## 格子場の量子論のシミュレーションに 向けた機械学習手法の開発



大阪国際工科専門職大学 助教



akio\_at\_yukawa.kyoto-u.ac.jp

MLPhys Foundation of "Machine Learning Physics" Grant-in-Aid for Transformative Research Areas (A)

## Outline

- 自己紹介
- 素粒子物理学とは?
- 格子QCDとその目的・問題点
- 理論: ニューラルネットを用いたQCD
- コード: QCD+MLのためのコード開発
- これからの展望

## 自己紹介: 富谷昭夫 素粒子物理 + 機械学習



#### 研究分野 素粒子理論の数値計算が専門の物理屋 機械学習をつかって計算できることを増やしたい

#### 主な論文 <u>https://scholar.google.co.jp/citations?user=LKVqy wAAAAJ</u>

#### Detection of phase transition via convolutional neural networks

Journal of the Physical Society of Japan 86 (6), 063001 ニューラルネットを使った相転移検出

#### Digital quantum simulation of the schwinger model with topological term via adiabatic state proparation

state preparation									
B Chakraborty, M Honda, T Izubuchi, Y Kikuchi, A Tomiya									
arXiv preprint arXiv:2001.00485									

量子コンピュータで 場の理論計算

#### 略歴

- :大阪大学にて博士号取得(素粒子理論) 2015 2015 - 2018: 華中師範大学・ポスドク (中国、武漢) 2018 - 2021: 理研/BNL・ポスドク (米国、NY)
- :大阪国際工科専門職大学・テニュア助教 2021 -

#### 科研費

#### 学術変革A 計画班A01代表

MLPhys Foundation of "Machine Learning Physics" Grant-in-Aid for Transformative Research Areas (A)

その他量子計算等

#### その他:

Deep Learning and physics (セミナー)運営 シン仮面ライダーの監修をしました



## (素粒子理論) 物理学とは?

## 理論物理学 物理学の大きさによる分類

ざっくりとしたスケール



あと宇宙も研究対象。

## 素粒子物理学 期待値を通じて「この世の確率分布」を探る

- •素粒子物理 = 相対性理論と量子力学を使い、<u>物質と力の根源を探る</u>
- 量子力学:予測可能なのは期待値のみ
- 物理量O (たとえば粒子の質量) の量子力学的な期待値の計算

$$\underbrace{\mathbb{E}[O]}_{\mathbb{I}[O]} = \int \cdots \int (d\phi_1 d\phi_2 \cdots) O(\phi_1, \phi_2, \cdots) \underbrace{p(\phi_1, \phi_2, \cdots)}_{\text{確率分布}(知りたい)}$$

ワークフロー (数年~数10年規模):
いくつかのOでE[*O*]を測定(実験家)
→もっともらしい *p*(*φ*<sub>1</sub>…)を推定(理論家)
→ *p*(*φ*<sub>1</sub>…)と*O*' でE[*O*']を計算(理論家)
→*p*(*φ*<sub>1</sub>…)の予想が正しいかE[*O*']を測定してチェック(実験家)

# 素粒子物理学

- 10<sup>-19</sup> m 以上の大きさの実験を、**ほぼ**説明可能な確率分布p を知っている
- ・ (10<sup>-19</sup> m以上は、宇宙の歴史だと宇宙誕生後10<sup>-5</sup> 秒以降に相当)

• 
$$p = \frac{1}{Z} \exp(-S)$$
, (この確率分布で期待値 -> 実験・観測と比較)  
•  $S = \int d^4x \left[ \bar{\psi} D \psi(x) + \frac{1}{4} \sum_{a} F^{(a)}_{\mu\nu} F^{(a)}_{\mu\nu} + \phi \bar{\psi} y \psi(x) + \phi^{\dagger} D^2 \phi(x) + V(\phi) \right]$ 

「神の数式」(by NHK スペシャル)

- 10<sup>-19</sup>mよりミクロの世界(or 宇宙誕生後10<sup>-5</sup> 秒以前)
  - → 実験と計算がまだまだ必要 たとえば、ILC やCERNのLHC、宇宙観測 etc
  - → また本当に上記の確率分布で良いのかは精密計算が必要

# 素粒子理論での従来法と問題

• 10<sup>-19</sup>m 以上の大きさの実験を、ほぼ説明可能な確率分布p を知っている

• 
$$p = \frac{1}{Z} \exp(-S)$$
, (この確率分布で期待値 -> 実験・観測と比較)  
•  $S = \int d^4x \left[ \bar{\psi} D \psi(x) + \frac{1}{4} \sum_{a} F^{(a)}_{\mu\nu} F^{(a)}_{\mu\nu} + \phi \bar{\psi} y \psi(x) + \phi^{\dagger} D^2 \phi(x) + V(\phi) \right]$ 

• 期待値の計算は下記とほぼ同じ (1万次元積分, g は定数)

$$\int d\phi_1 \int d\phi_2 \cdots \int d\phi_{10000} \exp\left[-\sum_i \left(\phi_i^2 + g\phi_i^4\right)\right]$$

- 不定積分は出来ない。計算手法は以下の2つ。
  - 手法1:gでテイラー展開 (gが大きい時に破綻)
  - 手法2: 数值積分!

## 既存手法の成功 期待値の計算、数値積分も難しい

積分したい式 (おもちゃ模型)

$$\int d\phi_1 \int d\phi_2 \cdots \int d\phi_{10000} \exp\left[-\sum_i (\phi_i^2 + g\phi_i^4)\right]$$



ただの多重積分、ただし10,000次元以上(多変数積分)。 台形法などで誤差を抑えて計算するには現在のスパコンでも100億年以上 かかってしまう。

## 既存手法の成功 なぜ普通に数値積分できない?

2次元積分でみてみる  $\mathbb{E}[O] = \frac{1}{Z} \int dx_1 dx_2 e^{-\frac{1}{2}(x_1^2 + x_2^2 + x_1 x_2)} O[x_1, x_2]$  Zは  $\mathbb{E}[1] = 1$ とする規格化定数  $P(x_1, x_2) = \frac{1}{Z} e^{-\frac{1}{2}(x_1^2 + x_2^2 + x_1 x_2)}$ 



ほとんど至る所で確率が0。 賽の目に切って数値積分だと、無駄が多い。 誤差は、積分の次元で支配される。 (マルコフ連鎖) モンテカルロを使いましょう。

## 格子QCDとその目的・問題点

# 格子QCD = 強い核力を第一原理的に計算

$$\begin{aligned} & A_{\mu}(x) \in \mathrm{su}(3) \\ & L_{QCD} = -\frac{1}{4} \mathrm{tr} \ F_{\mu\nu} F^{\mu\nu} + \bar{\psi} (\mathrm{i}\partial - gA - m) \psi \\ & F_{\mu\nu} = \partial_{\mu} A_{\nu} - \partial_{\nu} A_{\mu} - \mathrm{i}g[A_{\mu}, A_{\nu}] \\ & |\Omega(t)\rangle = \mathrm{e}^{-\mathrm{i}\hat{H}_{QCD}t} |\Omega\rangle \quad \leftarrow \mathbf{con i} \mathbf{h} \mathbf{con} \mathbf{b} \mathbf{con} \end{aligned}$$





- 量子電磁気学の拡張 (ゲージ群が非可換)
- 陽子中性子等の内部を第一原理・量子論的に記述
- 宇宙の歴史やミクロ世界の理解に重要
- 弱結合展開での計算以外の部分がessential
- 格子QCD = 離散時空上のQCD
  - 数学的に厳密な定式化、連続極限でQCD
  - 数値計算できる! (が大変)

# 格子QCD = 強い核力を第一原理的に計算

格子QCD = 離散時空上のQCD (~1万粒子程度のdofの多粒子の量子力学) 格子QCDを経路積分量子化 = 10万次元程度の**多重積分**で期待値を計算可能

 $\langle O \rangle = \int \prod_{n,\mu} dU_{\mu}(n) P_{\text{QCD}}[U] O[U] \qquad P_{\text{QCD}}[U] \propto e^{-S_{g}[U] - S_{q}[U]}$ 

2変数での例 
$$\langle O \rangle = \int dx_1 dx_2 P(x_1, x_2) O(x_1, x_2)$$
  
 $P(x_1, x_2) \propto e^{-\frac{1}{2}(x_1^2 + x_2^2 + x_1 x_2)}$ 



ほとんど至る所で確率が0。 台形則等のニュートンコーツ式の数値積分だと、 誤差は、積分の次元で支配

(マルコフ連鎖) モンテカルロで積分・期待値を計算

## 格子QCDとその目的・問題点 マルコフ連鎖モンテカルロで量子論の計算ができる

<u>HMC法(ハイブリッドモンテカルロ法)のステップ</u>



()のステップ  $H[x,p] = \frac{1}{2}p_1^2 + \frac{1}{2}p_2^2 + S(x_1,x_2)$ 1.  $x = \{x_1, x_2\}$ を用意する 2.  $p = \{p_1, p_2\}$ をガウス分布からサンプルする("サイコロを振る") 3.  $x = \{x_1, x_2\} \ge p = \{p_1, p_2\}$ を上記のハミルトニアンから出て くる(仮想的な)運動方程式に従って、数値積分し、x', p'を得る 4.  $H[x,p] \ge H[x',p']$ を比較し、メトロポリス法を適用し、 × の列を得る



配位(サンプル)をたくさん集めて平均すれば積分したことになる

## 格子QCDとその目的・問題点 マルコフ連鎖モンテカルロで量子論の計算ができる

<u>HMC法(ハイブリッドモンテカルロ法)のステップ</u>



(美)のステップ  $H[x,p] = \frac{1}{2}p_1^2 + \frac{1}{2}p_2^2 + S(x_1, x_2)$ 1.  $x = \{x_1, x_2\}$ を用意する 2.  $p = \{p_1, p_2\}$ をガウス分布からサンプルする("サイコロを振る") 3.  $x = \{x_1, x_2\} \ge p = \{p_1, p_2\}$ を上記のハミルトニアンから出て くる(仮想的な)運動方程式に従って、数値積分し、x', p'を得る 4.  $H[x,p] \ge H[x',p']$ を比較し、メトロポリス法を適用し、 × の列を得る

$$\begin{split} \langle O \rangle &= \frac{1}{N_{\text{sample}}} \sum_{i=1}^{N_{\text{sample}}} O[x^{(i)}] + \frac{\text{const}}{\sqrt{N_{\text{sample}}}} \\ \text{格子QCDの経路積分も、ただの10^5次元多重積分} \\ \langle O \rangle_{a,L} &= \frac{1}{Z} \int \cdots \int \prod_{x \in \mathbb{Z}/L} \prod_{y \in \mathbb{Z}/L} \prod_{z \in \mathbb{Z}/L} \prod_{u=1}^{4} dU_{\mu}(x, y, z, t) e^{-S(U)} O[U] \end{split}$$

物凄く頑張るとHMCで格子QCDの期待値を評価できる。

# 格子QCDの計算はスパコンで! (1980年~)



#### <u>スパコンで計算して何がわかるのか?</u>

- 陽子/中性子の仲間(ハドロン)の質量
- 原子核同士の引力/斥力の様子 (星の生まれて死ぬまでを理解するのに必要)
- 高温での陽子/中性子等の溶解の様子 (宇宙の歴史に関わる)
- ダークマターの候補の性質 (実験で見つけるには性質を知っておく必要あり)
- 陽子/中性子内のクォークの様子
- 手計算で計算できない各種係数 (素粒子標準理論と実験との整合性チェックに必要) などなど...

### 格子QCDとその目的・問題点 色々出来てても問題もある。機械学習?





#### <u>機械学習で解決したい問題</u>

- (いろんな意味で) 計算量の削減。大きな体積、細かい格子間隔での計算をしたい - 符号問題の解決 (今日は触れません)
- 格子QCDに機械学習を用いる縛り
  - (マルコフ連鎖モンテカルロ法としての)厳密性。機械学習は近似的
  - ゲージ対称性、フェルミオン自由度(量子論特有)。機械学習は古典データ
  - コード■。ニューラルネット(Python/Jupyter)、格子QCD(C、C++、Fortran)...



## ニューラルネットを用いたQCD

## **理論:ニューラルネットを用いたQCD** Akio Tomiya フローベースアルゴリズム: 確率分布の簡単化

80年代や2010年ぐらいから、幾度と積分測度を簡単な物にできないか議論されてきた

$$\mathbb{E}[O] = \frac{1}{Z} \int \cdots \int \prod_{x=1}^{100} \prod_{y=1}^{100} \prod_{z=1}^{100} \prod_{t=1}^{100} d\phi_{x,y,z,t} e^{-S(\phi)} O[\phi_{x,y,z,t}]$$

 $ilde{\phi} = \mathscr{F}(\phi)$  フロー方程式 (変数変換)

もし $S(\mathcal{F}(\phi)) - \ln \det(\text{Jacobian}) = \phi^2$ ならば、

$$\mathbb{E}[O] = \frac{1}{Z} \int \cdots \int \prod_{x=1}^{100} \prod_{y=1}^{100} \prod_{z=1}^{100} \prod_{t=1}^{100} d\tilde{\phi} O[\mathscr{F}_{\tau}(\phi)] e^{-\tilde{\phi}^2} \det\left(\frac{\partial \phi}{\partial \tilde{\phi}}\right)$$

ガウス積分なので簡単そう。

適当にやるとヤコビアンは密になってしまい、現実的に計算できない

## **理論:ニューラルネットを用いたQCD** Akio Tomiya ニューラルネットでフロー方程式をミミックする



FIG. 1: In (a), a normalizing flow is shown transforming samples z from a prior distribution r(z) to samples  $\phi$  distributed according to  $\tilde{p}_f(\phi)$ . The mapping  $f^{-1}(z)$  is constructed by composing inverse coupling layers  $g_i^{-1}$  as defined in Eq. (10) in terms of neural networks  $s_i$  and  $t_i$  and shown diagrammatically in (b). By optimizing the neural networks within each coupling layer,  $\tilde{p}_f(\phi)$  can be made to approximate a distribution of interest,  $p(\phi)$ .

#### ニューラルネットで<u>可逆な</u>変数変換を再現

「自明な理論 ← → 相互作用のある理論」を結ぶ1:1写像

自明な理論(各点独立なガウス分布)でサンプルし、場の理論の配位に変換

Normalizing flow(正規化フロー) という仕組みでできる

ヤコビアンも簡単に計算可能!

arxiv 1904.12072, 2003.06413, 2008.05456





## **理論:ニューラルネットを用いたQCD** Akio Tomiya ゲージ対称性を取り扱えるニューラルネット?

- 非可換ゲージ対称性: QCDの指導原理
  - $A_{\mu}(x) \rightarrow G(x)A_{\mu}(x)G^{-1}(x) G(x)\partial_{\mu}G^{-1}(x)$ の下で経路積分不変
  - これがないと量子論として矛盾する
  - ゲージ場 = リー群(ユニタリー行列)の場
- 機械学習で出てくるデータ
  - 大域的対称性は考慮できる(畳み込み、同変性)
  - ニューラルネットは実数を処理する処理系
- どうやる?

### 理論:ニューラルネットを用いたQCD ゲージ対称性を保ちながらのデータ処理

- いろんな手法でニューラルネットにゲージ対称性を担保できる
- ゲージ同変ネット (Maximal torus), DeepMind, MIT+ arXiv: 2008.05456
  - リンクごとに対角化、U(1)の位相をニューラルネットへ
  - ゲージ変換とニューラルネットの処理は交換(同変)
- ゲージ共変ネット (trainable smearing) AT, Y. Nagai arXiv: 2103.11965
  - スメアリングの係数がbackprop で決定できる事を発見
  - Stout smearing *O*Neural ODE = Gradient flow !
  - ゲージ変換に対して共変的な構成
- 他にも実現あり

# 理論:ニューラルネットを用いたQCD Akio Tomiya MITグループによるFull QCDでの結果



- Flow based での、Full QCD in 4dの結果! HMCと無矛盾
- 超えるべき壁: 計算速度。Full Python...

• (速度は。。。あとスケーリングも気になる)



- ゲージ対称性を保った情報処理(擬フェルミオン場を入力し、 擬フェルミオン場を出力する)をゲージ対称性を保ったニューラルネットで
  - ディラック演算子も、uを入力、Dを返す情報処理
- ニューラルネットで、前処理行列の処理を近似。
- 反復を10倍程度の改善(ただし実際の実行速度はPython律速?)

https://arxiv.org/abs/2302.05419, C. Lehner, and T. Wettig

## コード: QCD+MLのためのコード開発 (2言語問題を解決する)

プログラミングにおける2言語問題を教えて下さい。何が問題なのでしょうか。

\$

プログラミングにおける「2言語問題」(Two Language Problem) は、主にデータ分析や科学計算 △ ▽ などの領域でよく見られる現象で、プログラマがパフォーマンスと柔軟性の両方を求めて、異なる プログラミング言語を使い分けることを指します。一般的には、1つの言語が高性能で低レベルな 処理に適している一方で、もう1つの言語が柔軟性や生産性に優れているために発生します。

例えば、C++やFortranは高速な数値計算に優れていますが、プログラムの開発やデバッグには時 間がかかることがあります。一方、PythonやRなどのスクリプト言語は開発やデバッグが容易で、 様々なライブラリやツールが利用できますが、パフォーマンス面ではC++やFortranに劣ることが 多いです。

2言語問題が問題とされる理由は以下の通りです:

- 1. 学習コスト:異なる言語を習得し、使いこなすことに時間と労力が必要です。
- 2.保守性:複数の言語で書かれたコードは、統一されたコードベースよりも保守やデバッグが難しくなることがあります。
- インターフェースの複雑さ:異なる言語間でデータや機能を共有する際には、インターフェー スやラッパーを用意する必要があります。これにより、システムの複雑さが増すことがありま す。

近年、この問題に対処するために、Juliaなどの新しいプログラミング言語が登場しています。 Juliaは高性能な数値計算が可能でありながら、Pythonのような柔軟性や開発効率を提供すること を目指して設計されており、2言語問題の緩和に寄与しています。

29

AK

### コード: QCD+MLのためのコード開発 Akio Tomiya QCD+MLにおける2言語問題

- 格子QCD: 1億行、1億列の疎行列・線形方程式を何万回も解く
  - Massiveな並列化をスパコンの上でやりたい
  - GPUを使うとしてもマルチGPU前提ですよね...MPIバンザイ
  - C++かFortran 並のスピードが必要
- 機械学習(ニューラルネット): Pythonがデファクトスタンダード。100倍遅い。
  - Jupyter Notebookで実験したい。GPU化もほぼ自動で。
  - TensorFlow, PyTorch等でならコーディング出来る。Auto-Grad が無いと...
  - 複素数?頑張れば。
- C++とPythonの両方のコーディングが必要???
- Julia なら自然に両立できるんです

### コード: QCD+MLのためのコード開発 Akio Tomiya Speed of Julia ~ Clang

1.Open source scientific language (Just in time compiler/LLVM backend)
 2.Fast as C/Fortran (faster sometimes), Practical as Python
 3.Machine learning friendly

	Compiler	Benchmark (sec) single core	Туре	Parallelism	GPU	Pros	Cons👎	Column, row	Note
Julia (1.8)	JIT, LLVM	0.0014	Dynamic & Static	MPI, others	CUDA	Fast Practical ML feiendly	not major	column- major	
С	Clang (LLVM)	0.0033	Static	MPI, others	CUDA	Fast	Long codes	row-major	
Python +Numba	(CPython) JIT, LLVM	0.0131	Dynamic	Available	Numba- CUDA	Practical ML feiendly	Not fully supported	row-major (Numpy)	(Rosetta2 is used in benchmark)

#### C and Julia have similar speed

Benchmarks are performed on m1 mac mini (similar tendency on Xeon) Benchmark: Multiplications for 12dim vector and 12x12 complex matrix for 10^4 times (repeated 10 times)

#### コード: QCD+MLのためのコード開発 Akio Tomiya Code comparison (skip)



- Complex matrix (12x12) times complex vector (d=12)
  - One set= 10^4 times, and repeated 10 times and averaged
- Code of Julia looks like Python (short, simple) but fast as C Julia: 0.0014 (sec), C: 0.0033 (sec). Single core performance is similar

## コード: QCD+MLのためのコード開発 Akio Tomiya

LLVM = common backend for making binaries on multiple architectures



https://www.fujitsu.com/jp/about/businesspolicy/tech/fugaku/ https://ja.wikipedia.org/wiki/Apple\_M1 https://ja.wikipedia.org/wiki/Ryzen

https://ja.wikipedia.org/wiki/Xeon https://gigazine.net/news/20200623-japan-fugaku-fastest-supercomput

See: https://en.wikipedia.org/wiki/LLVM and related pages











#### コード: QCD+MLのためのコード開 Wilson inversion / MPI parallel, Strong Scaling $A^{kin}$ Tormiva $A^{Kin}$ Tormiva

#### **Tested on Yukawa-21@YITP**



良さそうに見える もうちょっとコントリビューターが必要



Neurips Workshop (https://ml4physicalsciences.github.io/2022/) で関連研究も発表

### QCD+MLのための理論とコード ひとまず出来ることはわかった。やるべきことも。

- 理論サイド: Flow based が良さそう。ただ他の可能性も探るべき。
  - 4d SU(3), Full QCDには(ようやく)手が届いた。
  - ソルバーへの利用もある。
  - ただし実装・実際の計算速度は別問題。まだまだ改善しないと。
- コードサイド: 2言語問題 = 格子QCDとニューラルネットのコーディング
  - Julia は2言語問題を自然に解決できる
  - 格子QCDのコードとFlowのJulia コードを作りました!
  - 人の手が足りていません。
- まだまだこれから...。





## **Applications**

#### Configuration generation with machine learning is developing

#### Configuration generation for 2d scalar

<u>Restricted Boltzmann machine + HMC: 2d scalar</u> A. Tanaka, AT 2017 The first challenge, machine learning + configuration generation. Wrong at critical pt. Not exact.

GAN (Generative adversarial network ): 2d scalar

Results look OK. No proof of exactness

Flow based model: 2d scalar, pure U(1), pure SU(N)

Mimicking a trvializing map using a neural net which is reversible and has tractable Jacobian. Exact algorithm, no dynamical fermions. SU(N) is treated with diagonalization.

#### L2HMC for 2d U(1) (Sam Foreman+ 2021)

Self-learning Monte Carlo (SLMC) for lattice QCD Non-abelian gauge theory with dynamical fermion in 4d

Using gauge invariant action with linear regression Exact. Costly (Diagonalize Dirac operator)

#### Self-learning Hybrid Monte Carlo for lattice QCD (SLHMC, This talk)

Non-abelian gauge theory with dynamical fermion in 4d arxiv 2103.1<sup>-</sup> Using covariant neural network to parametrize the gauge invariant action Exact

J. Pawlowski+ 2018 G. Endrodi+ 2018

arxiv 2010.11900 Y. Nagai, AT, A. Tanaka



arxiv 2103.11965 Y. Nagai, AT



40







## **Benchmark** Why Julia? (My personal opinion)

[1] https://akio-tomiya.github.io/julia\_in\_physics/ [2] <u>https://qr.ae/prgSG5</u>

- Modern scientific programming language
- Easy to make codes. Fast as C/C++ (Julia& C use LLVM)
- Fewer compiling/dependency issues.
- Many people are potentially interested in. (More than 400 people registered to "Julia in physics 2022 online workshop" [1]). 4,923 public repo on Github
- No two Language problem. "The fact that while the users are programming in a high-level language such as R and Python, the performance-critical parts have to be rewritten in C/C++ for performance". [2]
  - Neural network friendly (Flux.jl). Tensor networks also (iTensor.jl).
- Works on/with
  - Xeon, Radeon/Apple silicon/<u>A64FX</u>
  - MPI, GPU



## Akio Tomiya

Our lattice QCD codes are constructed by following repositories

