PROBING FOR TRACE ESTIMATION OF A PERMUTED MATRIX INVERSE CORRESPONDING TO A LATTICE DISPLACEMENT*

HEATHER SWITZER†, ANDREAS STATHOPOULOS‡, ELOY ROMERO§, JESSE LAEUCHLI¶, AND KOSTAS ORGINOS ‖

Abstract. Probing [26] is a general technique that is used to reduce the variance of the Hutchinson stochastic estimator for the trace of the inverse of a large, sparse matrix A [19]. The variance of the estimator is the sum of the squares of the off-diagonal elements of A^{-1}. Therefore, this technique computes probing vectors that when used in the estimator they annihilate the largest off-diagonal elements. For matrices that display decay of the magnitude of $|A^{-1}_{ij}|$ with the graph distance between nodes i and j, this is achieved through graph coloring of increasing powers A^p [25]. Equivalently, when a matrix stems from a lattice discretization, it is computationally beneficial to find a distance-p coloring of the lattice. In [23] a hierarchical coloring was proposed so that p can be increased at runtime as needed without discarding previous work.

In this work, we study probing for the more general problem of computing the trace of a permutation of A^{-1}, say PA^{-1}. The motivation comes from Lattice QCD where we need to construct "disconnected diagrams" to extract flavor-separated Generalized Parton functions. In Lattice QCD, where the matrix has a 4D toroidal lattice structure, these non-local operators correspond to a PA^{-1} where P is the permutation relating to some displacement $\vec{k}$ in one or more dimensions. We focus on a single dimension displacement (k) but our methods are general. We show that probing on A^p or $(PA)^p$ do not annihilate the largest magnitude elements. To resolve this issue, our displacement-based probing works on PA^p using a new coloring scheme that works directly on appropriately displaced neighborhoods on the lattice. We prove lower bounds on the number of colors needed, and study the effect of this scheme on variance reduction, both theoretically and experimentally on a real-world Lattice QCD calculation. We achieve orders of magnitude speedup over the un-probed or the naively probed methods.

Key words. trace of the inverse, Probing, Lattice QCD, lattice, sparse matrix, Hadamard, torus

AMS subject classifications. 05B20, 15A15, 65C05, 65F50, 68R10, 81V05

1. Introduction.

The approximation of the trace of a matrix function, $f(A)$, of a large sparse matrix A is a computationally challenging problem. Commonly used functions are the A^{-1} and $\log A$ (which is used to find the matrix determinant). In this paper we focus on $f(A) = A^{-1}$ which has many applications in statistics [19], quantum Monte Carlo [1], and data mining [8]. Our motivating application comes from lattice quantum chromodynamics (LQCD). In LQCD, the trace of the inverse of an operator discretized on a symmetric, four-dimensional, toroidal lattice representing space-time is often used to analyze the interactions, properties, and structure of hadrons on a subatomic scale [18]. The trace computations are part of larger scale Monte Carlo simulations and therefore do not require high accuracy but must induce no statistical bias.

*This work is supported by SURA under grant number C2019-FEMT-002-05, by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by Deakin University.

†Department of Computer Science, College of William & Mary, Williamsburg, VA (hmswitzer@email.wm.edu).

‡Department of Computer Science, College of William & Mary, Williamsburg, VA (andreas@wm.edu).

§Jefferson Laboratory, Newport News, VA (eromero@jlab.org).

¶School of Information Technology, Deakin University, Geelong, Victoria 3220, Australia (j.laeuchli@deakin.edu.au).

‖Jefferson Laboratory, Newport News, VA (knorgi@wm.edu).

Effective methods for computing $\mathbf{Tr}(A^{-1})$ exist for smaller matrices where sparse factorizations are possible [4, 9], but as the size of A increases they become computationally infeasible and stochastic estimation is the only alternative. A widely used method for this is the Hutchinson's trace estimator [19] which takes the form

$$\mathbf{Tr}(A^{-1}) \approx \frac{1}{s}\sum_{i=1}^{s} z_i^T A^{-1} z_i, \tag{1.1}$$

where z_i are s i.i.d. random noise vectors (RNV). The computational complexity therefore is dominated by the solution of the linear systems with some iterative method. The RNVs are chosen to have a Rademacher distribution, where each element is equal to ± 1 with probability 0.5. It is known that for this choice the estimator has variance

$$\mathrm{Var}(z^T A^{-1} z) = 2(\|A^{-1}\|_F^2 - \sum_{i=1}^{N}(A_{i,i})^2), \tag{1.2}$$

which is minimum over all random distributions for z_i when A is real [5]. The variance is the same for complex matrices, which is the case in LQCD, when $\mathbb{Z}_4$ Rademacher vectors are used, i.e., vectors with $\pm 1, \pm i$ values with probability 0.25. The variance formula shows that large off-diagonal elements contribute significant errors to the estimator and cause slow convergence. Many techniques have been introduced and studied to reduce the variance of the estimator by choosing vectors that better take advantage of the structure of the matrix [5, 8, 17, 26, 28].

One such technique is classical probing (CP). Probing is a general technique that uses graph coloring of the graph of an adjacency matrix A to construct structurally orthogonal probing vectors to extract specific non-zero entries of the matrix. For example, multiplying a diagonal matrix with a vector of ones recovers its diagonal. Similarly, when the adjacency matrix of a graph is k colorable, we can also recover the diagonal by multiplying the matrix with k vectors, each vector having ones in rows with the same color and zero elsewhere. In numerical optimization probing is applied on the graph of A^2 in order to compute the Hessian [14]. For trace estimation, CP constructs probing vectors from a coloring of the graph of A^p or equivalently the distance-p coloring of the graph of A, where $p \in \mathbb{Z}_+$ [26]. The idea is that for many sparse matrices the elements of A_{ij}^{-1} display a Green's function decay in magnitude with the distance between nodes i and j. Although A^{-1} is not sparse, using these probing vectors in the estimator removes from the variance (1.2) all elements (edges) of distance-p neighbors. A drawback of PC is that if a coloring for a certain distance p does not produce the required variance reduction, a higher distance coloring cannot reuse the quadratures computed with the previous probing vectors.

Hierarchical Probing (HP) was introduced to address the reuse issue [23, 20]. HP assigns colors to nodes in a hierarchical way so that two nodes that receive the same color for some distance p will never share the same color in higher distances. The technique also provided a computationally inexpensive way to produce a distance-p coloring for large p when the matrix graph is a regular, toroidal lattice. This toroidal structure appears in LQCD matrices which is also the focus of the current paper.

Deflation has also been used as a variance reduction technique [22, 12]. While probing techniques capture large elements from relatively small lattice distances, the low rank approximation of A^{-1} using the lowest magnitude singular triplets of A typically captures a large part of of the magnitude of A^{-1} at long distances. Thus,

the two approaches are complementary and, when used in tandem, can significantly accelerate the Monte Carlo estimator.

In this paper we extend probing for computing the trace of a permutation of A^{-1}. The motivation comes from LQCD computations of the flavor-separated Generalized Parton functions (GPDs) where the so-called "disconnected diagrams" need to be calculated [13, 3]. This translates to the need to find the sum of certain off-diagonal elements of A^{-1} that correspond to a displacement along the z dimension of the four-dimensional (space-time) LQCD lattice. This is a non-symmetric permutation of the rows of A^{-1}, where the index of a node x no longer refers to $[x_1, x_2, x_3, x_4]$, but instead $[x_1, x_2, x_3 + k, x_4]$. The associated trace problem is more challenging because the variance for PA^{-1} now includes the main diagonal A^{-1} which is of much larger magnitude than the one of PA^{-1}.

We propose an extension of CP that modifies a greedy coloring algorithm to consider not the node's original neighborhood but the neighborhood of its displacement. The idea applies to any permutation matrix and can be performed in a hierarchical way if desired. For toroidal lattices with z-displacement we prove lower bounds on the number of colors and study the effect of the algorithm on variance reduction both theoretically and with LQCD experiments. The method results in orders of magnitude variance reduction over conventional probing methods.

The rest of the paper is organized as follows: Section 2 introduces notation and discusses previous variance reduction techniques. Section 3 introduces the coloring algorithm with displacements, and studies its properties theoretically. Experimental result are shown in Section 4. Conclusions and some open questions are given in Section 5.

2. Background.

In this paper we seek the trace of PA^{-1}, where P is a permutation matrix, and A is a non-singular matrix of dimension N which can be complex valued as in the case of LQCD. Although our main idea applies to any P and A, the algorithm and the analysis is relevant to matrices stemming from a regular lattice discretization. Letting $\mathbb{Z}_n$ be the multiplicative group of integers modulo n, then a d-dimensional toroidal lattice is described as

$$\mathbb{Z}_D^d = \mathbb{Z}_{D_1} \times ... \times \mathbb{Z}_{D_d}, \tag{2.1}$$

where D_i is the size of dimension i. Two lattice nodes x and y are connected by an edge if their coordinate vectors $[x_1, ..., x_d]$ and $[y_1, ..., y_d]$, satisfy $\|x - y\|_1 = 1$ (in a modulo sense). In LQCD, the lattice represents the 4 dimensional space-time.

Variance reduction techniques for the Hutchinson trace estimator focus around two approaches; one derives an approximation to A^{-1} such as from deflation or preconditioning which we briefly address in Subsection 2.4; the other replaces the Rademacher vectors with ones that better take advantage of the structure of the matrix. Orthogonal columns of the Hadamard or Fourier matrix have been proposed [8] which can systematically annihilate specific diagonals of the matrix and thus reduce the variance in (1.2). The variance reduction is monotonic with the number of columns used but this method works no better than using solely RNVs as the patterns of diagonals removed are not typically the heaviest variance-contributing diagonals of A^{-1}. The following methods attempt to capture these heaviest elements directly.

2.1. Classical Probing.

The inverse of an $N \times N$ non-singular matrix A where $\|A\| < 1$ can be represented by the Neumann series $A^{-1} = \sum_{p=0}^{\infty}(I - A)^p$ [25]. As a result of this series being convergent, higher powers of $(I - A)^p$ provide a smaller

contribution to A^{-1}. Many matrices from Partial Differential Equations, Lattice QCD, and other applications display a significant decay in the elements of $(I-A)^p$ for larger values of p, further motivating the idea of Probing [6, 26]. In LQCD, in particular, a basic form of probing was first used in [27] and has become more popular with the name dilution since [11].

The CP (classical probing) method is not used to directly approximate A^{-1}, but instead to locate its largest elements using graph coloring. Based on the decay principle above and since $(I-A)^p$ and A^p have the same adjacency matrix, it is the first few powers of A^p that contribute to the largest elements of A^{-1}. Note that the neighborhood of a node x in the graph of A^p is the same as the distance-p neighborhood of x in the graph of A. Therefore, the computation of A^p can be avoided by working directly on the graph of A.

Assume that we have computed a distance-p coloring of the graph of A which results in m colors. Conceptually, if we permuted the nodes with the same color together, the graph of A^p would have m color-blocks along the diagonal that are diagonal matrices. We construct the following structurally orthogonal probing vectors z_j, $j = 1, 2, ...m$,

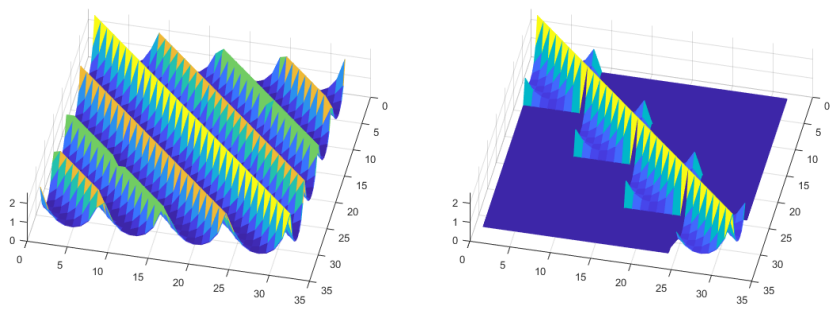
$$z_j(i) = \begin{cases} 1 & \text{if color}(i) = j \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

Notice that these vectors can recover exactly the trace $\mathbf{Tr}(A^p) = \sum_{j=1}^{m} z_j^T A^p z_j$, because they completely annihilate all matrix elements outside the color-blocks along the diagonal of A^p and because the color-blocks are diagonal matrices themselves. Although these diagonal blocks are dense matrices in the A^{-1}, using these z_j in the trace estimator (1.1) has the same effect of annihilating all off-diagonal blocks of A^{-1}, or equivalently, any neighbor at distance up to p from *any* node in the same color group. Then the accuracy of the trace estimation is the summation of the variances (1.2) of the diagonal color-blocks.

Figure 1 is used to display this effect. Let A be a 32-node 1D Laplacian matrix with periodic boundary conditions shifted by its smallest non-zero eigenvalue so it becomes non-singular. A distance-3 coloring of this matrix yields 4 colors. Consider the permutation vector *perm* that lists the indices of all nodes in order of their color label, i.e., nodes with color 1 come first, followed by color 2, 3, and 4. Plotting the A^{-1} symmetrically permuted by *perm* shows the color blocks along the diagonals (Figure 1a). Figure 1b shows $A^{-1} \odot HH^T$ permuted the same way, where the columns of H consist of the four probing vectors from Equation (2.2). It can be seen that every element outside the color-blocks along the main diagonal gets annihilated.

Computationally, a greedy, linear time coloring algorithm can be used, which for most matrices with regular sparsity patterns provides close to optimal number of colors. The bulk of the computation is spent on the iterative method that solves for the m linear systems $A^{-1}z_j$.

CP is a deterministic method. Many applications, such as LQCD, require an unbiased trace estimator (unless the deterministic accuracy can be guaranteed to be well below the statistical significance of the simulation). Moreover, if the probing vectors from the distance-p coloring do not provide sufficient accuracy, we seek ways to either use the $A^{-1} \odot HH^T$ as the matrix of the statistical estimator (1.1) or to extend CP to higher distances. In either case, the work spent on solving $A^{-1}z_j$ should be re-used and not discarded. This has been explored in [23, 20] as described next.



(a) A^{-1} permuted into color-blocks (b) Permuted A^{-1} after probing

Fig. 1: Using a shifted 1D Laplacian A with 32 nodes and boundary conditions, Figure 1a shows A^{-1} permuted into color-blocks based on a distance-3 coloring before probing vectors are applied. Figure 1b shows the result of these color-blocks after the probing vectors are applied to the shifted Laplacian inverse, $A^{-1} \odot HH'$.

2.2. Removing Deterministic Bias. The vectors z_j in (2.2) consist of a sub-vectors of all 1's. To remove the deterministic bias from the CP estimation, we can introduce random noise to the vectors z_j similarly to one step of Hutchinson ($s = 1$). Consider the noise vector $z_0 \in \mathbb{Z}_2^N$ and apply a Hadamard product between z_0 and each of the probing vectors z_j, $j = 1, ..., m$,

$$V = [z_0 \odot z_1, z_0 \odot z_2, ..., z_0 \odot z_m]. \tag{2.3}$$

As shown in [23], $VV^T = HH^T$ have the same non-zero pattern, but using the vectors v_j in (1.1) imparts no deterministic bias.

Moreover, given a sequence of random vectors, $z_0^{(i)}, i = 1, \ldots, s$, we can construct the vector sets $V^{(1)}, \ldots, V^{(s)}$ as above. Using these $s \times m$ vectors in (1.1) is the same as performing s steps of Hutchinson on the variance reduced matrix $A^{-1} \odot HH^T$.

2.3. Hierarchical Probing. Instead of applying the CP method for a fixed distance p followed by the Hutchinson stochastic estimator, it is more beneficial to continue with probing to higher distances as long as the elements of A^{-1} continue to display strong decay and as long as previous work can still be reused.

This is the goal of Hierarchical Probing (HP) which was initially proposed for matrices with lattice-type structure [23] and was later extended to arbitrary sparsity patterns [20]. The idea is to enforce a hierarchical coloring which ensures that probing vectors for smaller distance colorings belong in the subspace of the vectors generated for larger distances. Therefore the trace estimation reuses the already computed quadratures $z_j^T A^{-1} z_j$ and augments them with those from higher distances.

On lattices, we can generate a hierarchical coloring by recursively partitioning a d-dimensional lattice into 2^d sub-lattices, each receiving a different color. The non-overlapping sub-lattices guarantee that if two nodes share a color at distance p, they must also share a color at any smaller distance, and if two nodes do not share a color at distance p, they will not share a color at higher distances. Each recursion step doubles the distance between nodes of the same color. The recursion stops when all nodes are given a separate color or when the requested distance is reached. A red-

black coloring between recursion steps allows for intermediate colorings as the number of colors increases by a factor of 2^d at each recursion.

Instead of using (2.2), probing vectors for the HP can be generated efficiently as special permutations of the rows and columns of the Hadamard or Fourier matrices. The nested coloring implies a nesting of the subspaces of the probing vectors which can be used incrementally until the desired accuracy is achieved. Used in its unbiased form of (2.3) with $s = 1$, this method proved particularly flexible and effective in real world LQCD problems [16, 15].

HP was extended to arbitrary lattice sizes and in particularly general sparse matrices in [20]. These techniques can also be used with the algorithm of this paper if a hierarchical coloring is desired. However, because the number of colors required increase by a factor of 3-4 over the non-hierarchical version, we assume that users can choose a priori the required distance.

2.4. Deflation. A different way to reduce the variance of the estimator is to deflate the lowest singular triplets of A [12]. Given U and V a number of approximate left and right singular vectors of the smallest singular values of A, we can form the oblique projector $Q = AV(U^TAV)^{-1}U^T$ and split the trace computation into two parts,

$$\mathbf{Tr}(A^{-1}) = \mathbf{Tr}(A^{-1}Q) + \mathbf{Tr}(A^{-1}(I - Q)). \tag{2.4}$$

Because $\mathbf{Tr}(A^{-1}Q)$ is easily computed as the trace of the small matrix $(U^TAV)^{-1}$, we can apply the stochastic estimator on the $\mathbf{Tr}(A^{-1}(I - Q))$ which is expected to have smaller variance. The number of singular vectors needed to provide a significant variance reduction of the estimator is dependent on the spectral decay of the matrix A and can be computed using an iterative SVD solver on A [12] or as approximations from the coarse grid space of multigrid [22].

Deflation works complementary to probing. While probing effectively captures heavy elements of A^{-1} occurring within some distance p between nodes, deflation captures heavy connections between elements at long range distances. Therefore combining the two techniques has shown significant improvements over using one of these methods individually.

3. Probing for Permutations. Consider the problem of finding the trace of PA^{-1} where P is a permutation matrix. The problem arises in Lattice QCD where P corresponds to one or more displacements in the lattice. We will study this problem shortly, but let us first consider the problem for a general P.

The question is how to achieve the probing goals for PA^{-1}. The CP method would take powers of the matrix AP^T which does not relate to how information propagates through powers of A to generate A^{-1}. In other words, this method may not capture the largest elements of A^{-1} which are at close graph distances for each node, and thus does not satisfy the design goal of probing. Moreover, the powers $(AP^T)^p$ are much denser than the corresponding A^p which means a larger number of colors and thus probing vectors. Finally, AP^T is a non symmetric matrix so the graph coloring problem is not well defined, although this problem can be avoided by coloring the graph of the symmetric part of a matrix.

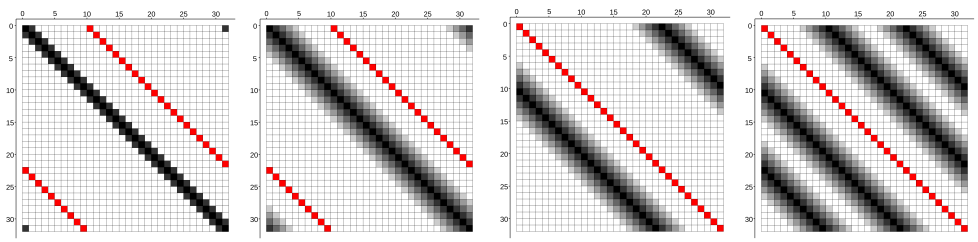
The solution is conceptually simple. Since $PA^{-1} = P\sum_{p=0}^{\infty}(I - A)^p$, we can first take powers of the matrix A, permute them, and then find the coloring on the associated graph of PA^p, or rather its symmetric part $PA^p + (PA^p)^T$. Despite its simplicity, when this method is applied to toroidal lattices stemming from our LQCD application

it creates connectivity patterns that our HP method cannot handle. However, these patterns allow for a CP-based algorithm specifically tailored for this application.

In LQCD, the application of disconnected diagrams requires the trace of a certain projected operator which for the purpose of this discussion can abstracted as the sum of all the elements of A^{-1} that correspond to a displacement $k \in \mathbb{Z}_+^d$, i.e., $\sum_x A_{ij}^{-1}$, where i is the index of the lattice node $x = [x_1, \ldots, x_d]$ and j is the index of node $x+[k_1, \ldots, k_d]$. Let P be the permutation matrix that places the required off-diagonal elements onto the main diagonal. The corresponding permutation index is computed in MATLAB as

```
perm = Coord2Index(mod(Index2Coord([1:N],D)+k,D), D);
```

where the two functions are the maps between lattice coordinates and the particular index ordering of the application. The inverse permutation P^T simply maps a lattice point y to $y - [k_1, \ldots, k_d]$. The idea of coloring the graph of $PA^p + (PA^p)^T$ is shown in Figure 2 for a 1D lattice with $k = 10$ and $p = 4$.



(a) Matrix A, 1D torus (b) Matrix of A^4 (c) Displace by 10 (d) Symmetrized

Fig. 2: Applying a power and displacement to a matrix representation of a 1D toroidal lattice. The red diagonal represents the locations of the elements corresponding to the wanted displacement.

As with CP we use a greedy linear time algorithm to color $PA^p + (PA^p)^T$. However, by working directly on the lattice we are able to speed up the distance-p coloring process. Given a node x with lattice coordinates $[x_1, .., x_d]$, we do not find the distance-p neighborhood of x, but rather the distance-p neighborhoods centered at

$$x^+ = [x_1 + k_1, \ldots, x_d + k_d] \text{ and } x^- = [x_1 - k_1, \ldots, x_d - k_d]. \tag{3.1}$$

Displacements in both k and $-k$ directions enforce a symmetric matrix structure. We denote the distance-p neighborhood of x for displacement k as,

$$\begin{aligned} N^d(x, k, p) &= N^d(x^+, 0, p) \cup N^d(x^-, 0, p) \\ &= \{y : \|y - x^+\|_1 \le p\} \cup \{y : \|y - x^-\|_1 \le p\}. \end{aligned} \tag{3.2}$$

During coloring, we exclude $\{x\}$ from the neighborhood, and when the dimension d is implied, we omit the superscript.

We make three observations. First, the main diagonal of the original A^{-1}, whose elements are typically of the largest magnitude, is part of the off-diagonal structure of PA^{-1} and contributes to the estimator variance. However, the (x, x) elements of this diagonal are now displaced to the (x, x^-) links in the $N^d(x, k, 0)$, so our new method eliminates them immediately for any probing distance. Second, because of the assumed decay, the elements of next-highest magnitude in A^{-1} will be in the

diagonals closest to the main or at distance $p = 1$ from it. The decay continues with higher distances p. Therefore the new algorithm includes in the neighborhoods $N^d(x, k, p)$ all original distance-p neighbors of the points x^+ and x^- as these will have the largest weight. Finally, we note that although $p = 0$ removes the old main diagonal (the graph of $P + P^T$), in practice probing is meaningful for $p \geq 1$.

3.1. Coloring with Displacements Algorithm. Once we have defined the neighborhood of each node in the displacement graph we can use a simple greedy approach to color it [21]. The number of colors translates to the number of iterations in the stochastic estimator. It is not as critical to minimize this number as more vectors/iterations could imply a larger variance reduction. However, this additional reduction beyond the best distance-p coloring is hard to quantify and may not be more effective than using extra random noise vectors. The order in which nodes are visited by the greedy algorithm is thus important.

We have experimented with some common visitation orders such as natural and red-black orderings, a completely random order, and a random red-black where the order of the nodes within a color is random. In addition, we tested a domain decomposition idea, where an independent set of the graph of A^i was constructed for various i's, and then breadth first search was used to add neighborhoods to each of these centers (for $i = 1$ this reverts to red-black). After extensive testing we observed that in most cases, natural and red-black orders achieved the least amount of colors. Surprisingly, thousands of runs of the random variants yielded only marginal improvements, and the domain decomposition idea deteriorated with increasing i. We believe this is due to the well-structured connections of the lattice.

Algorithm 3.1 shows how to work directly on the lattice $\mathbb{Z}_D^d$ to apply the greedy distance-p coloring algorithm for a displacement vector k, and for a user-defined visitation order. It returns a vector `Colors` which can be used in (2.2) to generate the probing vectors. To avoid re-computing the neighborhood for each lattice point, Algorithm 3.2 builds first a "stencil" of coordinate offsets that when added to the coordinates of some point x return the coordinates of the points in $N(x, k, p)$. Because every lattice node is of the same degree, it is clear that the maximum number of colors produced by the greedy algorithm is one more than the degree of a node, i.e., colors are less or equal to $|N(x, k, p)| + 1 = len(Stencil(:, 1)) + 1$. A bit array of this size can be used to record the colors used for each neighborhood and find the first color not in use. The colors returned by Algorithm 3.1 are used in (2.2) and then (2.3) to generate the unbiased probing vectors to be applied on the displaced inverse PA^{-1}.

The size of the distance-p L_1 ball on the lattice is $O(p^d)$ and the stencil contains two such balls in $N(x, k, p)$. To union the two stencil balls we have to remove duplicates when the balls overlap, which can be obtained by sorting the elements. This gives a complexity $O(p^d d \log p)$ to generate the stencil. The dominant part of the complexity is the linear time greedy algorithm which visits the $N(x, k, p)$ for each x, and therefore the algorithm's complexity is $O(Np^d)$.

Although the algorithm we presented is for any d-dimensional displacement, in practical LQCD problems the displacement occurs only in the z space-time direction. For convenience our theoretical discussion considers the displacement to be in the 1st dimension, i.e., $k = k_1$ and $k_2 = \ldots = k_d = 0$.

3.2. Lower Bound on the Number of Colors. The chromatic number of a graph must be at least the size of its maximal clique. In our problem, the neighborhood of every lattice node is the union of two L_1 balls so we seek to identify its maximal clique. This is complicated by the wrap-around property of the torus which adds

Algorithm 3.1 Displacement Coloring on a d-Dimensional Lattice

Input:
k = Displacement array of length d
D = Array of lattice dimension sizes of length d
p = Coloring distance

Output:
$Colors$ = Array of lattice colors

```
N = prod(D); Colors = zeros(N, 1);
Stencil = Create_Stencil(p, k, 1, zeros(1, len(D)));        # Find neighborhood offsets
for i = Make_Visiting_Order(N) do
    ix = Index2Coord(i, D);          # Convert the node index to a lattice coordinate
    # For each offset in the stencil, add it to ix to find ix's neighborhood
    Neighbor_Colors = [];
    for s = Stencil do
        n = Coord2Index(mod(ix + s, D));
        Neighbor_Colors = [Neighbor_Colors, Colors(n)];
    # Create a logical array to mark which colors are already in use
    Colors_In_Use = false(len(Stencil(:, 1)));
    for c = Neighbor_Colors do
        if c > 0 then
            Colors_In_Use(c) = true;
    # Find the first color not in use and set that to be i's color
    for j = 1 :len(Colors_In_Use) do
        if ~ Colors_In_Use(j) then
            Colors(i) = j;
            break;
```

Algorithm 3.2 Find coordinate offsets for each node in a neighborhood

Input:
x = d-dimensional array to store an offset; p = Coloring distance
k = Displacement array of length d; dim = Recursion/dimension level

Output:
$Stencil$ = A mapping of a lattice coordinate's neighbors

```
Create_Stencil(x, p, k, dim)
if dim == 1 then
    Stencil = [];                    # Empty array to hold all neighbor offsets
if dim == len(x)+1 then
    # Append the positively and negatively displaced offset to the stencil
    return unique([Stencil; x + k; x − k], 'rows');
# Find the distance-p neighborhood around x
for j = −p : p do
    x(dim) = j
    Stencil = [Stencil, Create_Stencil(x, p − |j|, k, dim + 1)]
return Stencil
```

additional constraints to the coloring and thus the results depend not only on k and p, but also on the size D_i of each dimension. To avoid this complication, we ignore the toroidal property which, for sufficiently large D_i, is equivalent to considering the lattice $\mathbb{Z}^d_\infty$ which is infinite in all d dimensions. By removing these constraints from the coloring algorithm, the size of the maximal clique of the infinite lattice may be smaller, and thus its size will still be a lower bound to the chromatic number of the finite toroidal lattice. We call the number of colors required to distance-p color the infinite lattice with displacement k in dimension 1, $col(\mathbb{Z}^d_\infty, k, p)$.

Without displacement, $k = 0$, each neighborhood $N(x, 0, p)$ is an L_1 ball of radius p. Any two points in this ball are at L_1 distance $2p$ or less. Therefore, the maximal clique of the distance-p graph of $N(x, 0, p)$ should be the nodes inside the L_1 ball of radius $\lfloor \frac{p}{2} \rfloor$. If p is odd, this L_1 ball is extended by one point in one dimension. The lower bound on the chromatic number is given by the size of this clique

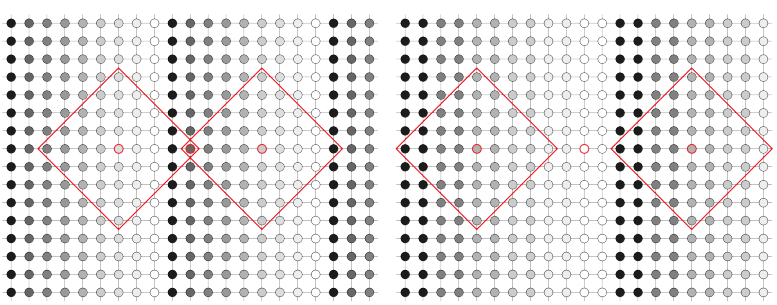
$$col(\mathbb{Z}^d_\infty, 0, p) = \begin{cases} |N^d(\mathbf{0}, 0, \frac{p}{2})| & \text{if } p \text{ is even} \\ |N^d(\mathbf{0}, 0, \lfloor \frac{p}{2} \rfloor)| + |N^{d-1}(\mathbf{0}, 0, \lfloor \frac{p}{2} \rfloor)| & \text{if } p \text{ is odd} \end{cases}, \tag{3.3}$$

where $\mathbf{0} = [0, ..., 0]$ is chosen as a representative neighborhood center. Recurrence relations can be derived to compute this number for any dimension, although general closed forms for an arbitrary number of dimensions are not known. More details can be found in [7, 23, 20].

With displacement ($k > 0$), the L_1 balls of a neighborhood $N(x, k, p)$ are not centered around the node x, resulting in different coloring patterns. We characterize the number of colors needed, first for $k \geq p$ and then for $k < p$. Proofs are given in Appendix A.

THEOREM 3.1. *Let $x \in \mathbb{Z}^d_\infty$. If $k \geq p$, then $\forall y \neq x$ with $y_1 = x_1$, it holds $y \notin N(x, k, p)$.*

The above theorem implies that when $k \geq p$ all nodes with the same x_1-coordinate can share the same color, reducing the d-dimensional coloring problem to a 1D problem. An example of this can be seen in Figure 3. To find the lower bound on the number of colors we consider the two sub-cases, $k = p$ and $k > p$, separately.



(a) $N^2(x, 4, 4)$ (b) $N^2(x, 6, 4)$

Fig. 3: The neighborhood $N^2(\mathbf{0}, k, p)$ and how 1D coloring is sufficient when $k \geq p$.

THEOREM 3.2. *If $k = p$, then $col(\mathbb{Z}^d_\infty, k, p) = 2p + 1$.*

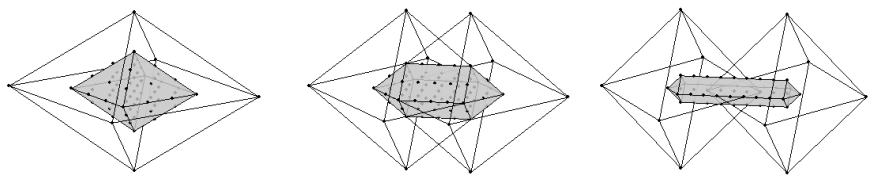
THEOREM 3.3. *If $k > p$, then $col(\mathbb{Z}^d_\infty, k, p) = \lceil \frac{2k}{k-p} \rceil = \lceil \frac{2p}{k-p} \rceil + 2$.*

When $k < p$, the two L_1 balls centered around x^- and x^+ overlap. Next, we identify the maximal clique in this neighborhood for which all points are at distance p or less considering displacement k. As before, we center the neighborhood at $x = \mathbf{0}$.

THEOREM 3.4. *Assume $(p+k)$ is even and $k < p$. Let, $\alpha = \lfloor \frac{p+k}{2} \rfloor$, $\beta = \lfloor \frac{p-k}{2} \rfloor$, and define the set*

$$C(d, \alpha, \beta) = \left\{ x : \|x\|_1 \leq \alpha \text{ and } \sum_{i=2}^{d} |x_i| \leq \beta \right\}. \tag{3.4}$$

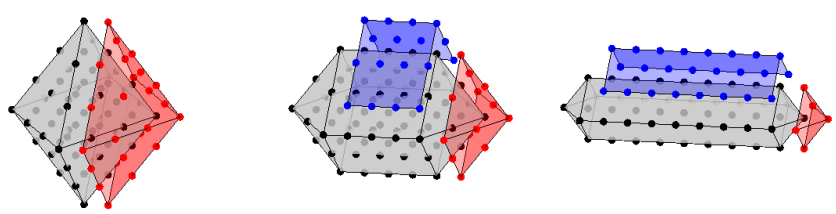
Then $\forall x, y \in C(d, \alpha, \beta)$, $x \in N(y, k, p)$, i.e., $C(d, \alpha, \beta)$ constitutes a distance-p clique.



(a) $p = 6$, $k = 0$ (b) $p = 6$, $k = 2$ (c) $p = 6$, $k = 4$

Fig. 4: The distance-p clique shown in grey of the neighborhood $N^3(\mathbf{0}, k, p)$ which is shown as wire frames, when $p > k$ and $(p + k)$ is even as described in Theorem 3.4.

For $(p + k)$ is odd, (3.3) shows that when $k = 0$ the clique needs to be extended by one hyper-surface. In Theorem 3.5 we prove that for $k > 0$ the clique requires two additional hyper-surfaces as depicted in Figure 5.



(a) $p = 7$, $k = 0$ (b) $p = 7$, $k = 2$ (c) $p = 7$, $k = 4$

Fig. 5: Distance-p cliques of $N^3(\mathbf{0}, k, p)$ when $(p+k)$ is odd as shown in Theorem 3.5. Set $C(d, \alpha, \beta)$ is the grey set in the center, set S is the red hyper-surface on the right, T is the blue hyper-surface on the top.

THEOREM 3.5. *Assume $(p+k)$ is odd and $p > k$. Define $C' = C(d, \alpha, \beta) \cup T \cup S$, where $C(d, \alpha, \beta)$ is defined in (3.4) and*

$$T = \{x : -(k-1) \leq x_1 \leq k \text{ and } 1 \leq x_2 < \beta + 1 \text{ and } \textstyle\sum_{i=2}^{d} |x_i| = \beta + 1\}, \tag{3.5}$$

$$S = \{x : k + 1 \leq x_1 \leq \alpha + 1 \text{ and } |x_2| \leq \beta \text{ and } \|x\|_1 = \alpha + 1\}. \tag{3.6}$$

Then $\forall x, y \in C'$, $x \in N(y,\ k,\ p)$, i.e., C' constitutes a distance-p clique.

Finally, to count the number of points in the clique for any combination of d, k, p we can use the recursive Algorithm 3.3. Table 1 shows the analytic formulas for the size of $C(d, k, p)$ obtained by the nested summations of points over all dimensions for lattices with $d = 1, 2, 3, 4$ when $p + k$ is even and $p > k$. For $p + k$ odd, we need to add also the size of $d - 1$ dimensional hyper-surfaces S and T. It is not hard to see that $|S| + |T| = |C(d - 1, \alpha, \beta)|$. Therefore, we arrive at the following general lower bound for the number of colors of our algorithm,

$$col(\mathbb{Z}^d_\infty, k, p) = \begin{cases} 2p+1 & \text{if } p = k \\ \lceil \frac{2k}{k-p} \rceil & \text{if } p < k \\ |C(d,\alpha,\beta)| & \text{if } p > k,\ p+k \text{ even} \\ |C(d,\alpha,\beta)| + |C(d-1,\alpha,\beta)| & \text{if } p > k,\ p+k \text{ odd} \end{cases} . \tag{3.7}$$

Algorithm 3.3 Recursive Function to Find the Lower Bound on Colors Needed

Input:

p = Coloring distance; k = Displacement (in the first dimension)
s = The current distance traveled; d = Current dimension level
min_Colors = Number of colors needed so far

```
Min_Num_Colors(p, k, s, d, min_Colors)
if k > p then
    min_Colors = ⌈2*p/(k−p)⌉ + 2;
    return min_Colors
if d == 0 then
    min_Colors = min_Colors + 1;
    return min_Colors
if d == 1 then
    min_Colors = min_Colors + 2*(⌊(p−k)/2⌋ − s) + 1;
    return min_Colors
for i = −⌊(p−k)/2⌋ + s : ⌊(p−k)/2⌋ − s do
    min_Colors = Min_Num_Colors(p, k, s + |i|, d − 1, min_Colors);
return min_Colors
```

d	Size of the clique $C(d, \alpha, \beta)$ for $p > k$ and $(p + k)$ even
1	$2\alpha + 1$
2	$-2\beta^2 + 4\alpha\beta + 2\alpha + 1$
3	$-\frac{8}{3}\beta^3 + (4\alpha - 2)\beta^2 + (4\alpha + \frac{2}{3})\beta + 2\alpha + 1$
4	$\frac{1}{3}(2(4\beta^3 + 6\beta^2 + 8\beta + 3)\alpha - 6\beta^4 - 8\beta^3 - 6\beta^2 + 2\beta + 3)$

Table 1: Formulas for size of the clique $|C(d, \alpha, \beta)|$, if $p > k$ and $(p + k)$ is even, with $\alpha = \lfloor \frac{p+k}{2} \rfloor$ and $\beta = \lfloor \frac{p-k}{2} \rfloor$. If $(p + k)$ is odd, use (3.7).

3.3. Clearances. The LQCD application of disconnected diagrams requires the computation of traces not only for one but for multiple displacements (e.g., $k =$

$0, \ldots, 8$). Using different colorings to individually find each of the traces is computationally prohibitive as we would have to solve a different set of linear systems for each of the nine displacements. Therefore, it is natural to ask whether the probing vectors from one displacement can be used effectively for other ones. Theorem 3.6 shows that if a distance-p coloring generated for displacement k is used for displacement $k+\lambda$ or $k-\lambda$, then it clears at least distance $\max(p-\lambda, 0)$.

THEOREM 3.6. *$N(\mathbf{0}, k \pm \lambda, p-\lambda) \subseteq N(\mathbf{0}, k, p)$, for any $\lambda \leq p$.*

Based on this theorem, a specific (k,p)-coloring, i.e., a distance p-coloring for displacement k, will also be effective in reducing variance for nearby displacements. However, its effectiveness declines for farther displacements. In our LQCD experiments we show that choosing larger valued (k,p) pairs is more beneficial.

3.4. Multiple Displacements. The diminishing clearance achieved from (k,p)-coloring to farther displacements motivates the idea of finding a single distance-p coloring for a graph stemming from multiple displacements. The goal is to spread the effectiveness of a power p to more values of k, instead of using one k and a high p value, while still using less colors than all displacements individually. Given a list of displacements, $k_1, k_2, \ldots, k_n$, the neighborhood of a node x can be constructed as,

$$N(x, [k_1, ..., k_n], p) = N(x, k_1, p) \cup \ldots \cup N(x, k_n, p). \tag{3.8}$$

Algorithm 3.1 can be modified to do this by calling *Create_Stencil* for multiple different k vectors and unioning the created stencils together.

As expected from Theorem 3.6, we observed that the resulting clique is smaller when the displacements $k_1, k_2, \ldots, k_n$ are successive. In fact, when the distance between displacements is more than p, this method returns similar number of colors to coloring each displacement separately. However, in our LQCD experiments even successive multiple displacements did not yield improvements in variance over just using one of the higher displacements (say k_n) with distance larger than p. We believe this is due to the fact that smaller displacement traces have significant higher magnitude thus requiring less variance reduction. This is discussed in the experiments section.

3.5. Tiles. Despite the linear complexity of Algorithm 3.1, practical lattice sizes reach 64^4 and often larger, and the neighborhood size is $O(p^4)$ (e.g., for $k=0, p=10$ there are 8361 neighbors to visit). It is clear therefore that we should avoid running the method every time a new trace problem is solved. One solution is to generate and save in a database colorings for most useful lattice sizes. However, the regular structure of the lattice results in coloring patterns that repeat across the lattice. This is one of the motivations for tiling: we color a smaller toroidal lattice, the tile, and repeat its coloring throughout the lattice. Small tiles can be generated at runtime, and several common larger tiles can be saved in the aforementioned database.

The second motivation comes from the effect of lattice size to the number of colors. While our analysis was based on $\mathbb{Z}^d_\infty$, with a wrap-around structure the additional constraints make the number of colors sensitive to the lattice size. For example, the distance-1 coloring of a non-periodic 1D lattice requires 2 colors, while for the toroidal lattice we need 2 colors when D_1 is even and 3 colors when D_1 is odd. These effects are amplified in higher dimensions and larger distances. Interestingly, for a given combination (k,p), increasing the lattice size often results in a larger number of colors. Therefore, it is beneficial if a lattice can be composed with smaller tiles.

There are certain constraints that the tile size must satisfy. First, because the periodicity in the tile must match that of the lattice, a hyper-cubic tile must be used.

Second, the tile needs to be large enough to include an entire $N(x, k, p)$ neighborhood. Otherwise, the neighborhood will wrap-around the boundary and thus require more colors than a larger tile would need. This means that in dimensions without displacement the length needs to be at least $2p + 1$. The dimension with the displacement should have length at least $2(p + k) + 1$. For example, a $(k = 8, p = 8)$-coloring on a 4D lattice would require a tile of size at least 34×18^3.

A third constraint is that the tile dimensions must divide the dimensions of the lattice to ensure a valid coloring. In LQCD lattices have dimensions that are a power of two in size, occasionally including a factor of three. Therefore, the minimum size 34×18^3 tile of the previous example cannot be used. One solution is to consider tiles with each dimension length being the smallest power of two that is greater than the minimum required length. In the previous example, the tile size required for the $(8, 8)$-coloring on a 4D lattice would be 64×32^3. The drawback of this requirement is that tiles may become too large and some of their dimensions (in particular the one with displacement) may be longer than the size of the actual lattice. In such cases, we may limit the tile size in the offending dimension to D_i. This ensures a valid coloring, although with possibly a few more colors, but also standardizes the number of tiles we need to pre-compute and store. In the example above, if the lattice is of size 32×64^3, then the size of the $(8, 8)$-coloring tile becomes 32^4.

p	Displacement								
	0	1	2	3	4	5	6	7	8
1	4^4	8×4^3	8×4^3	16×4^3	16×4^3	16×4^3	16×4^3	32×4^3	32×4^3
2	8^4	8^4	16×8^3	16×8^3	16×8^3	16×8^3	32×8^3	32×8^3	32×8^3
3	8^4	16×8^3	16×8^3	16×8^3	16×8^3	32×8^3	32×8^3	32×8^3	32×8^3
4	16^4	16^4	16^4	16^4	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3
5	16^4	16^4	16^4	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3
6	16^4	16^4	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3
7	16^4	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3	32×16^3
8	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4
9	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4
10	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4	32^4

Table 2: Tile sizes for each (k, p)-coloring for a $32^3 \times 64$ lattice with the displacement in the first dimension (corresponding to the z, x, y, t dimensions of the application).

Table 2 shows the tiles sizes for different (k, p)-colorings chosen with the above policy for a 4-dimensional toroidal lattice of size $32^3 \times 64$. This is the lattice of our experiments in the next section. For clarity the table shows the displacement in the first direction, although our LQCD application requires it in the third dimension.

4. Experiments. We have implemented our code in C and in MATLAB. The computation of all lattice tiles in Table 2 was performed with the C code. All tests were run on the Femto subcluster at William & Mary where each compute node is a 32-core 960 Xeon Skylake with a clock speed of 2.1GHz. The timings for each of the (k, p)-colorings on a single thread are shown in Table 3, but the code can be easily parallelized. While iterating through each node must be sequential in nature to avoid coloring conflicts, gathering the color labels of a single node's neighbors is a read-only process that can be done independently. For example, the maximum number of neighbors each node can have for an $(8, 10)$-coloring is 16,681, allowing for decent speedups. A red-black scheme can also obviously be done in parallel, as the red nodes and black nodes can be separated and colored independently.

p	Displacement								
	0	1	2	3	4	5	6	7	8
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.01	0.00	0.01	0.01	0.01	0.02	0.01	0.01
3	0.01	0.02	0.02	0.02	0.03	0.05	0.04	0.04	0.04
4	0.21	0.33	0.39	0.41	0.82	0.82	0.82	0.83	0.83
5	0.44	0.67	0.79	1.70	1.74	1.75	1.75	1.75	1.76
6	0.84	1.22	2.92	3.16	3.29	3.32	3.32	3.34	3.33
7	1.44	4.13	4.93	5.44	5.68	5.79	5.82	5.84	5.85
8	38.88	55.23	64.81	71.35	77.19	76.69	77.84	78.54	77.99
9	61.63	85.06	97.91	108.21	114.82	119.90	121.16	121.22	122.22
10	91.16	121.33	143.77	157.81	167.94	175.82	179.13	180.09	180.38

Table 3: Time (in seconds) to run each (k, p)-coloring with tile sizes outlined in Table 2 and the resulting number of colors is shown in Table 4.

4.1. Number of Colors Computed. As the number of colors equates to the number of linear systems needing to be solved in Equation (1.1), we are interested in studying how close the number returned by the greedy algorithm is to the theoretical lower bounds summarized in (3.7). As discussed in Subsection 3.5, the lower bounds are for lattices without boundary restrictions so depending on lattice size we expect variability in the deviation from the lower bound.

p	Displacement								
	0	1	2	3	4	5	6	7	8
1	2/2	5/3	4/4	5/3	3/3	4/3	4/4	3/3	3/3
2	16/9	9/6	6/5	10/6	4/4	6/4	5/3	4/3	3/3
3	16/16	32/23	11/10	9/7	8/8	6/5	7/4	5/4	4/4
4	119/41	64/40	92/37	17/14	14/9	12/10	10/6	6/5	4/4
5	170/66	324/91	92/64	64/51	27/18	21/11	19/12	9/7	6/6
6	256/129	442/142	586/141	128/88	104/65	34/22	19/13	18/14	8/8
7	256/192	815/255	795/218	866/192	192/112	172/79	37/26	17/15	16/16
8	1037/321	976/368	1024/381	1206/294	1254/241	336/136	160/93	33/30	30/17
9	1298/450	2031/579	1024/544	1760/507	1577/370	1556/291	288/160	128/107	52/34
10	2220/681	2462/790	3238/837	1922/720	2082/633	1976/446	1954/341	256/184	264/121

Table 4: The first number is the smallest number of colors achieved for distance-p, displacement k, on the tiles of size as noted in Table 2. The second number is the lower bound for that (k, p) from (3.7).

Table 4 shows the least amount of colors achieved between natural and red-black orderings for our different (k, p)-colorings. Next to this number is the theoretical lower bound for each (k, p) combination where $k \in \{0, 1, ..., 8\}$ and $p \in \{1, 2, ..., 10\}$.

The ratio between the two numbers for all combinations is plotted in Figure 6. We observe that when $k \geq p$, the achieved number of colors is very close to the lower bound as the coloring problem becomes one-dimensional, which provides significantly fewer clique constraints. However, once the two displaced neighborhoods begin to overlap, the number of constraints increases and we see the boundary effects of the tiles. Nevertheless, the ratios for the most useful (k, p) combinations are 3 or less.

4.2. Comparisons to Other Methods. Based on the tiles outlined in Table 2, we generated probing vectors that were used in trace estimation experiments using the Chroma library from Jefferson Laboratory [10]. The $32^3 \times 64$ lattice generated by Chroma used a Clover fermion action with quark mass of -0.239. The gauge configuration is from the same ensemble listed as Ensemble B in [12]. More details about this ensemble can be found in [29]. As suggested in [12], we deflate with 200 largest singular vectors of A^{-1} which are computed using the PRIMME library [24]. The solution of linear systems is performed with the MG-proto library of Chroma [1].

[1] http://jeffersonlab.github.io/qphix and github.com/jeffersonlab/mg

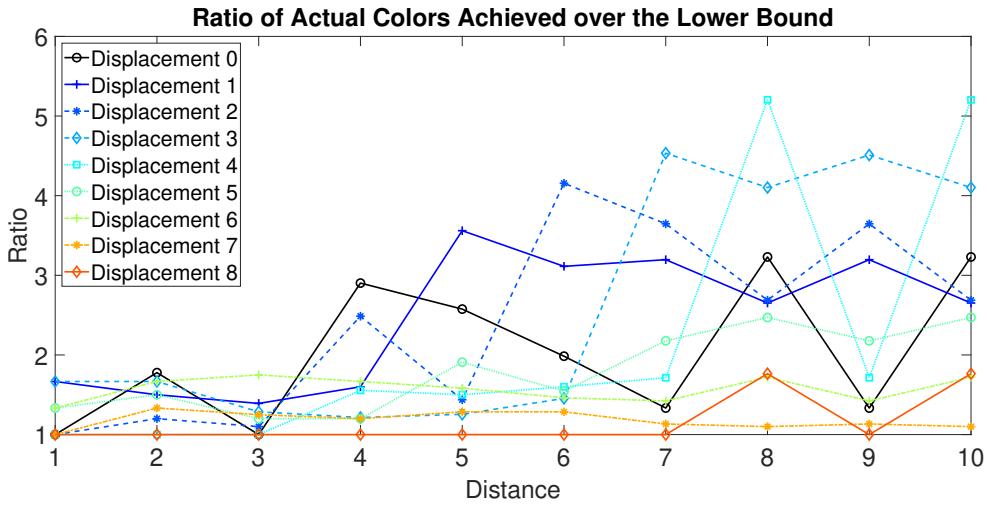


Fig. 6: The ratio of the minimum number of colors achieved with a (k, p)-coloring to the theoretical lower bound in Table 4. Each column is a different displacement.

We compare our displacement probing method against the unprobed Hutchinson method and against CP without displacement. Because in LQCD each lattice point has 12 degrees of freedom (for all spin-color combinations), all methods perform a probing of these 12 components (called spin-color dilution in the literature [6]). This amounts to taking a Kronecker product of each probing or random vector with a 12×12 identity matrix, and thus implies twelve linear systems must be solved for each random or probing vector. We also assume that the matrix A has already been deflated with 200 singular triplets.

Let $v(P_k A^{-1})$ be a shorthand for the variance (1.2) for the matrix $P_k A^{-1}$, where P_k is the permutation matrix that places the elements of A^{-1} corresponding to displacement k in the main diagonal (clearly $P_0 = I$). The unprobed Hutchinson method is run with $s_1 = 1,000$ Rademacher vectors to estimate the trace and variance of $P_k A^{-1}$. For the probing with displacements and the CP methods, let H be the $N \times m$ matrix with the required m probing vectors as columns, and considering $s_2 = 10$ Rademacher vectors, construct the $m \times s_2$ vectors $V^{(1)}, \ldots, V^{(s_2)}$ as in Subsection 2.2. These are used to estimate the trace and variance for each $P_k A^{-1}$.

To compare the methods in a meaningful way we must consider their effect under the same number of linear systems solved. For the Hutchinson method the computed variance of the s_1 quadrature values computed in (1.1) provides a good estimation of $v(P_k A^{-1})$. Similarly for the probing variants after s_2 Hutchinson steps we expect a good estimation of $v((P_k A^{-1}) \odot HH^T)$. However, each of the s_2 stochastic steps of the probing variants solves m linear systems, which implies that the speedup is

$$\text{Speedup over random } = \frac{v(P_k A^{-1})}{m \times v((P_k A^{-1}) \odot HH^T)}. \tag{4.1}$$

Table 6 shows the detailed results for trace and variance estimations as well as speedups for Hutchinson and for our new method for different combinations of k, p. The speedups for probing with displacements over solely random noise are also graphed in Figure 7. We make a few observations. First, the larger the displacement

k, the larger the speedup of the new method over random noise. Second, as mentioned before, distance 1 probing has the biggest impact as it removes the main diagonal of A^{-1}, as well as the elements at distance-1 away from the main diagonal. Third, the speedup increases with p but peaks at a certain distance, typically around $p = 6$ for smaller displacements and around $p = 9$ for larger displacements. This is expected as the elements of A^{-1} decay at higher distances making it less beneficial to probe them directly instead of randomly. Finally, for $k = 0$, probing is equivalent to CP and gives a speedup of 16 over Hutchinson which is slightly better than our previous HP method albeit giving up the hierarchical property.

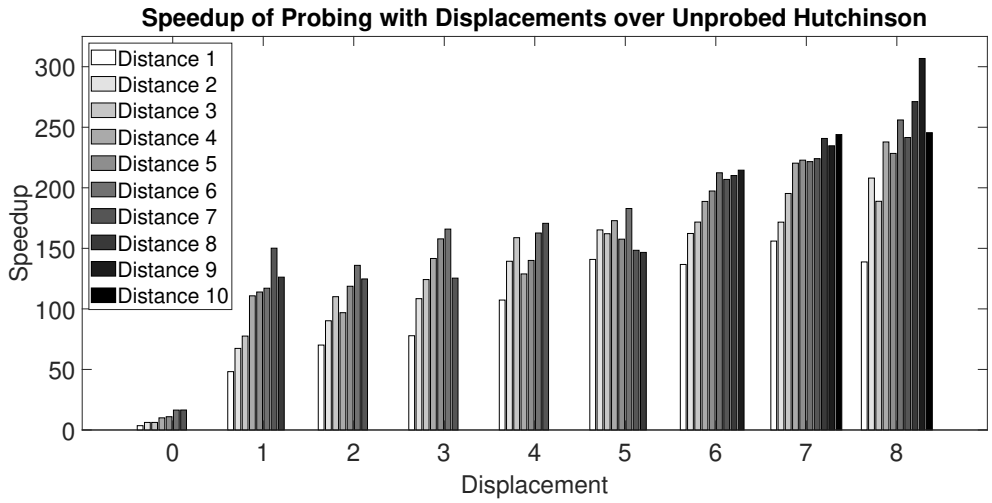


Fig. 7: Speedups of probing with displacements over unprobed Hutchinson using each (k, p)-coloring from Table 6 to find $tr(P_k A^{-1})$.

To solve the problem with displacements, practitioners had previously attempted to use CP or HP [2] or a more localized hopping parameter expansion [30]. We want to show the improvements of our method over CP. Let m_k be the number of probing vectors produced in the (k, p)-coloring to form H_k. Clearly the m_0 vectors forming H_0 are the CP vectors, which could be used to reduce the variance of the estimator for $P_k A^{-1}$. The speedup of probing with displacements over CP is then,

$$Speedup = \frac{v((P_k A^{-1}) \odot H_0 H_0^T) \times m_0}{v((P_k A^{-1}) \odot H_k H_k^T) \times m_k}. \tag{4.2}$$

In Figure 8 we can see this speedup increasing with displacement, although for small displacements it decreases with distance. This is because CP builds its neighborhood outward from the new diagonal, so it can only eliminate the original main diagonal when $p \geq k$. Even then, as the displacement grows the number of colors the new method needs to achieve a distance-p coloring becomes much smaller. For example, a $(0, 7)$-coloring uses 256 colors, while an $(8, 7)$-coloring only uses 16. Therefore, even if CP does remove the high-magnitude elements eventually, it can take many more probing vectors to do so.

4.3. Using one coloring for all displacements. Theorem 3.6 showed that a (k_0, p_0)-coloring would clear all nodes up to distance $p = max(0, p_0 - |k_0 - k|)$ for a

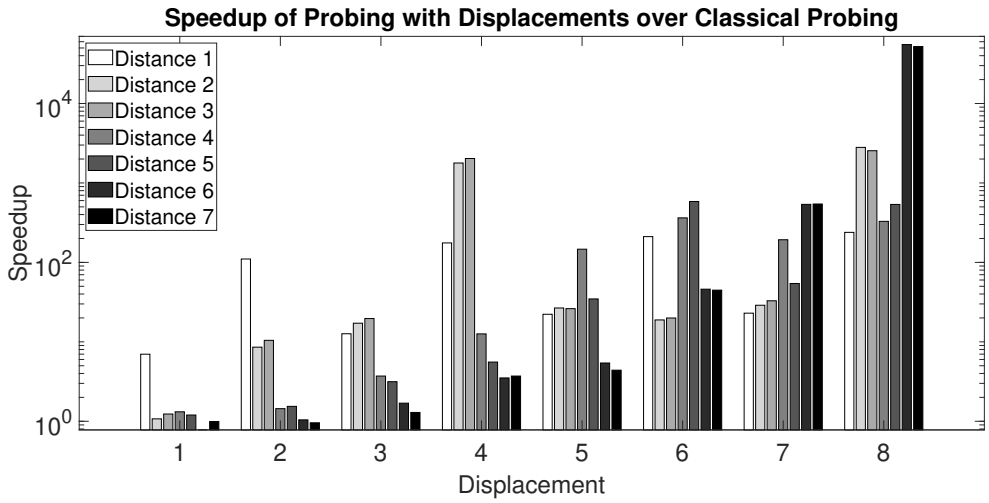


Fig. 8: Speedup of probing with displacements with (k, p)-colorings over classical probing with $(0, k)$-colorings to find $tr(P_k A^{-1})$ using (4.2).

displacement of k. Table 5 confirms this experimentally for the $(8, 10)$-coloring but also shows how many nodes are *not* annihilated beyond the distance described by the theorem. To obtain this, for each pair of (k, p), $k = 0, \ldots, 8, p = 1, \ldots, 12$, we go through every node x in the lattice and compute the percentage of nodes exactly at distance-p from x^+ or x^- that share the same color label as x. These are distance-p neighbors that are not annihilated by the $(8, 10)$-coloring. We report the average of this percentage over all N nodes. When the percentage is 0.00, it means that distance is "cleared", i.e., all nodes of that distance are annihilated from the variance.

Distance	Displacement								
	0	1	2	3	4	5	6	7	8
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	4.55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	9.38	1.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	3.33	3.41	0.63	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	1.40	1.76	0.35	0.00	0.00	0.00	0.00	0.00
7	2.52	0.00	0.78	1.05	0.22	0.00	0.00	0.00	0.00
8	4.69	1.29	0.00	0.49	0.69	0.15	0.00	0.00	0.00
9	2.01	2.56	0.79	0.00	0.33	0.47	0.10	0.00	0.00
10	0.00	1.16	1.64	0.53	0.00	0.24	0.34	0.07	0.00
11	1.66	0.00	0.77	1.14	0.38	0.00	0.18	0.26	0.06
12	3.13	1.05	0.00	0.54	0.83	0.29	0.00	0.13	0.20

Table 5: The average percentage of neighbors at exactly distance-p that do not get eliminated from the trace estimator when using a $(8, 10)$-coloring to find other displacements. The lattice size used is $32^3 \times 64$ with a tile size of 32^4.

The presence of zeros for any $p \leq 10 - |k - 8|$ confirms Theorem 3.6. For each k, we also observe a zero at distances $4i + (p_0 - |k_0 - k|), \forall i \in \mathbb{Z}_+$ which may be attributed to wrap-around effects and/or the red-black ordering that was used for the

$(8, 10)$-coloring. More importantly, however, the percentages of uncleared elements at larger distances is still very small, often less than 1%. This is because a coloring annihilates the distance-p neighbors of all nodes of the same color. For example, if x_1 and x_2 have the same color, some of the neighbors of x_1 may be longer distance neighbors of x_2 but they are annihilated for this p.

Next, we study the effects of this strategy on variance reduction. For each k, we take the (k, p_k)-coloring that gives the best speedup over random noise (from Figure 7) and use it to find the variance $v((P_n A^{-1}) \odot H_k H_k^T)$ for all other displacements $n = 0, \ldots, 8$. Figure 9 shows nine lines, one for each k, plotting its speedup over the Hutchinson method for all n. Each line achieves its maximum speedup at $n = k$ or for smaller k, at $n = k + 1$. It is unclear why this happens for smaller k, e.g., most pronounced for the $(0, 7)$-coloring, but it may have to do with the symmetrization. More importantly, the speedup does not reduce as steeply away from k as Theorem 3.6 would suggest because these colorings work very well for nearby displacements and still work well for more distant ones as described in Table 5.

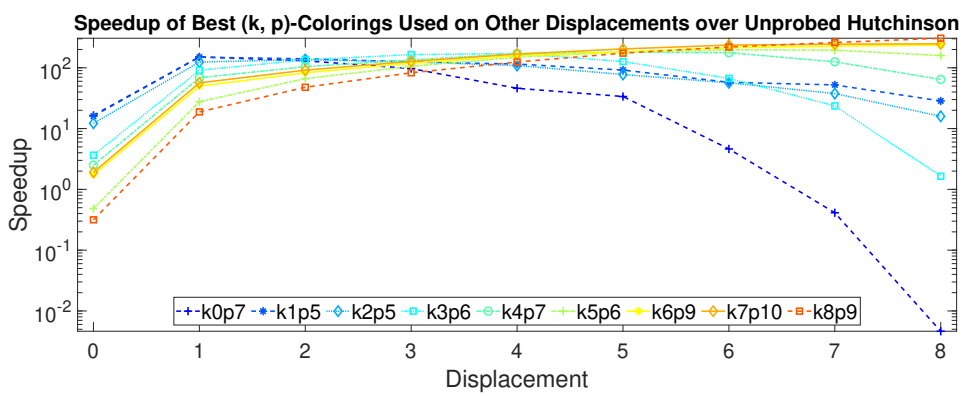


Fig. 9: The speedups over unprobed Hutchinson for each (k, p_k)-coloring to find $tr(P_n A^{-1})$, $\forall n \in \{0, ..., 8\}$

The above results help ascertain the efficiency of the approach, but they cannot help determine which coloring should be used to perform all displacement experiments. There are two reasons. First, the speedups reported depend on the number of probing vectors used. For example, the $(8, 9)$-coloring obtains a speedup of 300 at $k = 8$ but it's because it uses only 52 colors. Its variance is actually four times larger than that of $(7, 10)$-coloring which however uses 250 vectors and thus gets a lower speedup of 250. For a more accurate answer, the $(7, 10)$-coloring would be a better choice.

Second, a smaller variance is only meaningful relative to the value of the trace, and traces for different displacements vary significantly. In Table 6 we see that a variance of 3.275 for the $(0, 4)$-coloring gives 5 digits of accuracy for the trace of $k = 0$, while a variance of 2.332 for the $(8, 9)$-coloring hardly attains a digit for the trace of $k = 8$.

Therefore, to compare colorings over different displacements we introduce the normalized relative error metric which normalizes with respect to both the trace and the number of probing vectors needed. As before, for each k we pick the (k, p_k)-coloring with the best speedup over random. Let m_k be the number of colors it requires, and let M be the maximum number of colors over all colorings being compared (in this

case, $M = 815$). Then, for all $n = 0, \ldots, 9$, the normalized relative error is given by,

$$\frac{\sqrt{v((P_n A^{-1}) \odot H_k H_k^T)\frac{m_k}{M}}}{tr(P_n A^{-1})}. \tag{4.3}$$

The normalization to M ensures all colorings are compared as if they use the same number of probing vectors. The results of this shown in Figure 10.

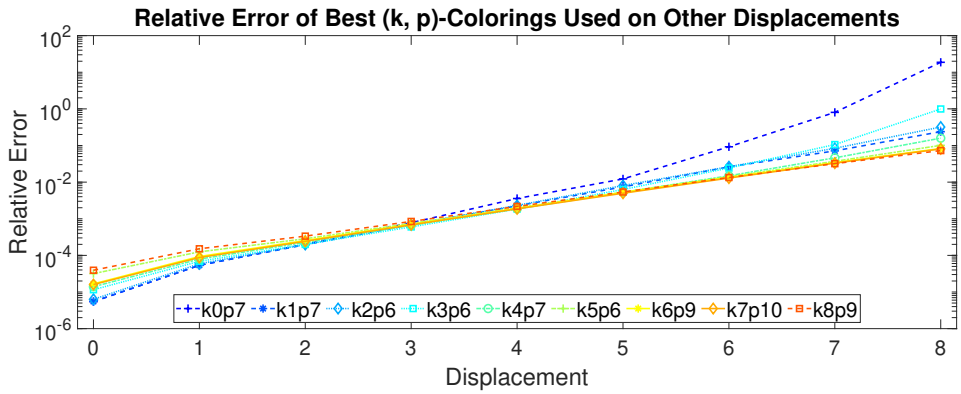


Fig. 10: The relative error (4.3) for each (k, p_k)-coloring used to find $tr(P_n A^{-1})$, $\forall n \in \{0, ..., 8\}$.

The fact that the trace decreases significantly in higher displacements provides a much clearer evaluation picture. For $2 \leq n \leq 6$, all (k, p_k)-colorings have similar normalized relative errors. However, the colorings from larger displacements, e.g., $(7, 10)$ or $(8, 9)$, yield at least 1 to 2.5 digits better accuracy for the same amount of work than colorings from small displacements. Because for displacements less than 4 the errors are already very small, the effort must be focused on the small traces of higher displacements. Therefore, it is best to use the $(7, 10)$-coloring for all displacements, and increase its distance if needed.

5. Conclusion. We have extended the idea of probing for variance reduction of the Hutchinson's trace estimator to the case of permuted matrices and in particular when this permutation corresponds to a lattice displacement k. This has an important application on disconnected diagrams in LQCD. The method works by computing a distance-p coloring not of the original neighborhood of each lattice point x but rather the points within a distance p around centers $x \pm k$.

We have provided a lower bound of the number of colors needed for a particular (k, p)-coloring, and discussed the impact of the lattice size on the number of colors achieved. We have also studied theoretically and experimentally the effect of using a single k, p-coloring for displacements other than k. We have shown that the variance reduction of using probing with displacements is orders of magnitude lower than solely using random noise vectors or than using classical probing that does not take the displacement into consideration. Also, as expected, the trace is smaller as the displacement increases which means that a (k, p)-coloring for larger k needs to be computed and then reused for lower k. This practically gives an additional 10-fold speedup for the LQCD application.

A few open problems could be considered further. The greedy orderings we considered in the greedy coloring approach did not vary substantially in the resulting number of colors, staying within a factor of 3 from the lower bound. It is unclear whether a different ordering can provide considerable reduction in the current number of colors. A second direction is to study the effect of the lattice or tile size to the coloring. Understanding this theoretically rather than experimentally, and providing also a lower bound on the number of colors based on a finite lattice size could be useful in understanding the limitations of the current approach. Finally, it is worth extending the theory and algorithms to the case where the decay of the elements in the matrix inverse depends on the L2 distance, which is closer to what LQCD theory predicts for long range distances.

REFERENCES

[1] K. Ahuja, B. K. Clark, E. D. Sturler, D. M. Ceperley, and J. Kim, *Improved scaling for quantum monte carlo on insulators*, SIAM Journal on Scientific Computing, 33 (2011), p. 1837–1859, https://doi.org/10.1137/100805467.

[2] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, H. Panagopoulos, and G. Spanoudes, *Complete flavor decomposition of the spin and momentum fraction of the proton using lattice QCD simulations at physical pion mass*, Phys. Rev. D, 101 (2020), p. 094513, https://doi.org/10.1103/PhysRevD.101.094513, https://arxiv.org/abs/2003.08486.

[3] C. Alexandrou, M. Constantinou, K. Hadjiyiannakou, K. Jansen, and F. Manigrasso, *Flavor decomposition for the proton helicity parton distribution functions*, Phys. Rev. Lett., 126 (2021), p. 102003, https://doi.org/10.1103/PhysRevLett.126.102003, https://arxiv.org/abs/2009.13061.

[4] P. R. Amestoy, I. S. Duff, J.-Y. L'excellent, Y. Robert, F.-H. Rouet, and B. Uçar, *On computing inverse entries of a sparse matrix in an out-of-core environment*, SIAM Journal on Scientific Computing, 34 (2012), https://doi.org/10.1137/100799411.

[5] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, Journal of the ACM, 58 (2011), p. 1–34, https://doi.org/10.1145/1944345.1944349.

[6] G. S. Bali, S. Collins, and A. Schaefer, *Effective noise reduction techniques for disconnected loops in Lattice QCD*, Computer Physics Communications, 181 (2010), pp. 1570–1583.

[7] M. Beck and S. Robins, *Computing the Continuous Discretely: Integer-Point Enumeration in Polyhedra*, Springer, 2007.

[8] C. Bekas, E. Kokiopoulou, and Y. Saad, *An estimator for the diagonal of a matrix*, Applied Numerical Mathematics, 57 (2007), p. 1214–1229, https://doi.org/10.1016/j.apnum.2007.01.003.

[9] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Inc., USA, 1986.

[10] R. G. Edwards and B. Joó, *The chroma software system for lattice qcd*, Nuclear Physics B - Proceedings Supplements, 140 (2005), p. 832–834, https://doi.org/10.1016/j.nuclphysbps.2004.11.254.

[11] J. Foley., K. J. Juge, A. O'Cais, M. Peardon, S. Ryan, and J.-I. Skullerud, *Practical all-to-all propagators for lattice qcd*, Comput. Phys. Commun., 172 (2005), pp. 145–162, https://arxiv.org/abs/hep-lat/0505023.

[12] A. S. Gambhir, A. Stathopoulos, and K. Orginos, *Deflation as a method of variance reduction for estimating the trace of a matrix inverse*, SIAM Journal on Scientific Computing, 39 (2017), https://doi.org/10.1137/16m1066361.

[13] A. S. Gambhir, A. Stathopoulos, K. Orginos, B. Yoon, R. Gupta, and S. Syritsyn, *Algorithms for disconnected diagrams in lattice qcd*, Proceedings of 34th annual International Symposium on Lattice Field Theory — PoS(LATTICE2016), (2016), https://doi.org/10.22323/1.256.0265.

[14] A. H. Gebremedhin, F. Manne, and A. Pothen, *What color is your Jacobian? Graph coloring for computing derivatives*, SIAM Rev., 47 (2005), pp. 629–705.

[15] J. Green, N. Hasan, S. Meinel, M. Engelhardt, S. Krieg, J. Laeuchli, J. Negele, K. Orginos, A. Pochinsky, and S. Syritsyn, *Up, down, and strange nucleon axial form*

factors from lattice qcd, Phys. Rev. D, 95 (2017), p. 114502, https://doi.org/10.1103/PhysRevD.95.114502, https://link.aps.org/doi/10.1103/PhysRevD.95.114502.

[16] J. Green, S. Meinel, M. Engelhardt, S. Krieg, J. Laeuchli, J. Negele, K. Orginos, A. Pochinsky, and S. Syritsyn, *High-precision calculation of the strange nucleon electromagnetic form factors*, Phys. Rev. D, 92 (2015), p. 031501, https://doi.org/10.1103/PhysRevD.92.031501, https://link.aps.org/doi/10.1103/PhysRevD.92.031501.

[17] H. Guo, *Computing traces of functions of matrices*, Dec 1999.

[18] R. Gupta and B. Carlsten, *Lattice qcd at los alamos*, https://www.lanl.gov/science-innovation/science-programs/office-of-science-programs/high-energy-physics/theoretical-physics/lattice.php.

[19] M. Hutchinson, *A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines*, Communication in Statistics- Simulation and Computation, 19 (1990), pp. 432–450, https://doi.org/10.1080/03610918908812806.

[20] J. Laeuchli and A. Stathopoulos, *Extending hierarchical probing*, 2018, http://www.cs.wm.edu/~andreas/publications/extended_HP.pdf.

[21] J. Mitchem, *On Various Algorithms for Estimating the Chromatic Number of a Graph*, The Computer Journal, 19 (1976), pp. 182–183, https://doi.org/10.1093/comjnl/19.2.182, https://doi.org/10.1093/comjnl/19.2.182, https://arxiv.org/abs/https://academic.oup.com/comjnl/article-pdf/19/2/182/1139205/19-2-182.pdf.

[22] E. Romero, A. Stathopoulos, and K. Orginos, *Multigrid deflation for lattice qcd*, Journal of Computational Physics, 409 (2020), p. 109356, https://doi.org/10.1016/j.jcp.2020.109356.

[23] A. Stathopoulos, J. Laeuchli, and K. Orginos, *Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices*, SIAM Journal on Scientific Computing, 35 (2013), pp. S299–S322, https://doi.org/10.1137/120881452, https://arxiv.org/abs/1302.4018.

[24] A. Stathopoulos and J. R. McCombs, *Primme: Preconditioned iterative multimethod eigensolver—methods and software description*, ACM Trans. Math. Softw., 37 (2010), https://doi.org/10.1145/1731022.1731031, https://doi.org/10.1145/1731022.1731031.

[25] G. W. Stewart, *Matrix Algorithms: Volume 1, Basic Decompositions*, Society for Industrial Mathematics, 1998, https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Matrix%20Algorithms%20(Vol.%201_%20Basic%20Decompositions)%20%5BStewart%201998-12%5D.pdf.

[26] J. M. Tang and Y. Saad, *A probing method for computing the diagonal of a matrix inverse*, Numerical Linear Algebra with Applications, 19 (2011), p. 485–501, https://doi.org/10.1002/nla.779.

[27] W. Wilcox, *Noise methods for flavor singlet quantities*, in Interdisciplinary Workshop on Numerical Challenges in Lattice QCD, 8 1999, https://arxiv.org/abs/hep-lat/9911013.

[28] M. N. Wong, F. J. Hickernell, and K. I. Liu, *Computing the trace of a function of a sparse matrix via Hadamard-like sampling*, Dept. of Mathematics, Hong Kong Baptist University, 2004.

[29] B. Yoon, R. Gupta, T. Bhattacharya, M. Engelhardt, J. Green, B. Joó, H.-W. Lin, J. Negele, K. Orginos, A. Pochinsky, and et al., *Controlling excited-state contamination in nucleon matrix elements*, Physical Review D, 93 (2016), https://doi.org/10.1103/physrevd.93.114506, http://dx.doi.org/10.1103/PhysRevD.93.114506.

[30] R. Zhang, H.-W. Lin, and B. Yoon, *Probing nucleon strange and charm distributions with lattice QCD*, (2020), https://arxiv.org/abs/2005.01124.

Appendix A.

Proof for Theorem 3.1: From the definition of x^+, x^-, the assumption $k \geq p$ implies $|x_1 - x_1^+| = |x_1 - x_1^-| \geq p$, and $\sum_{i=2}^{d} |x_i - x_i^+| = \sum_{i=2}^{d} |x_i - x_i^-| = 0$. Let $y \in Z_\infty^d$ with $y \neq x$ and $y_1 = x_1$. Since $|y_1 - x_1^+| \geq p$, $\|y - x^+\|_1 = |x_1 - x_1^+| + \sum_{i=2}^{d} |x_i - x_i^+| > p$. The same argument applies for the distance from x^-. Therefore, $y \notin N(x, k, p)$. □

Proof for Theorem 3.2: From Theorem 3.1, if $k = p$, the coloring problem reduces to coloring in one dimension. Then the neighborhood definition covers $4p + 1$ indices in the first dimension, from $-2p$ to $+2p$. The maximal clique size of the $2p$-distance graph of these nodes is the 1D unit ball of half the size which includes indices from $-p$ to $+p$. The size of this clique is exactly $2p + 1$ nodes which is also the number of colors needed. □

Proof for Theorem 3.3: As previously noted, if $k \geq p$, the coloring problem gets reduced to one dimension. Consider two nodes on $\mathbb{Z}^1_\infty$ that are exactly $2k$ links apart, i.e., node i and $i+2k$. Because $k > p$, node $i+2k$ does not belong in $N(i,k,p)$ and thus the two nodes can take the same color. Therefore, the problem reduces to coloring a tile of $2k$ consecutive nodes that can then repeat to color the entire $\mathbb{Z}^1_\infty$ lattice.

Consider $2k$ consecutive nodes, 0 to $2k-1$. The first $k-p$ can share color 1 as they do not belong in each other's neighborhood. The following $k-p+1, \ldots, 2(k-p)$ nodes have at least one of the nodes in the first group as neighbor so they must take a different color, say 2. Similarly, every $k-p$ group of nodes must take a different color. The last node has neighbors $i \geq 2k-k-p = k-p$, so it cannot reuse any color including color 1 because the tile needs to repeat. Then the total number of colors is the partitioning of $2k$ nodes in $k-p$ groups. Note that $\lceil \frac{2k}{k-p} \rceil = \lceil \frac{2(p+k-p)}{k-p} \rceil = \lceil \frac{2p}{(k-p)} \rceil + 2$. □

Proof for Theorem 3.4: First note that $\alpha - \beta = k$. Let $x, y \in C(d, \alpha, \beta)$. Then the following hold:

$$|x_1| + \sum_{i=2}^{d} |x_i| \leq \alpha, \qquad |y_1| + \sum_{i=2}^{d} |y_i| \leq \alpha \tag{5.1}$$

$$\sum_{i=2}^{d} |x_i| \leq \beta, \qquad \sum_{i=2}^{d} |y_i| \leq \beta. \tag{5.2}$$

WLOG assume $x_1 \leq y_1$. Then it is sufficient to show that x belongs in the left neighborhood around y^-, i.e., $x \in N(y^-, 0, p)$ or $|y_1^- - x_1| + \sum_{i=2}^{n} |y_i - x_i| \leq p$.

We distinguish two cases for the distance between x_1 and y_1.

(a) $y_1 - x_1 \geq k > 0$. Then using (5.1), we have $|y_1^- - x_1| + \sum_{i=2}^{n} |y_i - x_i| = y_1^- - x_1 + \sum_{i=2}^{n} |y_i - x_i| \leq |y_1| + |x_1| - k + \sum_{i=2}^{n} |y_i| + \sum_{i=2}^{n} |x_i| \leq p + k - k = p$.

(b) $0 \leq y_1 - x_1 < k$. Then using (5.2), we have: $|y_1^- - x_1| + \sum_{i=2}^{n} |y_i - x_i| = -y_1 + x_1 + k + \sum_{i=2}^{n} |y_i - x_i| \leq k + \sum_{i=2}^{n} |y_i| + \sum_{i=2}^{n} |x_i| \leq k + p - k = p$.

□

Proof for Theorem 3.5: For brevity we denote $C = C(d, \alpha, \beta)$. Let $x, y \in C'$, i.e., they belong in one of the sets C, T, S. Because of symmetry, we consider the following pairs of conditions for (x, y): (C, C), (T, T), (S, S), (C, T), (C, S), (T, S).

Notice that the set C is the clique obtained by $p' = p - 1$ and k. Then, case (C, C) is covered by Theorem 3.4 which bounds the (displaced) distance of any two points in C by $p' = p - 1 < p$. This observation can be used to show similarly the cases (C, T) and (C, S). Specifically for (C, T), $x \in C$ and any $y \in T$ will be exactly at distance 1 from some point in C, which means $\|x - y\| \leq p' + 1 = p$. For (C, S), a $y \in S$ is also at distance 1 from any point in C by extending the first dimension.

As in Theorem 3.4, we assume $x_1 \leq y_1$ and show that x belongs in the left neighborhood around y^-, i.e., $x \in N(y^-, 0, p)$ or $\delta = |y_1^- - x_1| + \sum_{i=2}^{d} |y_i - x_i| \leq p$. We also use the following property of absolute values,

$$|f - g| - |f| - |g| = \begin{cases} 0, & \text{if } fg \leq 0, \\ -2\min(|f|, |g|), & \text{otherwise.} \end{cases} \tag{5.3}$$

• *Case* (T, T)*:*

Using the last two conditions of (3.5), the corresponding part of (5.3) for x_2, y_2, and

$2\beta = p - k - 1$ we have,

$$\begin{aligned}\sum_{i=2}^{d} |y_i - x_i| &\le |y_2 - x_2| + \sum_{i=3}^{d} |y_i| + \sum_{i=3}^{d} |x_i| \\ &\le |y_2 - x_2| + (\beta + 1 - |y_2|) + (\beta + 1 - |x_2|) \\ &= 2\beta + 2 - 2\min(|x_2|, |y_2|) \le p - k - 1.\end{aligned}$$

Then $\delta = |y_1^- - x_1| + \sum_{i=2}^{d} |y_i - x_i| \le |y_1 - k - x_1| + p - k - 1$. Using the first condition in (3.5) we have,
If $y_1 - k \ge x_1$ then $\delta \le y_1 - k - x_1 + p - k - 1 \le (k) - k + (k-1) + (p-k-1) = p - 2 < p$.
If $y_1 - k < x_1$ or $y_1 - x_1 < k$ then $\delta \le x_1 - y_1 + k + (p - k - 1) \le k + p - k - 1 < p$.
• *Case* (S, S)*:*
Again we prove $x \in N(y^-, 0, p)$. Based on the conditions in (3.6), $\sum_{i=3}^{d} |x_i| = \alpha + 1 - x_1 - |x_2|$, and $\sum_{i=3}^{d} |y_i| = \alpha + 1 - y_1 - |y_2|$, and since $2\alpha = p + k - 1$ we have,

$$\begin{aligned}\delta &\le |y_1^- - x_1| + |y_2 - x_2| + \sum_{i=3}^{d} |y_i| + \sum_{i=3}^{d} |x_i| \\ &= |y_1^- - x_1| + |y_2 - x_2| + 2\alpha + 2 - |x_2| - |y_2| - x_1 - y_1 \\ &= p + k + 1 + (|y_1 - k - x_1| - x_1 - y_1) + (|y_2 - x_2| - |x_2| - |y_2|) \\ &\le p + k + 1 + (k + |y_1 - x_1| - x_1 - y_1) - 2\min(|x_2|, |y_2|) \\ &\le p + 2k + 1 - 2\min(|x_1|, |y_1|) - 2\min(|x_2|, |y_2|) \\ &= p + 2k + 1 - 2(k+1) = p - 1 < p.\end{aligned}$$

• *Case* (T, S)*:*
Let $x \in T$, $y \in S$. From the defining conditions, $x_1 \le k < y_1$. We work similarly with the previous cases, replacing the $\sum_{i=3}^{d}$, and noting that $\alpha + \beta = p - 1$,

$$\begin{aligned}\delta &\le |y_1^- - x_1| + |y_2 - x_2| + (\alpha + 1 - |y_1| - |y_2|) + (\beta + 1 - |x_2|) \\ &= p + 1 + (|y_1 - k - x_1| - |y_1|) + (|y_2 - x_2| - |y_2| - |x_2|) \\ &\le p + 1 + (|y_1 - k - x_1| - |y_1|).\end{aligned}$$

If $y_1 - k \ge x_1$, then $|y_1 - k - x_1| - |y_1| = y_1 - k - x_1 - y_1 = -k - x_1 \le -k + (k-1) = -1$. Thus $\delta \le p$.
If $y_1 - k \le x_1$, then $|y_1 - k - x_1| - |y_1| = x_1 + k - y_1 - y_1 \le 2k - 2(k+1) = -2$. Thus, $\delta \le p - 1 < p$. □

Proof for Theorem 3.6: We consider only the $k + \lambda$ case as the $k - \lambda$ has a similar proof. Because of symmetry, we also consider only the positive displacements x^+ and y^+ from (3.1). It is sufficient to show that if $x \in N(\mathbf{0}, k + \lambda, p - \lambda)$, then $x \in N(y^+, 0, p)$. From (3.2) we have $\sum_{i=2}^{n} |x_i| + |x_1 - (k + \lambda)| \le p - \lambda$. We distinguish the following cases.

(a) If $x_1 - k \ge \lambda$, then also $x_1 \ge k$, and thus $\sum_{i=2}^{n} |x_i| + x_1 - k - \lambda \le p - \lambda \Rightarrow \sum_{i=2}^{n} |x_i| + |x_1 - k| \le p \Rightarrow x \in N(\mathbf{0}, k, p)$.
(b) If $x_1 < k + \lambda$, then $\sum_{i=2}^{n} |x_i| - x_1 + k \le p - 2\lambda$. We distinguish two sub-cases.
  (b.1) If $x_1 \le k$, then $\sum_{i=2}^{n} |x_i| + |x_1 - k| \le p - 2\lambda \le p \Rightarrow x \in N(\mathbf{0}, k, p)$.
  (b.2) If $x_1 > k$ and since $x_1 - k < \lambda$, then $\sum_{i=2}^{n} |x_i| + k - x_1 \le p - 2\lambda \Rightarrow \sum_{i=2}^{n} |x_i| + x_1 - k \le p - 2\lambda + 2(x_1 - k) < p - 2\lambda + 2\lambda = p \Rightarrow x \in N(\mathbf{0}, k, p)$.

□

Appendix B

			1,000 RNVs w/o Probing	10 RNVs w/ Probing		
k	p	Approx. Trace	Variance	Colors	Variance	Speedup
0	1	6,339,643.7	249,827.7	2	35,075.3	3.56
	2			16	2,502.5	6.24
	3			16	2,501.2	6.24
	4			119	209.6	10.02
	5			170	134.2	10.95
	6			256	59.4	16.43
	7			256	59.1	16.50
1	1	652,636.1	2,341,455.9	5	9,721.2	48.17
	2			9	3,861.4	67.38
	3			32	943.5	77.55
	4			64	330.2	110.78
	5			324	63.4	113.94
	6			442	45.2	117.08
	7			815	19.1	150.13
	8			976	19.0	126.21
2	1	185,764.9	2,360,726.0	4	8,415.7	70.13
	2			6	4,362.0	90.20
	3			11	1,949.6	110.08
	4			92	264.9	96.87
	5			96	207.2	118.70
	6			586	29.6	135.95
	7			795	23.8	124.73
3	1	56,047.8	2,364,612.0	5	6,076.0	77.84
	2			10	2,180.4	108.45
	3			9	2,115.0	124.22
	4			17	982.1	141.63
	5			64	234.1	157.82
	6			128	111.4	165.89
	7			866	21.8	125.41
4	1	17,893.6	2,388,516.5	3	7,420.1	107.30
	2			4	4,285.6	139.33
	3			8	1,880.1	158.81
	4			14	1,323.9	128.87
	5			27	631.8	140.02
	6			104	141.2	162.67
	7			192	72.9	170.66
5	1	6,059.5	2,358,840.1	4	4,186.9	140.85
	2			6	2,379.5	165.22
	3			6	2,425.6	162.08
	4			12	1,137.2	172.86
	5			21	712.8	157.58
	6			34	379.3	182.90
	7			172	92.4	148.40
	8			332	48.4	146.65
6	1	2,183.3	2,392,640.1	4	4,375.9	136.70
	2			5	2,948.7	162.29
	3			7	1,990.8	171.69
	4			10	1,267.6	188.75
	5			19	638.1	197.36
	6			19	592.9	212.40
	7			37	312.5	206.92
	8			160	71.2	210.12
	9			288	38.7	214.56
7	1	836.9	2,378,138.9	3	5,081.5	156.00
	2			4	3,463.9	171.64
	3			5	2,435.2	195.31
	4			6	1,798.8	220.35
	5			9	1,185.9	222.82
	6			18	596.2	221.62
	7			17	624.3	224.07
	8			33	299.3	240.76
	9			128	79.2	234.68
	10			256	38.1	243.95
8	1	339.3	2,381,007.2	3	5,719.0	138.78
	2			3	3,814.9	208.04
	3			4	3,151.8	188.86
	4			4	2,502.7	237.85
	5			6	1,737.5	228.40
	6			8	1,162.5	256.02
	7			16	616.1	241.53
	8			30	292.7	271.18
	9			52	149.2	306.80
	10			264	36.7	245.51

Table 6: The estimation of traces and variances for 1,000 RNVs run without probing for different values of k and p compared to probing with displacements and 10 RNVs.