Gauge-equivariant multigrid neural networks

Christoph Lehner and Tilo Wettig (University of Regensburg)
arXiv:2302.05419 [hep-lat]
arXiv:2304.10438 [hep-lat]

Lattice 2023, Fermilab, July 31, 2023
Introduction

Parallel-transport convolution layers

Wilson-clover Dirac operator

High-mode preconditioners

Low-mode preconditioners
  Standard construction
  Gauge-equivariant construction

Multigrid preconditioners

Summary and outlook
Introduction
Preconditioning

• In lattice QCD, wall-clock time is typically dominated by solution of Dirac equation
  \[ Du = b \]
  • Usually done by an iterative solver (here, GMRES)

• Time to solution is determined by condition number of Dirac matrix
  • Condition number increases dramatically in continuum limit and for physical quark mass
  • Thus number of iterations also increases dramatically (“Critical slowing down”)

• Way out: **Preconditioning**
  • Find a preconditioner \( M \) such that \( M \approx D^{-1} \)
  • Define \( v = M^{-1}u \) and use
    \[ DMM^{-1}u = (DM)v = b \]
    to solve for \( v \) with preconditioned matrix \( DM \) (smaller condition number)
  • Then \( u = Mv \)
Measure of performance

\[
\text{Iteration count gain} = \frac{\text{Iteration count without preconditioner}}{\text{Iteration count with preconditioner}}
\]

- Iteration count refers to outer solver (here, GMRES)
• Iterative solution of $Du = b$

$$u_{k+1} = f(D, b, u_k) \quad \text{with} \quad u_k \to u \quad \text{(true solution)}$$

• Residual

$$r_k = b - Du_k \quad \text{with} \quad r_k \to 0$$

• Residual $r_k$ can formally be expanded in the eigenmodes $|n\rangle$ of $D$

→ Preconditioner should reduce low- and high-mode contributions to $r_k$

• State-of-the-art algorithms (multigrid) are designed to do this

• We follow this paradigm, but here we learn the preconditioner
Multigrid in a nutshell

- Multigrid has two components
  - **Smoother**: Reduces error from high modes
  - **Coarse-grid correction**: Reduces error from low modes
    - “Restriction” to a coarse grid
    - Approximate solution of Dirac equation on coarse grid
    - “Prolongation” of solution vector from coarse to fine grid
- Can be done on multiple levels

- Multigrid setup
  - The art of multigrid: How to construct suitable restriction and prolongation operators?
  - Observation: Low eigenmodes are “locally coherent”  
    Lüscher, arXiv:0706.2298 [hep-lat]  
    (i.e., they are locally well approximated by a relatively small number of vectors)
    → Construct vectors that approximately span the near-null space
      Block these vectors to define the restriction operator (and use $P = R^\dagger$)
  - Setup is expensive but needs to be done only once per gauge-field configuration
    (can then be reused for multiple RHS)
1. Multigrid algorithms in lattice QCD
   - Brannick, Brower, Clark, Osborn, Rebbi
   - R. Babich et al.
   - Frommer et al.
   - Boyle
   - Brannick et al.
   - Brower, Clark, Strelchenko, Weinberg
   - Brower, Clark, Howarth, Weinberg

2. Neural networks for multigrid (but not for gauge theories), e.g.,
   - Katrutsa, Daulbaev, Oseledets
   - He & Xu
   - Greenfeld, Galun, Basri, Yavneh, Kimmel
   - Eliasof, Ephrath, Ruthotto, Treister
   - Huang, Li, Xi
3. Gauge-equivariant neural networks (but not for solving Dirac equation), e.g.,

- Cohen, Weiler, Kicanaoglu, Welling
- Finzi, Stanton, Izmailov, Wilson
- Luo, Carleo, Clark, Stokes
- Kanwar et al.
- Boyda et al.
- Favoni, Ipp, Müller, Schuh
- Abbott et al.
- Bacchio, Kessel, Schäfer, Vaitl
- Aronsson, Müller, Schuh

4. Neural-network preconditioners for Schwinger model

- Calì et al.
5. Gauge-equivariant multigrid setup and coarse gauge fields (late 1980s/early 1990s)
   - Amsterdam group (Hulsebos, Smit, Vink)
   - Israel group (Ben-Av et al.)
   - Boston group (Brower et al.)
   - Hamburg group (Kalkreuter et al.)
Parallel-transport convolution layers
Parallel transport

- Consider a field \( \varphi(x) \) with \( x \in S \) (space-time lattice, \( \text{dim} = d \)) and \( \varphi \in V_I = V_G \otimes V_{\tilde{G}} \) (gauge space: \( V_G = \mathbb{C}^N \), non-gauge space: \( V_{\tilde{G}} = \mathbb{C}^{\tilde{N}} \))

- Also consider an \( \text{SU}(N) \) gauge field \( U_\mu(x) \) acting on \( V_G \)

- Define the parallel-transport operator for a path \( p = p_1, \ldots, p_{n_p} \) with \( p_i \in \{\pm 1, \ldots, \pm d\} \)

\[
T_p = H_{p_{n_p}} \cdots H_{p_2} H_{p_1}
\]

with

\[
H_\mu \varphi(x) = U_\mu^\dagger(x - \hat{\mu}) \varphi(x - \hat{\mu})
\]

- \( H_\mu \) transports information by a single hop in direction \( \hat{\mu} \)
- \( H_\mu \) acts on field; new field \( H_\mu \varphi \) is evaluated at \( x \)
- Example: \( T_p = H_{-1} H_{-2} H_{-1} H_2 H_2 \)
Gauge equivariance

• A gauge transformation by \( \Omega(x) \in SU(N) \) acts in the usual way

\[
\varphi(x) \rightarrow \Omega(x)\varphi(x)
\]

\[
U_\mu(x) \rightarrow \Omega(x)U_\mu(x)\Omega^\dagger(x + \hat{\mu})
\]

• Such gauge transformations commute with \( T_p \) for any path \( p \)

\[
T_p \varphi(x) \rightarrow \Omega(x)T_p \varphi(x)
\]

• This is an example of **gauge equivariance** (a.k.a. gauge covariance):

An object (here: \( \varphi \)) and the transformed object (here: \( T_p \varphi \))

transform in the same way under a gauge transformation.

• Building gauge equivariance into the model implies that the model does not have to

learn the gauge symmetry \( \rightarrow \) **Same expressivity with fewer weights**
Parallel-transport convolutions

- Parallel-transport convolution layer and local parallel-transport convolution layer

\[ \psi_a(x) \overset{PTC}{=} \sum_b \sum_{p \in P} W_{ab}^p T_p \varphi_b(x) \]

\[ \psi_a(x) \overset{LPTC}{=} \sum_b \sum_{p \in P} W_{ab}^p(x) T_p \varphi_b(x) \]

- \( a \) = output feature index
- \( b \) = input feature index
- \( P \) = set of paths
- \( W_{ab}^p \) acts on \( V_\tilde{G} \) (here: 4 × 4 spin matrix)
- Elements of \( W \): trainable layer weights

- Layers are gauge-equivariant
- No activation function since we want to learn a linear preconditioner
- Graphical conventions
  - Feature = Plane
  - Layer = Paths + Arrow
Communication avoidance

- On machines with many nodes, subvolumes are assigned to different MPI processes.
- We also consider models where no information is communicated between subvolumes (by setting the links $U_\mu(x)$ connecting subvolumes to zero).
- We find that the performance of these models (in terms of iteration count gain) is only slightly worse compared to those with communication.
  → Overall wall-clock time could be lower since no time is spent on communication.
Wilson-clover Dirac operator
Dirac operator

- The Wilson Dirac operator can be written in terms of single hops:

\[
D_W = \frac{1}{2} \sum_{\mu=1}^{4} \gamma_\mu (H_-^\mu - H_+^\mu) - \frac{1}{2} \sum_{\mu=1}^{4} (H_-^\mu + H_+^\mu - 2) + m
\]

- For Wilson-clover, consider closed paths with four hops and define

\[
Q_{\mu \nu} = H_-^\mu H_-^\nu H_+^\mu H_+^\nu + H_-^\nu H_+^\mu H_+^\nu H_-^\mu + H_+^\nu H_-^\mu H_-^\nu H_+^\mu + H_+^\mu H_+^\nu H_-^\mu H_-^\nu
\]

Then

\[
D_{WC} = D_W - \frac{c_{sw}}{4} \sum_{\mu, \nu=1}^{4} \sigma_{\mu \nu} F_{\mu \nu}
\]

with

\[
F_{\mu \nu} = \frac{1}{8} (Q_{\mu \nu} - Q_{\nu \mu}) \quad \sigma_{\mu \nu} = \frac{1}{2} (\gamma_\mu \gamma_\nu - \gamma_\nu \gamma_\mu)
\]
Numerical details and eigenvalue spectrum

- $V = 8^3 \times 16$, $\beta = 6.0$ (pure gauge), $c_{SW} = 1$, periodic boundary conditions for all fields
- Quark mass $m$ is tuned so that $D_{WC}$ is near criticality (i.e., real part of smallest nonzero eigenvalue $\approx 0$)
- $\rightarrow$ Solution of $Du = b$ is challenging problem

\[
Q = 0 \quad (m = -0.6)
\]

\[
Q = 1 \quad (m = -0.5645)
\]
High-mode preconditioners
Model setup and training strategy

- High-mode part of Dirac spectrum is related to short-distance behavior
  → Expect one or two layers with small number of hops to show gain in iteration count
- Consider a linear model $M$ mapping a vector $x$ to $Mx$
- Supervised learning approach with training step as follows:
  - Pick random vector $v$ from Gaussian distribution (mean zero, standard deviation 1)
  - Compute training tuple $(D_{wc}v, v)$ and optimize cost function
    \[ C = |MD_{wc}v - v|^2 \]
    → Model learns to map $D_{wc}v$ to $v$ (and hence $M \approx D_{wc}^{-1}$)
  - Optimizer is Adam  
  - Derivatives w.r.t. model weights computed using backpropagation
- Training data set is unbounded in size → No need to add a regulator
- Cost function is dominated by high modes

Kingma & Ba, arXiv:1412.6980 [cs.LG]
Models chosen for high-mode preconditioner

- **One layer, one hop** (i.e., 9 paths)
  \[ T_0 = 1, \ T_1 = H_1, \ T_2 = H_2, \ T_3 = H_3, \ T_4 = H_4, \ T_5 = H_{-1}, \ T_6 = H_{-2}, \ T_7 = H_{-3}, \ T_8 = H_{-4} \]

- **One layer, two hops:** extend the above by 56 two-hop paths
  \[ H_a H_b \quad \text{with} \quad a, b \in \{-4, -3, -2, -1, 1, 2, 3, 4\} \quad (a \neq -b) \]

- **“Deep” network of two one-hop layers:**
  - \( 1 \rightarrow 1 \rightarrow 1 \): Two successive layers with one hop each
  - \( 1 \rightarrow 2 \rightarrow 1 \): Two output features in first layer, two input features in second layer

- **PTC** (layer weights constant) and **LPTC** (layer weights depend on \( x \))

- **Communication avoidance:** \( U_\mu(x) \equiv 0 \) between subvolumes of size \( 4^3 \times 8 \)
Results for high-mode preconditioner (one layer, one hop)

- No gain from LPTC (and they require more training)
- Communication-avoiding version only slightly worse (could be amortized)
• $1 \rightarrow 2 \rightarrow 1$ model performs best (and gives $\sim$ twice the gain of 1 layer/1 hop model)
• Since layers are linear, deep models are not more expressive than shallow models with same number of hops (but easier to train b/o smaller number of weights)
→ 2-hop model should reach similar performance with improved training procedure
- No retraining required for (i) different configuration from same ensemble, (ii) configuration with different $\beta$, (iii) different mass
- $m = -0.55$ is not tuned to criticality $\rightarrow$ Easier initial problem $\rightarrow$ Smaller gain
- Performance varies slightly between configurations
Low-mode preconditioners
Possible approaches

- **Low-mode part** of Dirac spectrum is related to **long-distance behavior**
  → Need deep network of (L)PTC layers to propagate information over long distances

- Alternative: Use multigrid paradigm
  - Define coarse version of the lattice
  - Define restriction and prolongation operations (= layers)
  - Preserve low-mode part of Dirac spectrum
Low-mode preconditioners

Standard construction
Standard approach: No gauge degrees of freedom on the coarse grid

• Define a coarse grid $\tilde{S}$ with fields $\tilde{\varphi}(y)$, where $y \in \tilde{S}$ and $\tilde{\varphi} \in \tilde{V}_I$
• $\tilde{V}_I$ has no gauge degrees of freedom $\rightarrow$ No gauge transformations on $\tilde{V}_I$
• $B = \text{block map from } \tilde{S} \text{ to } S$ (i.e., sites $B(y)$ on fine grid correspond to $y$ on coarse grid)
• Restriction and prolongation layer (with $R = P^*$)

$$\tilde{\psi}(y) \overset{RL}{=} \sum_{x \in B(y)} W(y, x) \varphi(x)$$

$$\psi(x) \overset{PL}{=} W(y, x)^\dagger \tilde{\varphi}(y)$$
Restriction and prolongation layers

- Find $s$ vectors in the near-null space of $D$

$$Du_i \approx 0 \quad (i = 1, \ldots, s)$$

- Apply GMRES for $D$ with source vector $= 0$ and random initial guess (solve to $10^{-8}$)
  - This removes high-mode components and leaves linear combination of low modes
- Block the $u_i$
  - One site $y \in \tilde{S}$ corresponds to a set of sites (or block) $B(y) \in S$
  - Blocked vector $u^y_i$ lives on the sites of $B(y)$
- Orthonormalize the $u^y_i$ within each block $\rightarrow \tilde{u}^y_i$
- Then the prolongation map is defined by

$$W(y, x)^\dagger = \sum_{i=1}^{s} \tilde{u}^y_i(x)\hat{e}_i^\dagger$$

no trainable weights

with $x \in B(y)$ and $\hat{e}_i$ the canonical unit vectors of $\tilde{V}_I$
Model setup and training strategy

• Coarse-grid operator is defined as

\[ \tilde{D} = RD_{WC}P \]

with \( R \) and \( P \) defined by restriction and prolongation layers

• Now need approximate solution of Dirac equation involving \( \tilde{D} \)

• Coarse-grid model for preconditioner \( \tilde{M} \) contains single LPTC layer with zero- and one-hop paths and gauge fields replaced by \( 1 \) (layer is denoted by cLPTC)

• Same training strategy as before, with cost function

\[ C = |\tilde{M} \tilde{D}v - v|^2 \]
Results for low-mode preconditioner (cLPTC layer)

- Iteration count gain refers to inversion of $\tilde{D}$ (we use $\tilde{S} = 2^3 \times 4$ and $s = 12$)
- Longer training period compared to high-mode preconditioner
- Transfer learning works with moderate retraining
Low-mode preconditioners

Gauge-equivariant construction
Now: Explicit gauge degrees of freedom on the coarse grid

- Same coarse grid $\tilde{S}$ as before, but now $\tilde{\varphi}(y) \in V_G \otimes \tilde{V}_G$ ($V_G$ = same local gauge space as on fine grid)
- Define a reference site $B_r(y) \subset B(y)$ on the fine grid
- Goal: Find restriction and prolongation layers such that $\tilde{\varphi}(y) \rightarrow \tilde{\Omega}(y)\tilde{\varphi}(y)$ under gauge transformations $\Omega$, where

$$\tilde{\Omega}(y) = \Omega(B_r(y))$$
Restriction and prolongation layers

- Define RL/PL by pooling and subsampling layers
  \[ \text{RL} = \text{SubSample} \circ \text{Pool} \quad \text{PL} = \text{Pool}^\dagger \circ \text{SubSample}^\dagger \]

- Pooling layer
  \[
  \text{Pool} \varphi(x) = \sum_{q \in Q} W_q(x) T_q \varphi(x)
  \]
gauge-invariant weights (now trainable)

  with \( q = (p, \tilde{U}) \), path \( p \), gauge field \( \tilde{U} \), \( T_q = T_p(\tilde{U}) \), and \( W_q(x) \in \text{End}(V_{\tilde{G}}) \) (spin matrices)
  (in practice, we use a variety of differently smeared links \( \tilde{U} \))

- Subsampling layer
  \[
  \text{SubSample} \varphi(y) = \varphi(B_r(y))
  \]
Training setup: How to train RL/PL?

- Obvious idea: Train $\text{PL} \circ \text{RL}$ as an autoencoder that preserves the low modes
  - Use cost function $C = |\text{PL} \circ \text{RL} v_\ell - v_\ell|^2$ with fine-grid vectors $v_\ell$ from near-null space
  - Result: Did not perform well in multigrid preconditioner!
- What was missing?
  - $\text{PL} \circ \text{RL}$ should also project high eigenmodes to zero
  - Also encourage $\text{RL} \circ \text{PL} = 1$ (so that $P = \text{PL} \circ \text{RL}$ is proper projection operator with $P^2 = P$)
- Combined cost function

$$C = |\text{PL} \circ \text{RL} v_\ell - v_\ell|^2 + |\text{PL} \circ \text{RL} v_h - P_\ell v_h|^2 + |\text{RL} \circ \text{PL} v_c - v_c|^2$$

- $v_h$ and $v_c$ are random vectors on fine and coarse grid, respectively
- $P_\ell$ is blocked low-mode projector

$$P_\ell = W^\dagger W \quad \text{with} \quad W(y, x)^\dagger = \sum_{i=1}^s \bar{u}_i^y(x) \hat{e}_i^\dagger$$

- Still costly since we need near-null space vectors, but see Outlook
For gauge-equivariant coarse layers we need coarse gauge field

- **Option 1:** Plain coarse-gauge-field construction
  - Let $y$ and $y'$ be neighboring points on the coarse grid with $B_r(y') - B_r(y) = b\hat{\mu}$
  - The corresponding coarse-grid gauge field is then
    \[ \tilde{U}_\mu(y) = U_\mu(B_r(y)) \cdots U_\mu(B_r(y) + (b - 1)\hat{\mu}) \]

- **Option 2:** Galerkin coarse-gauge-field construction
  \[ \tilde{U}_\mu(y) = \tilde{D}(y, y + \mu) \quad \text{with} \quad \tilde{D} = RL \circ D_{WC} \circ PL \]

- Both options transform correctly under gauge transformations (on coarse grid)

- Coarse-grid model for preconditioner $\tilde{M}$ similar to standard version but with coarse gauge fields (instead of $1$)
Multigrid preconditioners
• **Combine the high- and low-mode models** to learn a model $M$ that approximates the short- and long-distance features of $D^{-1}$

• First create a short-distance model that accepts a second input feature (initial guess)
  • Model plays role of *smoother* in multigrid method
  • Initial guess from long-distance model acting on coarse grid

https://summerofhpc.prace-ri.eu/multithreading-the-multigrid-solver-for-lattice-qcd
Smother

• Recall: Iterative solver finds a sequence of $u_k$ that approximately solve $Du = b$ (exact solution for large $k$)
• Assume we have a high-mode model $M_h$ that approximates $D^{-1}$
• Smoother maps the tuple $(u_k, b)$ to $u_{k+1}$

\[
\begin{align*}
    u_{k+1} &= (1 - M_h D)u_k + M_h b \\
    &= u_k + M_h (b - Du_k)
\end{align*}
\]

("iterative relaxation approach" or "defect correction" with defect $b - Du$)
Smother model setup and training strategy

• In smoother iteration

\[ u_{k+1} = u_k + M_h(b - Du_k) \]  \hspace{1cm} (\star)

both \( D \) and high-mode model \( M_h \) can be represented by (L)PTC layers

→ Train a model \( M_s \) to map \((u_k, b)\) to a \( u_{k+r} \) (with \( r \in \mathbb{N}^+ \))

• Model must have two input features and one output feature
• Every smoother iteration (\( \star \)) corresponds to two (L)PTC layers
  → Construct \( M_s \) using \( 2r \) successive layers (here with up to one hop each)
• We use \( r = 2 \) since it performed better
  than \( r = 1 \) in full multigrid model

• Cost function

\[ C = |M_s(u_k, b) - u_{k+r}|^2 \]

For training, use random vectors \( u_k, b \) and \( u_{k+r} \) given by (\( \star \))
• Iteration count gain from using $M_s$ as preconditioner for $Du = b$ with initial guess zero
• Performance is $\sim$ twice that of $M_h$ with 1 layer/1 hop (since $r = 2$)
• Trained PTC model is used as initial weights for LPTC model (but no benefit from LPTC)
• Duplicate the input feature and preserve one copy for smoother
• Restrict other copy to coarse grid and apply our coarse-grid model
• Prolongate result to fine grid
• Combine copy of initial feature and result of coarse-grid model to two input features for smoother (= last four layers)
• Additional multigrid levels: Recursively replace coarse-grid layer by entire model
• Duplicate the input feature and preserve one copy for smoother
• Restrict other copy to coarse grid and apply our coarse-grid model
• Prolongate result to fine grid
• Combine copy of initial feature and result of coarse-grid model to two input features for smoother (= last four layers)
• Additional multigrid levels: Recursively replace coarse-grid layer by entire model
Training strategy for multigrid model

- First train layer weights of individual models
- Performance can be further improved by continued training with cost function

\[ C = |M b_h - u_h|^2 + |M b_\ell - u_\ell|^2 \]

- \( b_h = D_{WC} v_1, u_h = v_1, b_\ell = v_2, u_\ell = D_{WC}^{-1} v_2 \)
- \( v_1 \) and \( v_2 \) are random vectors with \( |b_h| = |b_\ell| = 1 \)
Results: Critical slowing down (CSD) for $Q = 1$

- Iteration count of GMRES to $10^{-8}$ precision with and without preconditioner
- CSD eliminated by standard multigrid model and model with Galerkin gauge fields
- Small remnants of CSD with plain coarse gauge fields
Summary and outlook
• We reformulate the problem of constructing a (multigrid) preconditioner in the language of gauge-equivariant neural networks.

• We find that such networks can learn the general paradigms of multigrid, significantly reduce the iteration count of the outer solver, and eliminate critical slowing down. Both for standard and gauge-equivariant construction of restriction/prolongation.

• Transfer learning: If we change the gauge-field configuration or system parameters like $\kappa$ and $\beta$, only very little or no extra training is needed.

• We can implement communication avoidance naturally.

• We provide a flexible implementation interface (GPT) for experimentation and further studies.
Outlook

• Setup (determination of spin matrices for restriction/prolongation layers) currently still costly because near-null space is needed
  • Future: Remove this cost by gauge-invariant models with these spin matrices as output
  • Use energy density, topological-charge density, Wilson loops
  • Useful for ensemble generation (where setup cost cannot be amortized)

• Apply our methods to Dirac operators whose spectrum encircles the origin (e.g., DWF)

• Benchmarking on large lattices and comparison to state-of-the-art multigrid (larger volumes should lead to larger iteration count gain)
Backup slides
• Performance greatly improved over individual high-/low-mode models
• Continued training converges very quickly
• Transfer learning works again after brief retraining
More details on the pooling layer

- Gauge field $\tilde{U}$ in $T_p(\tilde{U})$ needs to satisfy

$$\tilde{U}_\mu(x) \rightarrow \Omega(x)\tilde{U}_\mu(x)\Omega^{\dagger}(x + \mu)$$

In practice, we use a variety of differently smeared links

- Complete set of paths $P$ transports every element of $B(y)$ exactly once to $B_r(y)$

$$|P| = |B(y)|$$

- $\tilde{\varphi} = \text{RL} \varphi$ yields $\tilde{\varphi}(y) \rightarrow \tilde{\Omega}(y)\tilde{\varphi}(y)$ under gauge transformations $\varphi(x) \rightarrow \Omega(x)\varphi(x)$
• Need prescription for $q$ in

$$\text{Pool } \varphi(x) = \sum_{q \in \mathbb{Q}} W_q(x) T_q \varphi(x)$$

with $q = (p, \tilde{U})$, path $p$, gauge field $\tilde{U}$, $T_q = T_p(\tilde{U})$

• For fixed $i$, we define paths $p^{(ij)}$ that connect all elements of $B(y)$, enumerated by $j = 1, \ldots, |B(y)|$, to the reference site $B_r(y)$. For different $i$ we use different prescriptions for the paths $p^{(ij)}$, and then use the couples $q_{ij} = (p^{(ij)}, \tilde{U}^{(i)})$.

• We define four different prescriptions $\hat{p}_1, \ldots, \hat{p}_4$ (depth first, breadth first, lexicographic, reverse lexicographic)

and set $p^{(ij)} = p^{(j)}_{i \mod 4}$
• Concretely, we use 9 different gauge fields $\bar{U}^{(i)}$ with $i = 1, \ldots, 9$. We construct the $\bar{U}^{(i)}$ by applying $i(i - 1)/2$ steps of $\rho = 0.1$ stout smearing to the unsmeared gauge fields $U$. Smearing radius proportional to $\sqrt{i(i - 1)}$.

• Hence we have 9 different spin-matrix fields $W_1(x), \ldots, W_9(x)$.

• In practice, it is sufficient to use the same weights in $PL$ and $RL$ so that $PL = RL^\dagger$. Found no benefits from general case.

• Coarse-grid size $2^3 \times 4$