AI-Assisted Higher-Order Hopping-Parameter Expansion in Lattice QCD

Tatsuya Wada (Kyoto University / YITP)

Collaborators: Masakiyo Kitazawa (Kyoto U.)

ChatGPT Codex (GPT-5.5, OpenAI)

Claude Code (Opus 4.8, Anthropic)

Based on arXiv:2606.31492

Lattice 2026, University of Maryland, College Park USA — July 31, 2026

YUKAWA INSTITUTE FOR THEORETICAL PHYSICS

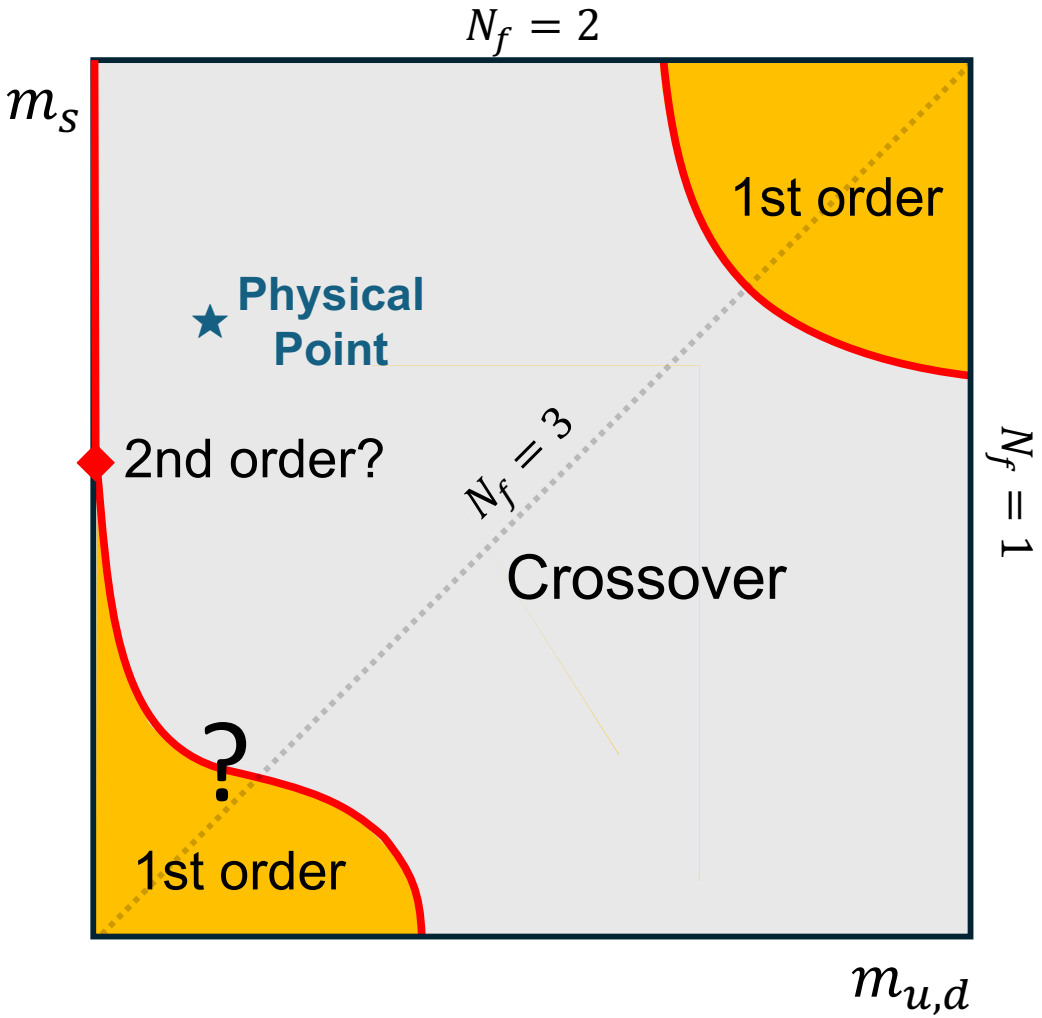
KYOTO UNIVERSITY FOUNDED 1897 KYOTO JAPAN

GitHub

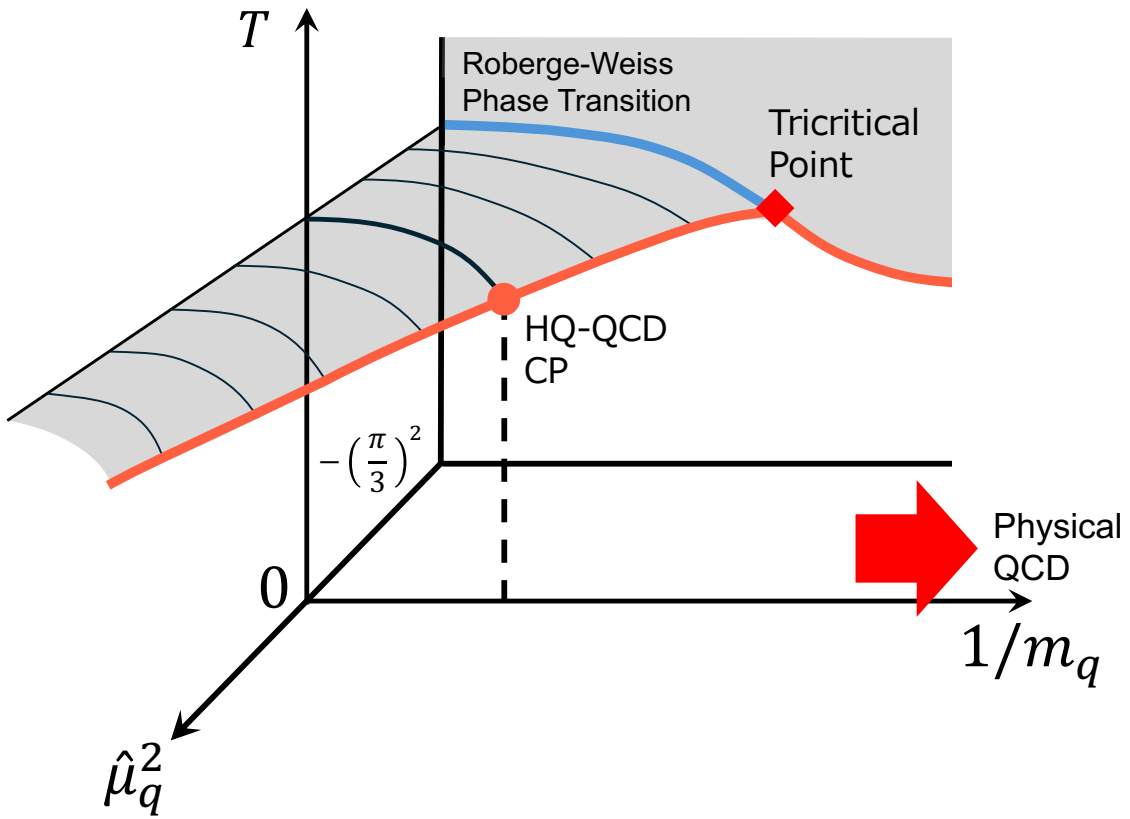
Why Hopping Parameter Expansion?

Kiyohara+('21); Ashikawa+('24); Wada+('25~)

Columbia Plot at $\mu = 0$



Phase Structure at $\mu = i\mathbb{R}$



HPE = expansion in $\boldsymbol{1/m_q}$ → systematically controls the **heavy-quark corner** of these diagrams

Why Hopping Parameter Expansion?

How to calculate numerically?

$$\langle \bar{\psi}_x \psi_x \rangle = \mathrm{Tr}[M^{-1}]$$

M: large sparse matrix
$\sim 10^6 \times 10^6$

Use **HPE** and **Noisy Estimator !**

$$\langle \bar{\psi}_x \psi_x \rangle = \mathrm{Tr}[M^{-1}] = \Sigma_{n=4}^{\infty} \kappa^n \mathrm{Tr}[B^n] = \underline{\Sigma_{n=4}^{N-2} \kappa^n \mathrm{Tr}[B^n]} + \kappa^N \underline{\mathrm{Tr}[B^N M^{-1}]}$$

exact HPE　　noisy estimator

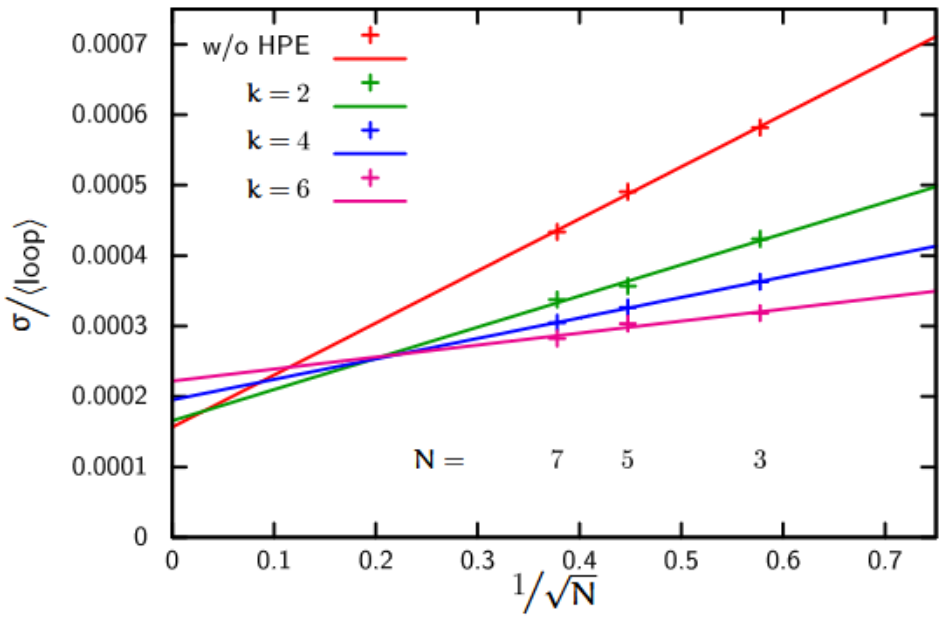
$$M = 1 - \kappa B$$

—Exact evaluation of lower-order terms in HPE can reduce statistical errors

—However, previous studies used HPE up to only κ^6 terms (NLO level).

➡ **Let's extend HPE to higher order!**

Guelpers+, 1309.2104



Hopping Parameter Expansion

Wilson fermion

$$S_q = \sum_{x,y} \bar{\psi}(x) \left(\delta_{x,y} - \kappa \sum_{\mu} \left[(1-\gamma_\mu) U_\mu(x) \delta_{x+\hat{\mu},y} + (1+\gamma_\mu) U_\mu^\dagger(y) \delta_{x-\hat{\mu},y} \right] \right) \psi(y)$$

$$\equiv \bar{\psi}(x) M(x,y) \psi(y)$$

$$M(x,y) = \delta_{x,y} - \kappa B(x,y)$$

$$\kappa = \frac{1}{2m+8} \quad : \text{Hopping Parameter}$$

$$B(x,y) = \sum_{\mu} \left[(1-\gamma_\mu) U_\mu(x) \delta_{x+\hat{\mu},y} + (1+\gamma_\mu) U_\mu^\dagger(y) \delta_{x-\hat{\mu},y} \right] \quad : \text{Hopping Matrix}$$

Partition function using Wilson fermion

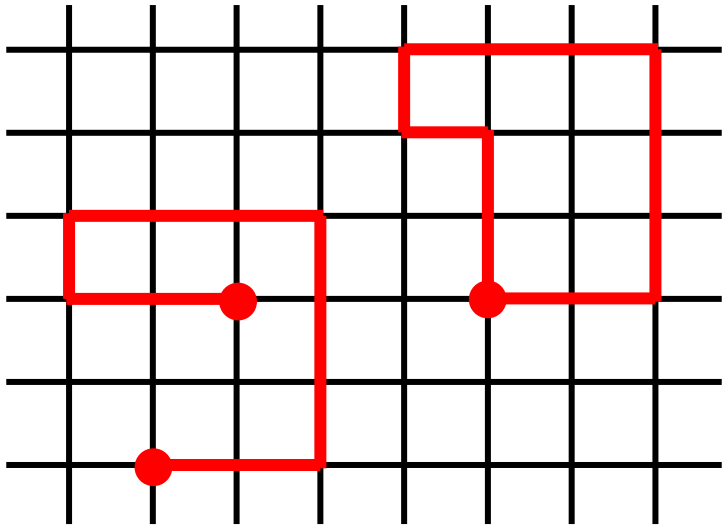
$$Z_{\text{Quark}} = \int \mathcal{D}\bar{\psi}\mathcal{D}\psi \; e^{-S_q} = \det(1-\kappa B) = \exp(\log\det(1-\kappa B)) = \exp(\operatorname{tr}\log(1-\kappa B))$$

$$\operatorname{tr}\log(1-\kappa B) = -\sum_{n=0}^{\infty} \frac{\kappa^n}{n} \operatorname{tr}(B^n)$$

Hopping Parameter Expansion

$$\mathrm{tr}\log(1-\kappa B) = -\sum_{n=0}^{\infty} \frac{\kappa^n}{n}\,\mathrm{tr}(B^n)$$

$$\mathrm{tr}(B^n)\begin{cases} = 0 & (\text{Opened path}) \\ \neq 0 & (\text{Closed path}) \end{cases}$$



$\mathrm{tr}(B^n)$: The sum over closed hopping paths of length n.

$W(n)$: The sum over Wilson-type closed loops of length n

$W(4)$ $W(6)$

Plaquette Rectangle Chair Crown

1 type 3 types

Purpose of this study

We implement the higher order terms of Hopping parameter expansion.

Higher Order Terms in the HPE

Can we implement the higher order terms of Hopping parameter expansion?

1. Classification of loops contributing to higher-order terms

Loop length n	Closed non-backtracking paths
4	48
6	912
8	28,944
10	976,800
12	34,953,120
14	1,302,548,688
16	50,125,332,432

The number of possible loop patterns grows exponentially with the loop length.

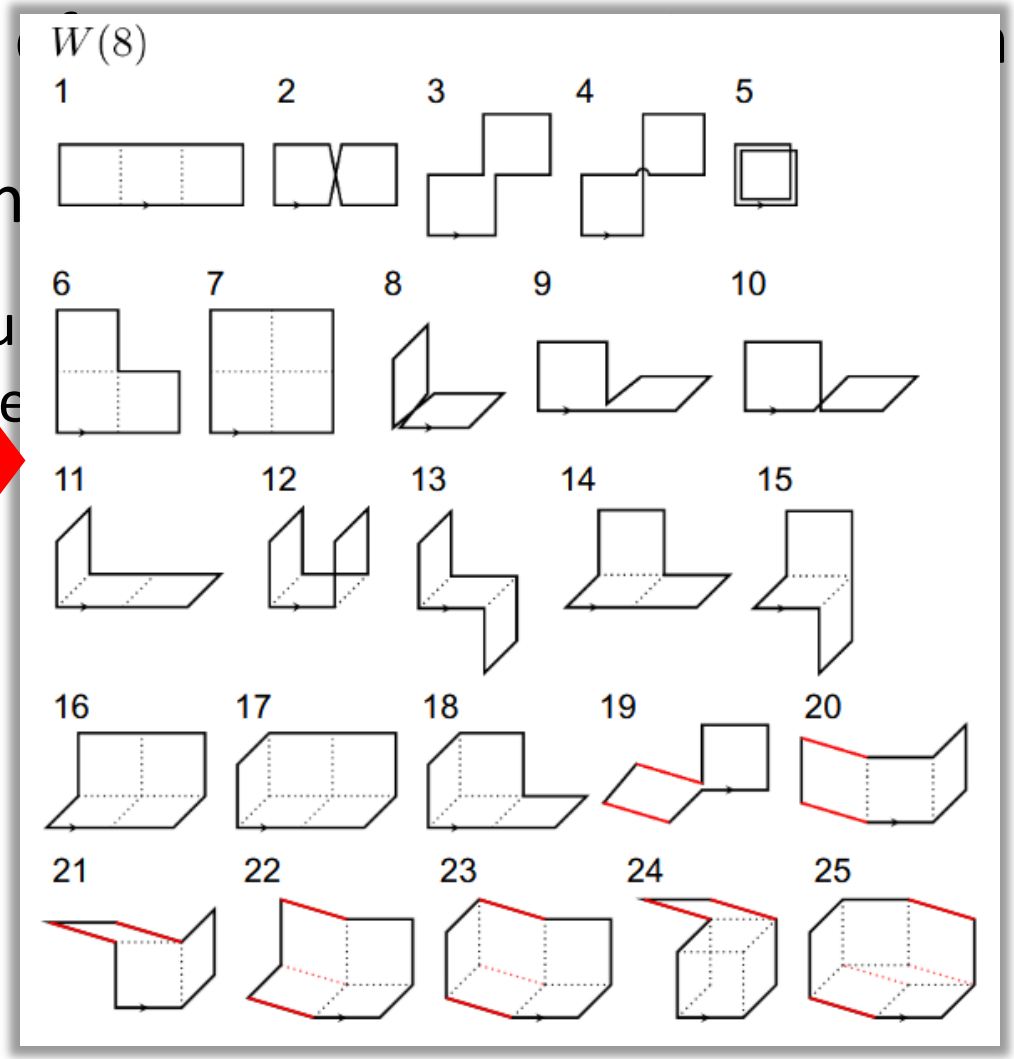
Higher Order Terms in the HPE

Can we implement the higher order terms ( ) ?

1. Classification of loops contributing to h

Loop length n	Closed non-backtracking paths
4	48
6	912
8	28,944
10	976,800
12	34,953,120
14	1,302,548,688
16	50,125,332,432

The nu
xp ne



Higher Order Terms in the HPE

Can we implement the higher order terms of Hopping parameter expansion?

1. Classification of loops contributing to higher-order terms

Loop length n	Closed non-backtracking paths
4	48
6	912
8	28,944
10	976,800
12	34,953,120
14	1,302,548,688
16	50,125,332,432

The number of possible loop patterns grows exponentially with the loop length.

2. Efficient implementation of higher-order terms in Monte Carlo simulations

HPE enables efficient Monte Carlo updates (Heatbath method).
However, when higher-order terms are included,
loop measurements become the dominant computational bottleneck.

Results of Classification

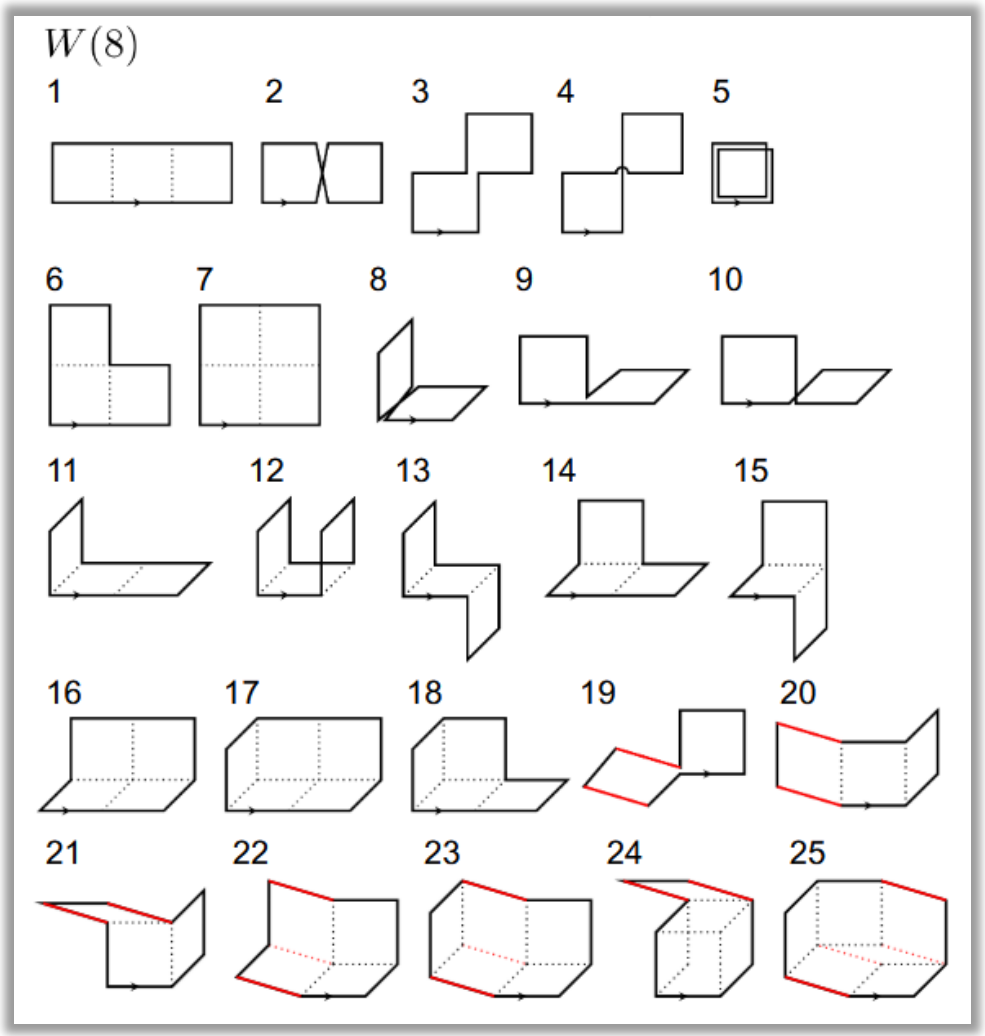
1. Classification of loops contributing to higher-order terms

We must classify the all possible loops by the symmetry
— cyclic permutation
— reverse
— cubic rotation/reflection ($H(4)$)

$n = 8$ can be classified by hand.
However, $n \geq 10$ is hard.

→ Make Classification algorithm

	N_{shape}
$W(4)$	1
$W(6)$	3
$W(8)$	24
$W(10)$	189
$W(12)$	3701

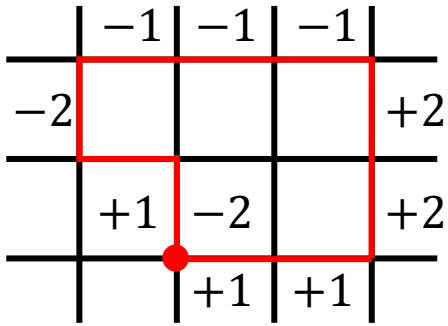


Shape Classification

1. Classification of loops contributing to higher-order terms

i. Representation of a loop as a string $\{\pm1, \pm2, \pm3, \pm4\}$

ex.) +1+1+2+2-1-1-1-2+1-2



ii. Enumeration of all possible loops

— Closed loop = $\pm\mu$ must appear equal times

— No appendices = Exclusion of adjacent $+\mu$ and $-\mu$ steps

iii. Classification by **invariants** under permutations of the coordinate axes

ex) Directional step-count invariant

+1,+1,+2,+2,−1,−3,−1,−1,−2,+1,−2,+3

$[n_1, n_2, n_3, n_4] = [3,2,1,0] \longrightarrow [0,1,2,3]$

Number of μ-direction

$n_\mu \equiv \#(+\mu) = \#(-\mu)$

iv. Classification of all loops by canonicalization about

— cyclic permutation

— reverse

— cubic rotation/reflection ($H(4)$)

→ Humans designed the algorithm; AI wrote the efficient code.

Coefficients of HPE

Wilson loop type contribution

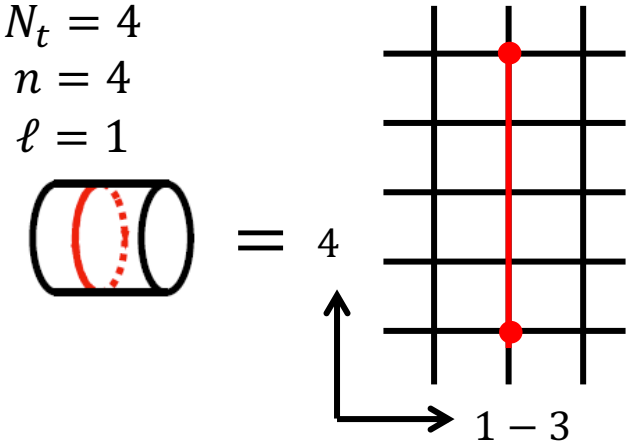
$$W(n) = -2N_{\rm c}\sum_j \frac{M_j D_j}{S_j}\mathrm{Re}\mathcal{W}_j(n)$$

$$B(x,y) = \sum_\mu \Big[(1-\gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)U_\mu^\dagger(y)\delta_{x-\hat{\mu},y}\Big]$$

Polyakov loop type contribution

$$L_\ell(N_t,n) = -(-1)^\ell N_{\rm c}\sum_j \frac{M_j D_j}{S_j}\mathcal{L}_j(N_t,n,\ell)$$

$N_t = 4$
$n = 4$
$\ell = 1$



- j : Label for shape class
- $W_j(n)$: Products of link variables $\mathcal{W}_j(n) = \sum_{\text{all trajectories in } j}^{M_j N_{\rm site}} \mathrm{tr}_{\rm c}[U_{x_1,\mu}U_{x_2,\nu}\cdots]/(N_{\rm c}N_{\rm site}M_j)$
- M_j: # of different trajs. in j
- D_j: Contribution of Dirac space $D_j = \mathrm{tr}_{\rm D}[(1-\gamma^\mu)(1-\gamma^\nu)\cdots]$
- S_j: Symmetry factor
- ℓ : Winding number (Wilson type $\ell = 0$)

Results of Classification

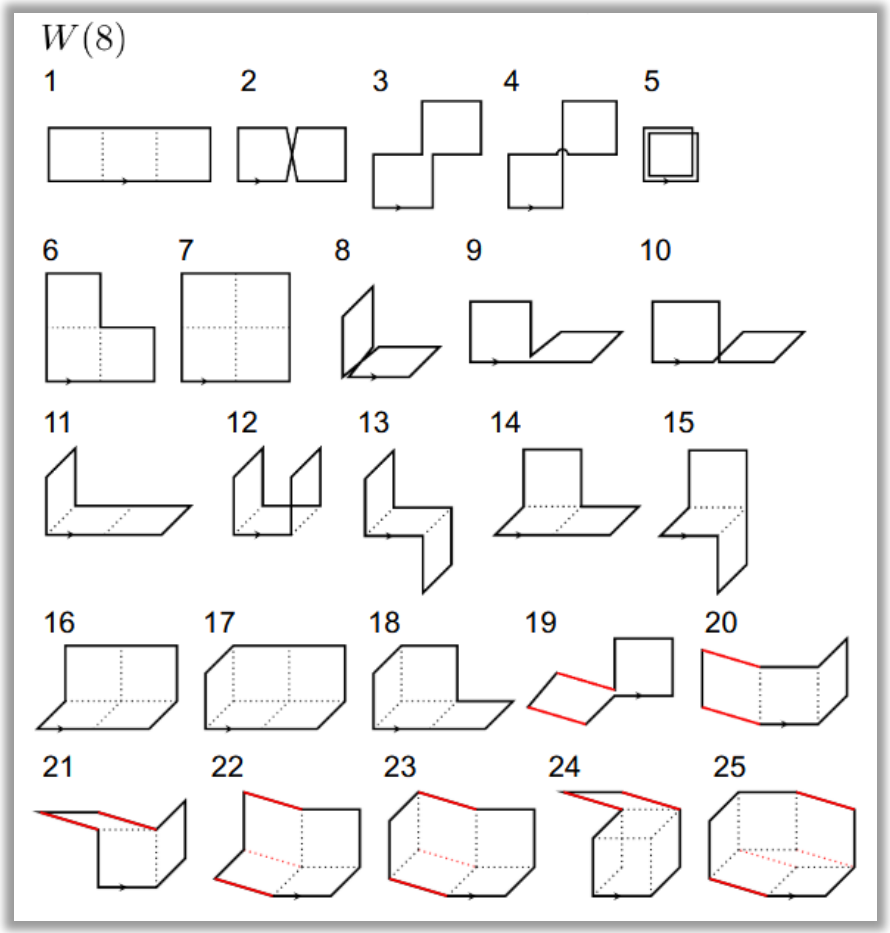


TABLE IV. Shape classification of $W(8)$.

j	trajectory	M_j	S_j	D_j	type
1	$11\bar{1}2\bar{1}\bar{1}\bar{2}$	12	1	−128	U=U
2	$12\bar{1}\bar{2}\bar{1}2\bar{1}$	12	1	32	PP
3	$12\bar{1}\bar{2}\bar{1}\bar{2}12$	12	1	−32	PP
4	$12\bar{1}\bar{2}\bar{2}\bar{1}21$	12	1	64	PP
5	$12\bar{1}\bar{2}1\bar{2}\bar{1}$	6	2	32	P^2
6	$112\bar{1}2\bar{1}\bar{2}\bar{2}$	24	1	−64	ULU
7	$1122\bar{1}\bar{1}\bar{2}\bar{2}$	6	1	−128	□
8	$12\bar{1}23\bar{2}\bar{3}\bar{2}$	48	1	32	PP
9	$12\bar{1}\bar{2}3\bar{1}31$	96	1	32	PP
10	$12\bar{1}2\bar{1}3\bar{1}\bar{3}$	96	1	0	PP
11	$112\bar{1}\bar{1}3\bar{2}\bar{3}$	96	1	−64	U=U
12	$123\bar{2}1\bar{2}\bar{3}\bar{2}$	24	1	−32	U=U
13	$123\bar{2}\bar{1}\bar{2}32$	48	1	−32	U=U
14	$112\bar{1}3\bar{1}\bar{3}\bar{2}$	192	1	−32	ULU
15	$123\bar{2}\bar{2}1\bar{2}\bar{3}$	96	1	−32	ULU
16	$1123\bar{1}\bar{1}\bar{3}\bar{2}$	48	1	−64	L==L
17	$1123\bar{1}\bar{1}2\bar{3}$	48	1	−64	L==L
18	$1233\bar{1}3\bar{2}\bar{3}$	192	1	−32	crown + U
19	$12\bar{1}234\bar{3}\bar{4}$	96	1	16	PP
20	$123\bar{2}1\bar{4}3\bar{4}$	96	1	−32	U=U
21	$123\bar{2}4\bar{1}\bar{4}\bar{3}$	192	1	−16	ULU
22	$123\bar{1}4\bar{3}2\bar{4}$	48	1	−32	ΣΣ
23	$123\bar{1}4\bar{2}3\bar{4}$	96	1	−32	ΣΣ
24	$1234\bar{1}4\bar{2}\bar{3}$	192	1	−16	crown + U
25	$1234\bar{1}2\bar{3}\bar{4}$	24	1	−16	4d-crown

$$W(n) \xrightarrow{\text{Weak limit}} -2N_c \sum_j \frac{M_j D_j}{S_j}$$

The coefficients of the Wilson loops can be calculated analytically in the weak coupling limit. [Wakabayashi et al., 2021].
This quantity provides a necessary condition.

Human Parts – Implementation

2. Efficient implementation of higher-order terms in Monte Carlo simulations

Implementation of Reference Program

Reference Program

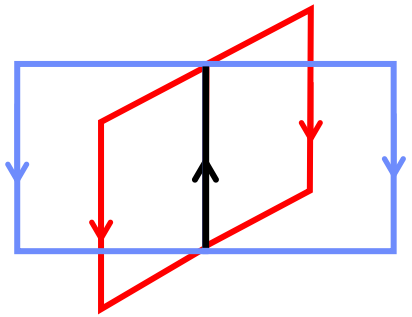
Enumerate all trajectories from every lattice site

Computationally expensive but reliable

	N_{shape}	N_{traj}
$W(4)$	1	6
$W(6)$	3	76
$W(8)$	24	$1.71 \cdot 10^3$
$W(10)$	189	$3.80 \cdot 10^4$
$W(12)$	3701	$1.03 \cdot 10^6$

Numerical Cost: $(n-2)N_{\mathrm{traj}}$ MMs/site

MM: matrix multiplication of $SU(N_c)$



W(8): 214 staples

W(10): 6300 staples

1 staple = 48 MMs/site

: Costs of all direction staples

Human Parts – Implementation

2. Efficient implementation of higher-order terms in Monte Carlo simulations

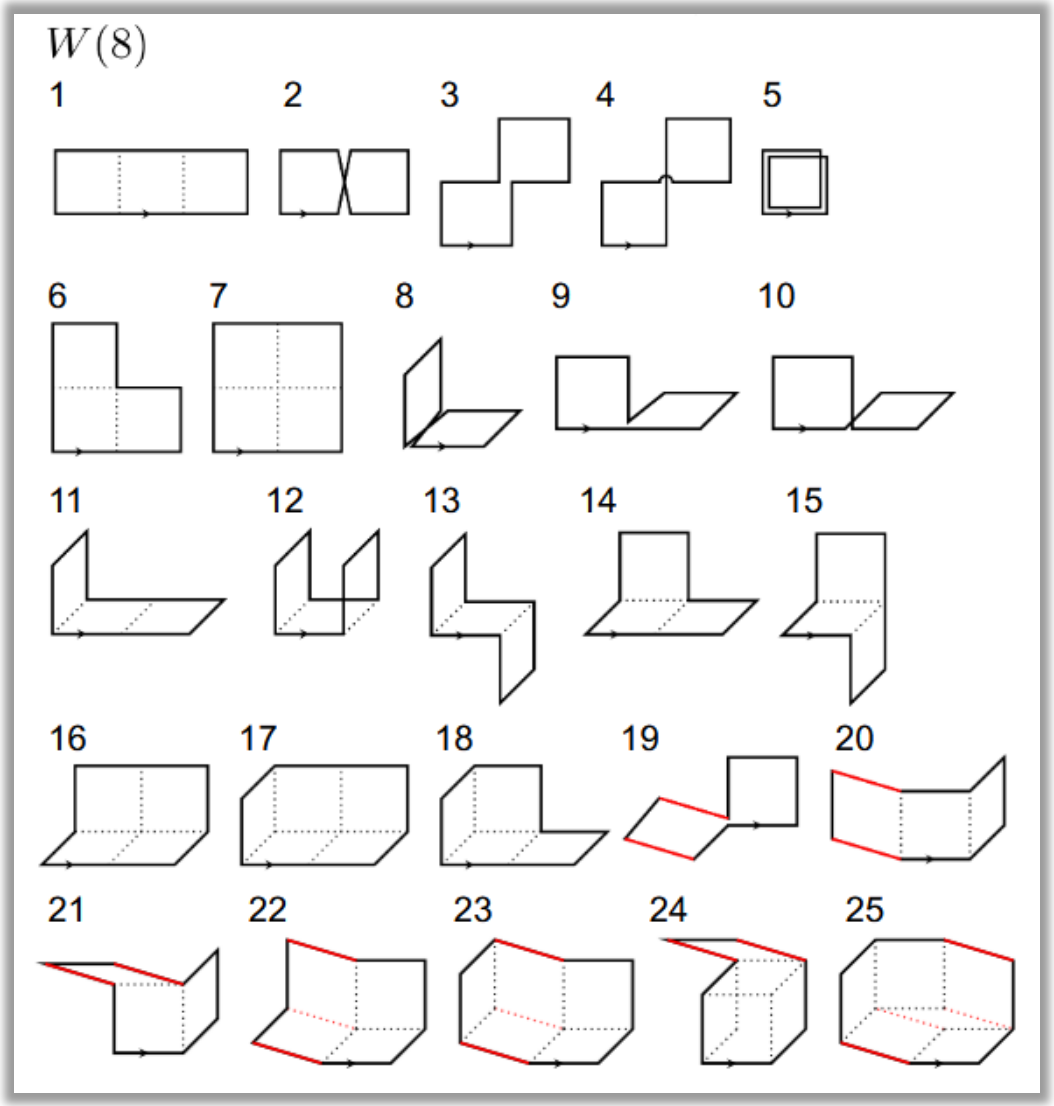
Developing N2LO by Human

Evaluate trajectories efficiently
based on the “shape-based” algorithm

- taking partial summation
- recycle single-staples
- and plaquette

Efficiency of our latest code for W(8)

214 staples → **20 staples**



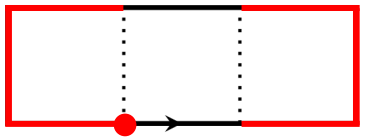
Human Parts – Implementation

2. Efficient implementation of higher-order terms in Monte Carlo simulations

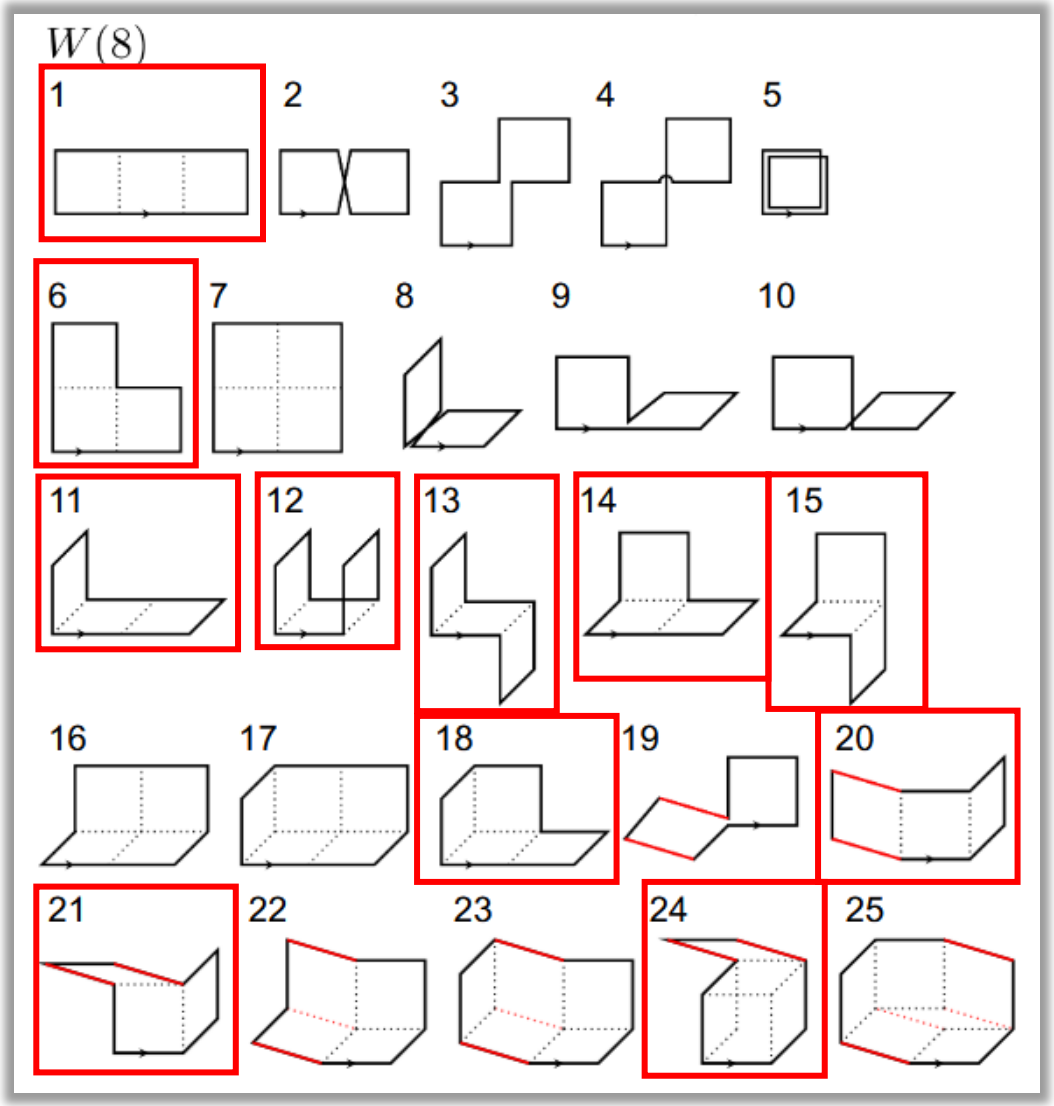
Developing N2LO by Human

Evaluate trajectories efficiently
based on the “shape-based” algorithm

- taking partial summation
- recycle single-staples
- and plaquette



Naïve : 6 MMs
Staple : 1 MM



Efficiency of our latest code for W(8)

214 staples → **20 staples**

Human Parts – Implementation

2. Efficient implementation of higher-order terms in Monte Carlo simulations

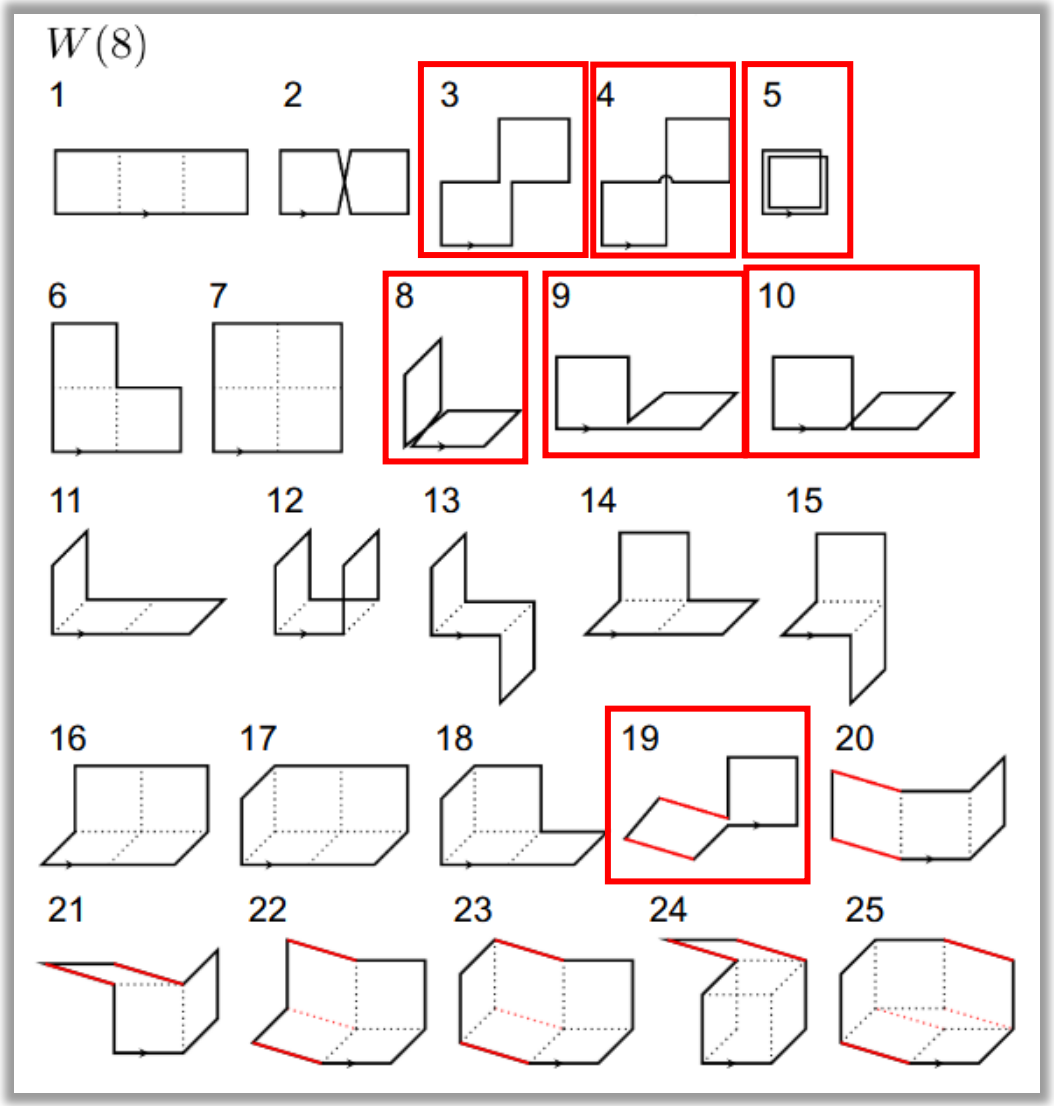
Developing N2LO by Human

Evaluate trajectories efficiently
based on the “shape-based” algorithm

- taking partial summation
- recycle single-staples
- and plaquette

Efficiency of our latest code for W(8)

214 staples → **20 staples**



Human Parts – Implementation

2. Efficient implementation of higher-order terms in Monte Carlo simulations

Developing N2LO by Human

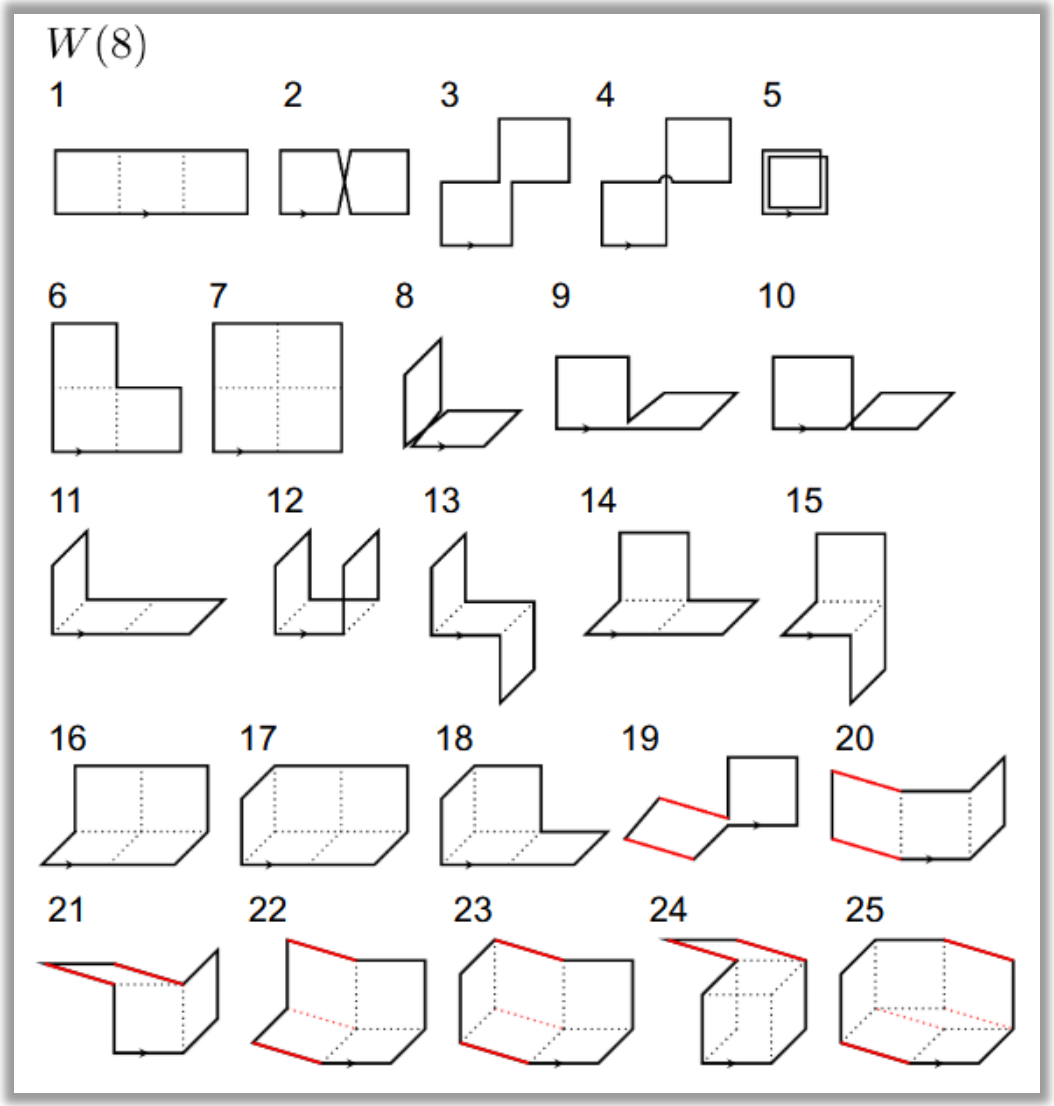
Evaluate trajectories efficiently
based on the “shape-based” algorithm

- taking partial summation
- recycle single-staples
- and plaquette

Efficiency of our latest code for W(8)

214 staples → **20 staples**

→ **12 staples (Latest ver.)**



Introduce AI Coding Agent

We asked

— to improve N2LO = $W(8)$

— and extend it to N3LO = $W(10)$

— check validity automatically & repeat trials until fast code is obtained

https://github.com/WTatsuya2000/HO-HPE

with inputs of

— human-written codes

— list of shape classifications

AI immediately generated a code for N3LO, which is about 10 times faster than the reference program.

AI's Answer

Instead of the Shape-Based algorithm,

Trie algorithm can accelerate the evaluation substantially!

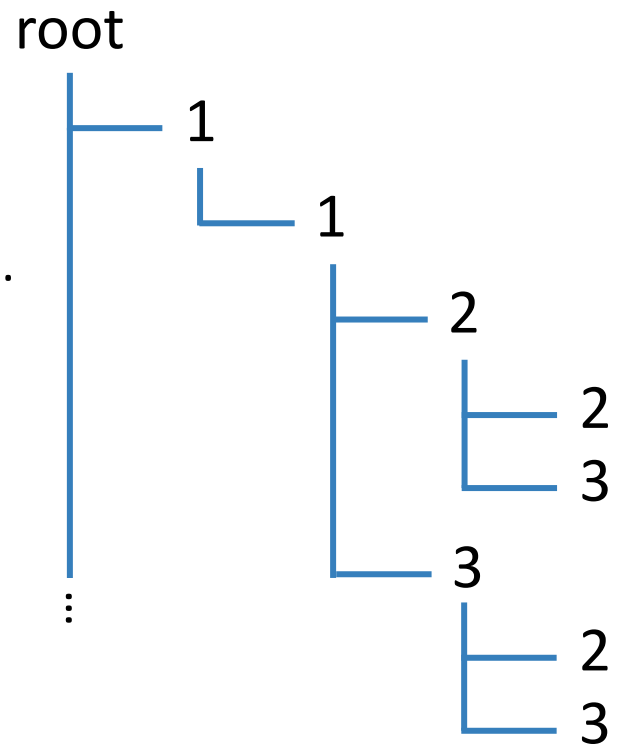
Trie Algorithm

Trie = re**trie**ve

i. Separate loop into prefix and suffix parts

ex.) $1122\bar{1}\bar{1}2\bar{2} \rightarrow [1122][\bar{1}\bar{1}2\bar{2}]$
$1123\bar{1}\bar{1}2\bar{3} \rightarrow [1123][\bar{1}\bar{1}2\bar{3}]$
$1123\bar{1}\bar{1}3\bar{2} \rightarrow [1123][\bar{1}\bar{1}3\bar{2}]$
$1132\bar{1}\bar{1}3\bar{2} \rightarrow [1132][\bar{1}\bar{1}3\bar{2}]$
$1133\bar{1}\bar{1}3\bar{3} \rightarrow [1133][\bar{1}\bar{1}3\bar{3}]$

Trajectories with a common prefix share the same partial matrix product.



ii. Calculate the prefix/suffix partial product per each site

$$(U_1) \rightarrow (U_1U_1) \rightarrow \begin{cases} (U_1U_1U_2) \rightarrow \begin{cases} (U_1U_1U_2U_3) \\ (U_1U_1U_2U_2) \end{cases} \\ (U_1U_1U_3) \rightarrow \begin{cases} (U_1U_1U_3U_2) \\ (U_1U_1U_3U_3) \end{cases} \end{cases}$$

Each trie edge requires only one additional matrix multiplication.

iii. Combine the prefix and suffix parts based on step i.

Improvement by Trie Algorithm

Trie algorithm accelerates Calculation!!

	Reference	Trie-based (Measured)
W(8)	214	28
W(10)	6.3×10^3	460
W(12)	1.8×10^5	8.9×10^3

(Units: Staple)

You can check the code form this QR

For $n = 8$, the shape-based algorithm (12 Staples) is more efficient than the trie-based algorithm.

The trie-based algorithm = Generalization the shape-based algorithm

As length n increases, the trie algorithm becomes more advantageous because the number of reusable common subpaths grows.

Conclusion and Future Works

- ✓ We(Human) enumerated and classified the loops contributing to higher-order HPE terms and developed a reliable reference program.
- ✓ Based on the reference calculation and the physical requirements, AI proposed and implemented an efficient trie-based algorithm.

Applications

- Evaluations of $\mathrm{Tr}\ \mathrm{Ln}\ M$, $\mathrm{Tr}\ M^{-1}$ / 2-point functions / clover Wilson
- Improving the algorithm for HPE coefficients
- AI agent to other lattice perturbative expansions, such as strong coupling expansion

Our conclusion about AI

"AI is a powerful source of algorithmic ideas, but successful research still depends on human researchers to formulate the problem, prepare reliable validation tools, and refine AI-generated results."

Matrix Multiplication of $SU(N_c)$

$$P_n = U_1 U_2 \cdots U_n$$

$$P_1 = U_1 U_2$$
$$P_2 = P_1 U_2$$
$$\vdots$$
$$P_{n-1} = P_{n-2} U_{n-1}$$
$$P_n = P_{n-1} U_n$$

+1 +1 +1

$n - 1$ MMs

$$T_n = \mathrm{Tr}(U_1 U_2 \cdots U_n)$$

$$P_1 = U_1 U_2$$
$$P_2 = P_1 U_2$$
$$\vdots$$
$$P_{n-1} = P_{n-2} U_{n-1}$$
$$P_n = \mathrm{Tr}(P_{n-1} U_n)$$

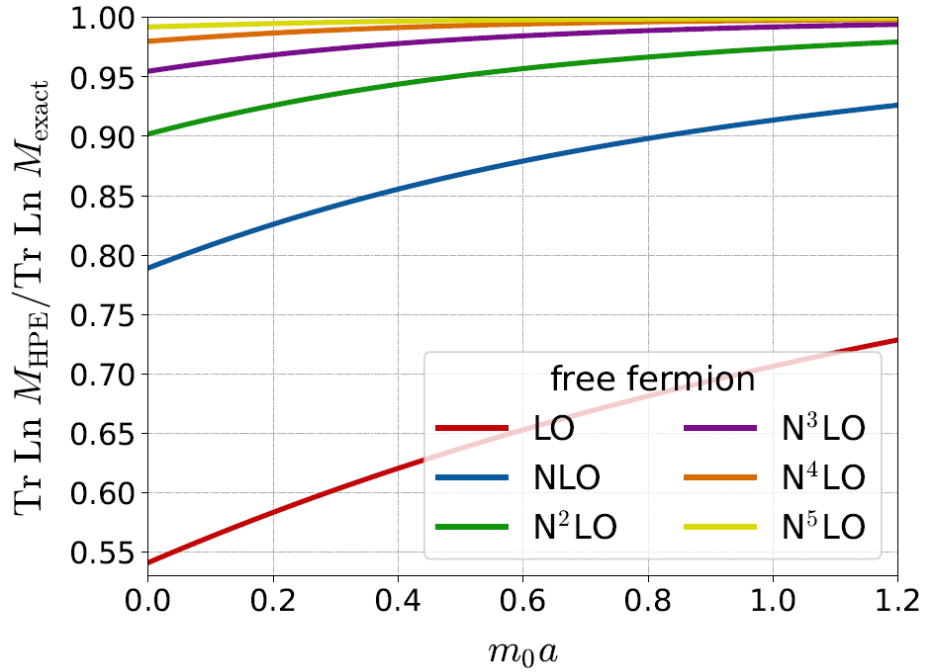
+1 +1 +0

$n - 2$ MMs

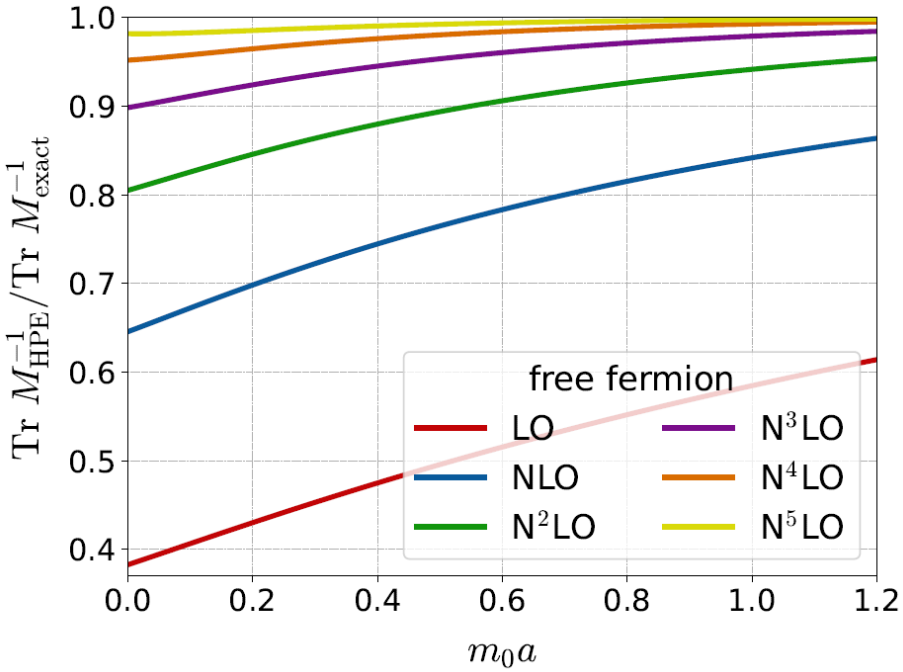
Convergence of HPE

— HPE of free Wilson fermion converges up to the chiral limit.
— Radius of convergence: $\kappa_c = 1/8$.

Convergence of Tr Ln M

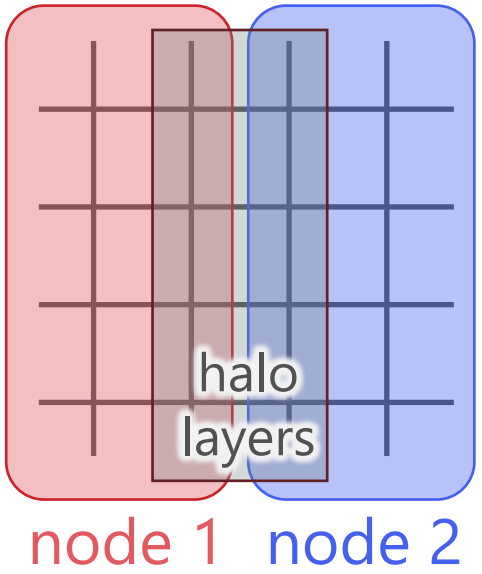


Convergence of Tr M^{-1}



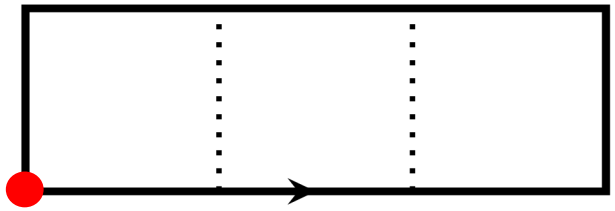
free fermions

Implementation of MPI

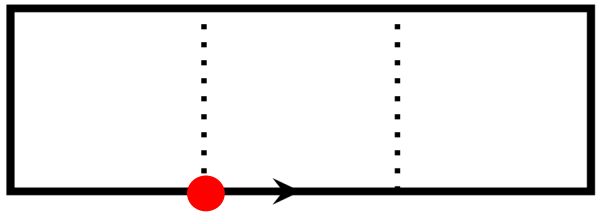


—AI does not understand requirements for MPI environment at first.

However, we introduce the halo layers and propose the optimization od starting point to reduce the number of halo layers.
Finally, AI make optimization code!



3 halo layer



2 halo layer

Why Hopping Parameter Expansion?

How to calculate numerically?

$$\langle \bar{\psi}_x \psi_x \rangle = \mathrm{Tr}[M^{-1}]$$

Two difficulties
① Matrix inverse
② Trace

M: large sparse matrix
$\sim 10^6 \times 10^6$

② "Noisy Estimator"

$$\mathrm{Tr}[M^{-1}] \simeq \Sigma_{i=1}^{R} \langle i|M^{-1}|i\rangle$$

$|i\rangle$: randomly generated normal vectors

① Solve linear equations

$$M|x\rangle = |i\rangle \Rightarrow |x\rangle = M^{-1}|i\rangle$$

$$\Rightarrow \mathrm{Tr}[M^{-1}] \simeq \Sigma_{n=1}^{R} \langle i|x\rangle$$

— Statistical error improves with increasing R.
— However, it requires many matrix inversions.

Wilson Fermion

Dirac fermion

$$L = \bar{\psi}(\gamma^\mu \partial_\mu + m)\psi$$

$$L = \bar{\psi}(\gamma^\mu \partial_\mu + m)\psi - \frac{a}{2}\bar{\psi}\partial^2\psi$$

Discretized Naïve fermion

$$L = \sum_{x,y,\mu} \bar{\psi}(x)\left(\frac{\delta_{x+\hat{\mu},y} - \delta_{x-\hat{\mu},y}}{2a}\gamma^\mu + m\right)\psi(y)$$

$$a\bar{\psi}\partial^2\psi = \bar{\psi}(x)\left(\delta_{x+\hat{\mu},y} + \delta_{x-\hat{\mu},y} - 2\delta_{x,y}\right)\psi(y)$$

$$L = \sum_{x,y,\mu} \bar{\psi}(x)\left((m+4)\delta_{x,y} - \sum_{\mu}\left[\frac{1-\gamma_\mu}{2}\delta_{x+\hat{\mu},y} + \frac{1+\gamma_\mu}{2}\delta_{x-\hat{\mu},y}\right]\right)\psi(y)$$

$$\psi \to \frac{1}{\sqrt{m+4}}\psi$$

$$\kappa = \frac{1}{2m+8}$$

$$L = \sum_{x,y} \bar{\psi}(x)\left(\delta_{x,y} - \kappa\sum_{\mu}\left[(1-\gamma_\mu)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)\delta_{x-\hat{\mu},y}\right]\right)\psi(y)$$