を計算するには、以下のようにする：

R(x, t) := u_t + u u_x + α u_(xxx),
∇R(x, t) = [u_(tt) + u_t u_x + u u_(tx) + α u_(txxx), u_(tx) + u_x u_x + u u_(xx) + α u_(xxxx)]

[0155]
すべての微分項は、1つの2-ジェットと2つの7-ジェットのプッシュフォワードで計算できる：

J^(1) := d^7 u(x, e_x, 0, 0, 0, 0, 0)
J^(2) := d^7 u(x, e_x, 0, 0, e_t, 0, 0, 0)
J^(3) := d^2 u(x, e_t, 0)

u_x = J^(1)[1], u_(xx) = J^(1)[2], u_(xxx) = J^(1)[3], u_(xxxx) = J^(1)[4], u_(xxxxx) = J^(1)[5],
u_(txxx) = (J^(2)[7] - J^(1)[8])/35, u_(tx) = (J^(2)[5] - u_(xxxxx))/5, u_t = J^(2)[4] - u_(xxxx),
u_(tt) = J^(3)[2]

[0156]
g-KdV方程式に対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールがg-KdV方程式と残差の勾配を解析し、各項の構造を特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、複数のジェット構造を組み合わせる）
(3) テンソル縮約エンジンが各ジェットを構築し、テイラーモード自動微分を用いて関数を通じてジェットを前方に伝播させる
(4) 前方伝播の結果から必要な成分を抽出し、適切なスケーリング係数を適用して各項の値を得る
(5) 残差と残差の勾配を計算し、PINN損失とgPINN損失を組み合わせて最終的な損失を得る

[0157]
この方法により、勾配正則化を含むPINNも効率的に訓練することができる。

実施例7：分離不可能で効果的に高次元のPDE

[0158]
分離不可能で効果的に高次元のPDEは、d次元単位球B^d上で定義された非線形で分離不可能な高次元厳密解u_(exact)(x)を通じて定義される：

Lu(x) = f(x), x ∈ B^d
u(x) = 0, x ∈ ∂B^d

ここで、Lは線形/非線形演算子、f(x) = Lu_(exact)(x)である。零境界条件により、厳密解に関する情報が境界条件を通じて漏れることはない。

[0159]
以下の演算子を考える：
- ポアソン方程式：Lu(x) = ∇^2 u(x)
- アレン・カーン方程式：Lu(x) = ∇^2 u(x) + u(x) - u(x)^3
- サイン・ゴードン方程式：Lu(x) = ∇^2 u(x) + sin(u(x))

[0160]
厳密解として、以下のものを考える（すべてのc_i ～ N(0, 1)）：
- 二体相互作用：u_(exact)(x) = (1 - ||x||^2_2) ∑^(d-1)_(i=1) c_i sin(x_i + cos(x_(i+1)) + x_(i+1) cos(x_i))
- 三体相互作用：u_(exact)(x) = (1 - ||x||^2_2) ∑^(d-2)_(i=1) c_i exp(x_i x_(i+1) x_(i+2))

[0161]
これらの方程式に対してSTCNを適用することで、従来の方法と比較して大幅な高速化とメモリ削減を実現できる。

[0162]
分離不可能で効果的に高次元のPDEに対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールがPDEを解析し、微分演算子（この場合はラプラシアン）を特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（この場合、l = 2、v^(1) = e_j、v^(2) = 0）
(3) サンプリング戦略生成モジュールが確率的サンプリング戦略を構築する（この場合、インデックスjを[1, d]から均一にサンプリングする）
(4) テンソル縮約エンジンがランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる
(5) 前方伝播の結果から2次の接線を抽出し、スケーリング係数d/|J|を乗じてラプラシアン演算子の推定値を得る
(6) 非線形項（u(x) - u(x)^3やsin(u(x))など）を計算し、ラプラシアン演算子の推定値と組み合わせて残差を計算する
(7) 残差の平均二乗誤差を最小化するようにニューラルネットワークのパラメータを更新する

[0163]
この方法により、100万次元以上の高次元PDEも効率的に解くことができる。

実施例8：セミリニア放物型PDE

[0164]
セミリニア放物型PDEは、初期条件が指定された以下の形式で定義される：

∂u(x,t)/∂t = Lu(x,t), (x,t) ∈ R^d × [0,T]
u(x,t) = g(x), (x,t) ∈ R^d × {0}

ここで、g(x)は既知の解析的で時間に依存しない関数であり、初期条件を指定する。Tは終端時刻である。

[0165]
以下の演算子を考える：
- セミリニア熱方程式：Lu(x,t) = ∇^2 u(x,t) + (1 - u(x,t)^2)/(1 + u(x,t)^2)
- アレン・カーン方程式：Lu(x,t) = ∇^2 u(x,t) + u(x,t) - u(x,t)^3
- サイン・ゴードン方程式：Lu(x,t) = ∇^2 u(x,t) + sin(u(x,t))

[0166]
これらの方程式に対してSTCNを適用することで、高次元問題を効率的に解くことができる。

[0167]
セミリニア放物型PDEに対するSTCNの実装は以下のステップで構成される：

(1) 演算子分析モジュールがPDEを解析し、微分演算子（この場合はラプラシアンと時間微分）を特定する
(2) ジェット構造最適化モジュールが最適なジェット構造を決定する（ラプラシアンの場合、l = 2、v^(1) = e_j、v^(2) = 0）
(3) サンプリング戦略生成モジュールが確率的サンプリング戦略を構築する（この場合、インデックスjを[1, d]から均一にサンプリングする）
(4) テンソル縮約エンジンがランダムジェットをサンプリングし、テイラーモード自動微分を用いて関数を通じてランダムジェットを前方に伝播させる
(5) 前方伝播の結果から2次の接線を抽出し、スケーリング係数d/|J|を乗じてラプラシアン演算子の推定値を得る
(6) 時間微分∂u(x,t)/∂tを計算し、ラプラシアン演算子の推定値と非線形項を組み合わせて残差を計算する
(7) 残差の平均二乗誤差と初期条件の誤差を最小化するようにニューラルネットワークのパラメータを更新する

[0168]
この方法により、高次元のセミリニア放物型PDEも効率的に解くことができる。

8. 産業上の利用可能性

[0169]
本発明は、以下の産業分野で広く利用可能である：

[0170]
1. 科学計算：高次元偏微分方程式の数値解法において、本発明は従来は計算が不可能だった超高次元問題（100万次元以上）を解決できるようにする。これにより、量子力学、流体力学、統計物理学などの分野での大規模シミュレーションが可能となる。具体的には、多体量子系のシュレーディンガー方程式、ナビエ・ストークス方程式、フォッカー・プランク方程式などの高次元PDEを効率的に解くことができる。

[0171]
2. 機械学習：物理情報ニューラルネットワーク（PINN）の高速化により、物理法則に基づいた機械学習モデルの訓練が効率化される。これは、自動運転、ロボティクス、医療画像処理など、物理法則が重要な役割を果たす応用分野で有用である。また、高次元最適化問題や逆問題の解決にも応用できる。例えば、PINNを用いた逆問題解決により、医療画像再構成、地震波トモグラフィー、非破壊検査などの分野での応用が可能となる。

[0172]
3. 量子多体系シミュレーション：多数の相互作用する粒子を含む量子系のシミュレーションにおいて、本発明は高次元シュレーディンガー方程式の効率的な解法を提供する。これにより、新材料設計、量子コンピューティング、量子化学などの分野での研究が促進される。具体的には、量子多体系の基底状態や励起状態の計算、量子相転移の解析、量子ダイナミクスのシミュレーションなどが可能となる。

[0173]
4. 金融モデリング：複雑な金融派生商品のための高次元ブラック・ショールズ方程式の効率的な解法を提供する。これにより、リスク管理、オプション価格付け、ポートフォリオ最適化などの金融工学の応用が向上する。特に、多資産オプション、バスケットオプション、エキゾチックオプションなどの価格付けや、高次元確率ボラティリティモデルの解析が可能となる。

[0174]
5. 計算流体力学：複雑な境界条件を持つ詳細かつ正確な流体シミュレーションを可能にする。これは、航空宇宙工学、気象予測、海洋学、生体力学などの分野で重要である。具体的には、乱流シミュレーション、多相流シミュレーション、流体構造連成解析、気象・気候モデリングなどに応用できる。

[0175]
6. 分子動力学シミュレーション：多数の原子や分子の相互作用をモデル化する高次元系のシミュレーションを効率化する。これにより、創薬、材料科学、生化学などの分野での研究が促進される。具体的には、タンパク質折りたたみシミュレーション、薬物-標的相互作用の解析、新材料の物性予測などが可能となる。

[0176]
7. 画像処理と計算機視覚：高次元偏微分方程式に基づく画像処理アルゴリズム（例：非線形拡散方程式に基づくノイズ除去、セグメンテーション）の高速化を実現する。これにより、医療画像処理、顔認識、物体検出、シーン理解などの応用が向上する。

[0177]
8. 最適制御：高次元ハミルトン・ヤコビ・ベルマン方程式の効率的な解法を提供し、複雑な動的システムの最適制御問題の解決を可能にする。これは、ロボット工学、自動運転、航空宇宙制御、プロセス制御などの分野で重要である。

[0178]
9. 生物学的シミュレーション：細胞内シグナル伝達、遺伝子制御ネットワーク、神経回路などの複雑な生物学的システムのモデリングを可能にする。これにより、システム生物学、神経科学、合成生物学などの分野での研究が促進される。

[0179]
10. エネルギーシステム最適化：電力網、再生可能エネルギーシステム、スマートグリッドなどの複雑なエネルギーシステムの最適化問題を解決する。これにより、エネルギー効率の向上、コスト削減、信頼性向上などが実現できる。

[0180]
11. 通信ネットワーク設計：大規模通信ネットワークの設計と最適化問題を解決する。これにより、5G/6G通信、IoTネットワーク、衛星通信などの分野での応用が可能となる。

[0181]
12. 社会経済シミュレーション：多数のエージェントが相互作用する社会経済システムのシミュレーションを可能にする。これにより、経済政策分析、交通シミュレーション、感染症拡散モデリング、都市計画などの分野での応用が可能となる。

[0182]
本発明により、従来は計算が不可能だった超高次元問題（100万次元以上）を解決できるようになり、科学的発見やエンジニアリング設計の新たな可能性が開かれる。これにより、より複雑で現実的なモデルの解析が可能となり、様々な分野での技術革新が促進される。