Algorithms for tensor network renormalization

G. Evenbly*

Department of Physics and Astronomy, University of California, Irvine, California 92697-4575, USA

(Received 30 November 2015; revised manuscript received 16 December 2016; published 13 January 2017)

We discuss in detail algorithms for implementing tensor network renormalization (TNR) for the study of classical statistical and quantum many-body systems. First, we recall established techniques for how the partition function of a $2D$ classical many-body system or the Euclidean path integral of a $1D$ quantum system can be represented as a network of tensors, before describing how TNR can be implemented to efficiently contract the network via a sequence of coarse-graining transformations. The efficacy of the TNR approach is then benchmarked for the $2D$ classical statistical and $1D$ quantum Ising models; in particular the ability of TNR to maintain a high level of accuracy over sustained coarse-graining transformations, even at a critical point, is demonstrated.

DOI: 10.1103/PhysRevB.95.045117

I. INTRODUCTION

Tensor network renormalization [1] (TNR) is a recently introduced approach for coarse-graining tensor networks, with application to the efficient simulation of classical statistical many-body systems and quantum many-body systems. A key feature of TNR that differentiates it from previous methods for coarse-graining tensor networks, including Levin and Nave's tensor renormalization group [2] (TRG) as well as other subsequently developed approaches [3–10], is the use of unitary *disentanglers* in TNR that function to allow removal of all short-ranged correlation at each length scale. This proper removal of short-ranged correlation allows TNR to resolve significant computational and conceptual problems encountered by previous methods.

Despite the success and usefulness of TRG, it is known to suffer a computational breakdown when at or near a critical point [2], where the cost of maintaining an accurate effective description of the system grows quickly with coarse-graining step, due to the accumulation of short-ranged correlation. The use of disentanglers allows TNR to prevent this accumulation, such that TNR can maintain a description over repeated coarse-graining steps, or equivalently for very large system sizes, without requiring a growth of computational cost. Previous methods for coarse-graining tensor networks, such as TRG, are also conceptually problematic if they are to be interpreted as generating a renormalization group [11] (RG) flow in the space of tensors, in that they do not reproduce the expected structure of RG fixed points. This flaw was partially resolved with the proposal of tensor entanglement filtering renormalization [6] (TEFR), which reproduces the proper structure of gapped RG fixed points. On the other hand, TNR fully resolves this problem, reproducing the proper structure of gapped fixed points as well as producing scale-invariant fixed points when applied to critical systems corresponding to discrete versions of conformal field theories [12,13] (CFTs), thus correctly realizing Wilson's RG ideas [14] on tensor networks. By capturing scale invariance, and producing a rescaling transformation for the lattice consistent with conformal transformations of the field theory [15], TNR can produce an accurate description of the fixed-point RG map, from which the critical data characterizing the CFT can then be extracted.

The use of disentanglers in TNR, and their success in preventing the retention and accumulation of short-ranged degrees of freedom, is closely related to the use and success of disentanglers in entanglement renormalization [16] (ER) and in the multiscale entanglement renormalization ansatz [17] (MERA). This connection was formalized in Ref. [18], which showed that TNR, when applied to the Euclidean path integral of a quantum Hamiltonian H, can generate a MERA for ground, excited, and thermal states of H. Thus TNR also provides an alternative to previous algorithms [19] based on variational energy minimization for obtaining optimized MERAs, and also allows methods developed for extracting scale-invariant data from quantum critical systems using MERAs [20–24] to be generalized to classical statistical systems.

In this paper we introduce the numeric algorithms required to implement TNR for the study of $2D$ classical or $1D$ quantum many-body systems. Due to the use of disentanglers, which are key to the TNR approach, implementation of TNR requires more sophisticated optimization strategies than have been necessary in previous tensor RG approaches. This paper is organized as follows. First we discuss the standard techniques through which the partition function of a classical system or the Euclidean path integral of a quantum system can be expressed as a tensor network. Then we discuss the general principle of *local approximations* on which tensor RG schemes are based, before detailing the particular projective truncations involved in the TNR approach. Optimization algorithms for the implementation of TNR are then presented, and their performance benchmarked in the $2D$ classical and $1D$ quantum Ising models.

II. TENSOR NETWORK REPRESENTATIONS OF MANY-BODY SYSTEMS

In this section we discuss approaches for which the partition function of a $2D$ classical statistical system or the Euclidean path integral of a $1D$ quantum system can each be expressed as a square lattice network, which is the starting point for

*gevenbly@uci.edu

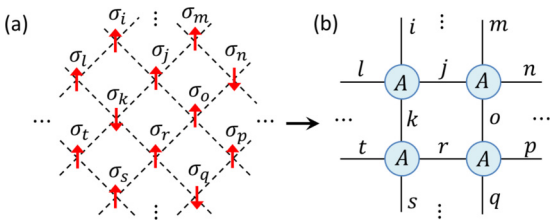

FIG. 1. (a) A square lattice of classical spins $\sigma \in \{+1,-1\}$. (b) The partition function of the classical system can be encoded as a square network (tilted 45° with respect to the spin lattice) of four-index tensors A_{ijkl}, with a tensor sitting in the center of every second plaquette of spins. Here each tensor A encodes the Boltzmann weights associated with the interactions of spins on the edges of the plaquette; see Eq. (3).

the TNR approach (and other tensor renormalization methods, such as TRG, in general).

A. Classical many-body systems

Here we describe an approach for expressing the partition function Z at temperature T of a $2D$ classical statistical system,

$$Z = \sum_{\{\sigma\}} e^{-H(\{\sigma\})/T}, \tag{1}$$

as a network of tensors. As a concrete example let us examine the classical Ising model on the square lattice, with Hamiltonian functional,

$$H(\{\sigma\}) = -\sum_{\langle i,j\rangle} \sigma_i \sigma_j, \tag{2}$$

where $\sigma_i \in \{+1,-1\}$ is an Ising spin on site i. We construct a representation of the partition function as a square-lattice tensor network composed of copies of a four-index tensor A_{ijkl}, where a tensor sits in the center of every second plaquette of Ising spins according to a checkerboard tiling, such that the square lattice of A tensors is tilted 45° with respect to the lattice of Ising spins; see also Fig. 1. Notice that this corresponds to having one tensor A for every two spins. We define the tensor A to encode the four Boltzmann weights $e^{\sigma_i\sigma_j/T}$ of the Ising spin interactions on the edges of the plaquette on which it sits,

$$A_{ijkl} = e^{(\sigma_i\sigma_j+\sigma_j\sigma_k+\sigma_k\sigma_l+\sigma_l\sigma_i)/T}, \tag{3}$$

such that the partition function is then given by the sum over all indices,

$$Z = \sum_{ijk\cdots} A_{ijkl}A_{mnoj}A_{krst}A_{opqr}\cdots. \tag{4}$$

This construction for expressing the partition function as a tensor network can be employed for any model with nearest-neighbor interactions on the square lattice, and can also be generalized to other lattice geometries and to models with longer range interactions.

B. Quantum many-body systems

Here we describe how, given a local Hamiltonian H for a $1D$ quantum system, an arbitrarily precise tensor network representation of the Euclidean time evolution operator $e^{-\beta H}$ can be obtained using a Suzuki-Trotter decomposition [25]. We assume, for simplicity, that Hamiltonian H is a sum of identical nearest-neighbor terms h,

$$H = \sum_r h_{r,r+1}. \tag{5}$$

We begin by expanding the time evolution operator as a product of evolutions over some small time step τ,

$$e^{-\beta H} = (e^{-\tau H})^{(\beta/\tau)}. \tag{6}$$

The evolution $e^{-\tau H}$ over small time step τ may then be approximated,

$$e^{-\tau H} \approx e^{-\tau H_{\text{odd}}} e^{-\tau H_{\text{even}}}, \tag{7}$$

where H_{odd} and H_{even} represent the contribution to H given from sites r odd or r even respectively, and an error of order $O(\tau)$ has been introduced. (Note that one can obtain an error $O(\tau^n)$, $n > 1$, by using a higher-order Suzuki-Trotter decomposition [26]). Since H_{odd} is a sum of terms that act on different sites and therefore commute, $e^{-\tau H_{\text{odd}}}$ is simply a product of two-site gates, and similarly for $e^{-\tau H_{\text{even}}}$,

$$e^{-\tau H_{\text{odd}}} = \prod_{\text{odd}r} e^{-\tau h_{r,r+1}},$$
$$e^{-\tau H_{\text{even}}} = \prod_{\text{even}r} e^{-\tau h_{r,r+1}}. \tag{8}$$

Thus, if one regards each two-site gate $e^{-\tau h}$ as a four-index tensor and Eqs. (8) and (7) are substituted into Eq. (6), a representation of the Euclidean path integral $e^{-\beta H}$ as a square-lattice tensor network is obtained; see also Fig. 2(a). Note that this representation of $e^{-\beta H}$ has incurred an error of order $O(\beta\tau)$, which can be diminished through use of a smaller time step τ.

While this network could potentially serve as the starting point for the TNR approach (or other algorithm for the renormalization of a tensor network) it is desirable to perform some preliminary manipulations before employing TNR. This initial manipulation involves (i) a transformation that maps to a new square-lattice network tilted 45° with respect to the initial network, followed by (ii) coarse-graining in the Euclidean time direction. Given that the initial tensor network is highly anisotropic for small time step τ, as the operator $e^{-\tau h}$ is very close to the identity, step (ii) is useful to obtain a tensor network representation of $e^{-\beta H}$ that is closer to being isotropic (and thus more suitable as a starting point for TNR).

Step (i) is accomplished by performing a modified step of the TRG algorithm as follows. The singular value decomposition (SVD) is taken across a vertical partition of the gate $e^{-\tau h}$,

$$e^{-\tau h} = (u\sqrt{s})(\sqrt{s}v^\dagger), \tag{9}$$

where the root of the singular weights s has been absorbed into each of the unitary matrices u and v, and likewise the eigen-decomposition is taken across a horizontal partition of the gate $e^{-\tau h}$,

$$e^{-\tau h} = (w\sqrt{d})(\sqrt{d}w^\dagger); \tag{10}$$

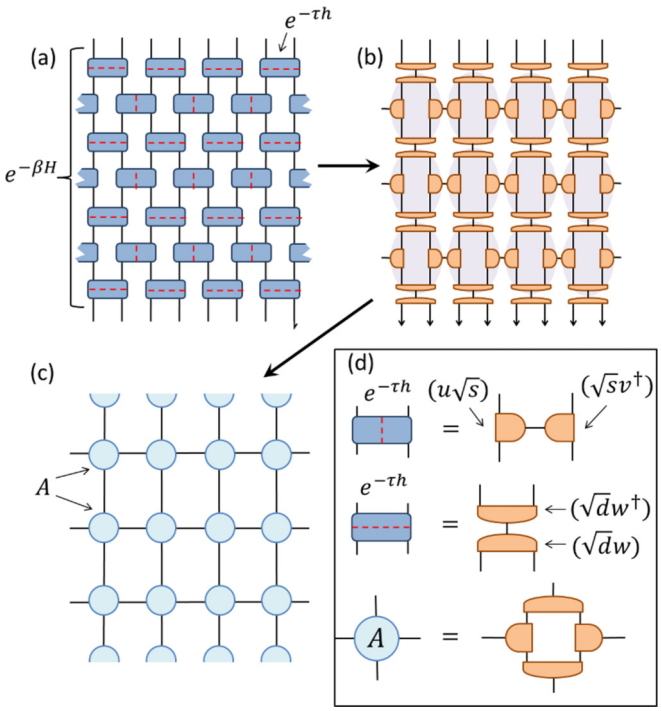

FIG. 2. (a) The imaginary time-evolution operator $e^{-\beta H}$ is expressed via Suzuki-Trotter expansion as a product of two-site gates $e^{-\tau h}$. Red dashed lines denote how the gates are to be decomposed at the next step. (b) The two-site gates $e^{-\tau h}$ are decomposed into a product of ternary tensors according to either a horizontal partition (accomplished via singular value decomposition) or a vertical partition (accomplished via eigen-decomposition). (c) Groups of four ternary tensors are contracted together to form four-index tensors A. (d) Depictions of the vertical and horizontal partitions of the two-site gates $e^{-\tau h}$, and definition of the four index tensors A.

see Fig. 2(d). Here w is a unitary matrix, which follows from $e^{-\tau h}$ being Hermitian, and d are the eigenvalues (which can be argued to be strictly positive for sufficiently small time step τ). The SVD and eigen-decompositions are performed throughout the network according to the pattern indicated in Fig. 2(a), and a new square network of tensors A, tilted 45° with respect to the original, is formed by contracting groups of the resulting tensors together as indicated in Figs. 2(b) and 2(c).

In step (ii) the network of tensors A is then coarse-grained in the Euclidean time direction using standard techniques, i.e., by combining pairs of rows together and then truncating the resulting squared bond index similarly to the higher-order tensor renormalization group (HOTRG) [8] method (see Appendix A for details), until the network is sufficiently isotropic in terms of its correlations. One way to examine how close the network is to being isotropic is to compute the spectra of the transfer matrices formed by tracing out the horizontal or vertical indices of a single A tensor, whose decay should match as closely as possible.

III. COARSE-GRAINING TENSOR NETWORKS

Consider a tensor network $\mathcal{G}$ consisting of copies of four-index tensors A_{ijkl} that we assume are arranged in an $L \times L$ square-lattice network with periodic boundary conditions. Our goal is to contract this network, or perhaps this network with single or multiple impurity tensors, to evaluate the scalar, denoted $\langle \mathcal{G} \rangle$, associated with the network. As an exact contraction of the network $\mathcal{G}$ is exponentially expensive in system size L one must rely on approximations in order to evaluate a large network. In this section we first describe the generic concept of *local approximations* that could be employed to approximate such a contraction, then discuss the class of local approximation used in TRG, namely the truncated singular value decomposition (SVD), before introducing the particular class of local approximation that the TNR algorithm is based on, which we call *projective truncations*.

A. Local approximations

Let $\mathcal{F}$ denote a subnetwork of tensors, for example a 2×2 block of tensors A, from the full network $\mathcal{G}$. The key idea underlying coarse-graining methods for tensor networks is that of the *local approximation*: that one can safely replace a subnetwork $\mathcal{F}$ with a different network of tensors $\tilde{\mathcal{F}}$ if they differ by a small amount ε,

$$\varepsilon \equiv \|\mathcal{F} - \tilde{\mathcal{F}}\|, \tag{11}$$

where we assume for convenience that $\mathcal{F}$ has been normalized such that $\|\mathcal{F}\| = 1$. If this condition is fulfilled, then the scalar $\langle \mathcal{G} \rangle$ associated with the contraction of network $\mathcal{G}$ will only differ by a small amount $O(\varepsilon)$ under replacement of subnetwork $\mathcal{F}$ by new subnetwork $\tilde{\mathcal{F}}$. Note that we use the Hilbert-Schmidt norm,

$$\|A\| = \sqrt{\mathrm{tTr}(A \otimes A^\dagger)}, \tag{12}$$

where "tTr" denotes the tensor trace, or equivalently the contraction of all indices, between two tensors of equal dimensions (or two networks with matching "open" indices); see also Fig. 3(a). In general, renormalization schemes for tensor networks, such as TRG or the focus of this paper, TNR, employ a pattern of local approximations over all positions on the tensor network, in conjunction with contractions, in order to generate coarser networks of tensors. For example, as illustrated in Fig. 3, local replacements of all 2×2 blocks of tensors $\mathcal{F}$ with a new network $\tilde{\mathcal{F}}$, consisting of a four-index core tensor surrounded by three-index tensors, can result in a coarser, $(L/2) \times (L/2)$ network of new four-index tensors. Assuming that sufficient accuracy could be maintained over repeated steps, this procedure could be iterated $O(\log_2 L)$ times, resulting in a network of $O(1)$ linear dimension which could then be exactly contracted.

B. Truncated singular value decomposition

In principle, any form of local approximation capable of yielding a small error ε in Eq. (11) could be viable as part of a coarse-graining scheme. In the original TRG algorithm proposed by Levin and Nave [2], and in many of the generalizations and improvements to TRG [5,7,8], the local approximations that are used are based on a truncated singular value decomposition (SVD) of single tensors [or a generalized form of the SVD known as the higher-order singular value decomposition [27] (HOSVD)], which we now discuss.

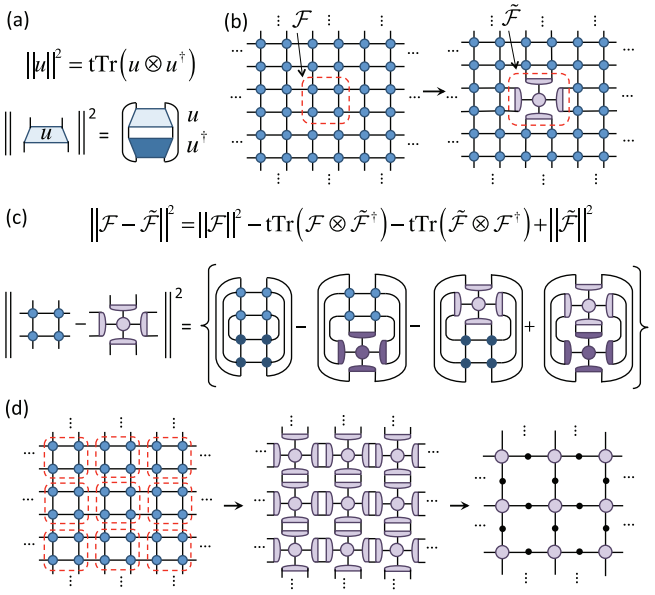

FIG. 3. (a) Depiction of (the square of) the Hilbert-Schmidt norm of a four-index tensor u. Note that a darker shade is used to represent the conjugate tensor, which is also drawn with opposite vertical orientation. (b) Given a square lattice tensor network, we wish to replace a 2×2 block of tensors $\mathcal{F}$ from the network with a different subnetwork of tensors $\tilde{\mathcal{F}}$. (c) The square of the difference between $\mathcal{F}$ and $\tilde{\mathcal{F}}$ under the Hilbert-Schmidt norm is depicted, where darker shades are used to depict conjugate tensors, which are drawn with opposite vertical orientation to regular tensors. The replacement in (b) is valid if the difference $\|\mathcal{F} - \tilde{\mathcal{F}}\|$ is sufficiently small. (d) Assuming that the local square-lattice network is homogeneous, one can replace $\mathcal{F}$ with $\tilde{\mathcal{F}}$ in all 2×2 blocks. A coarser square-lattice network is obtained after contraction between pairs of three-index tensors.

Consider a four-index tensor A_{ijkl} where each index is of dimension χ. If the tensor is viewed as a $\chi^2 \times \chi^2$ matrix according to the pairing of indices $A_{[ij][kl]}$ then the SVD can be performed,

$$A_{ijkl} = \sum_{m}^{\chi^2} u_{ijm} s_{mm} v_{mkl}, \tag{13}$$

where u and v are unitary according to grouping of indices $u_{[ij][m]}$ and $v_{[m][kl]}$, respectively, and s is a positive diagonal matrix of singular values λ, i.e., $s_{nm} = \delta_{nm}\lambda_m$, that we assume are ordered such that $\lambda_m < \lambda_{m+1}$. If we truncate the SVD to retain only the $\chi' < \chi^2$ largest singular values then the decomposition becomes approximate,

$$A_{ijkl} \approx \sum_{m}^{\chi'} u_{ijm} s_{mm} v_{mkl}, \tag{14}$$

where the truncation error ε, as defined in Eq. (11), is seen to equal the square root of the sum of the squares of the discarded singular values,

$$\varepsilon = \sqrt{\sum_{m=\chi'+1}^{\chi^2} (\lambda_m)^2}. \tag{15}$$

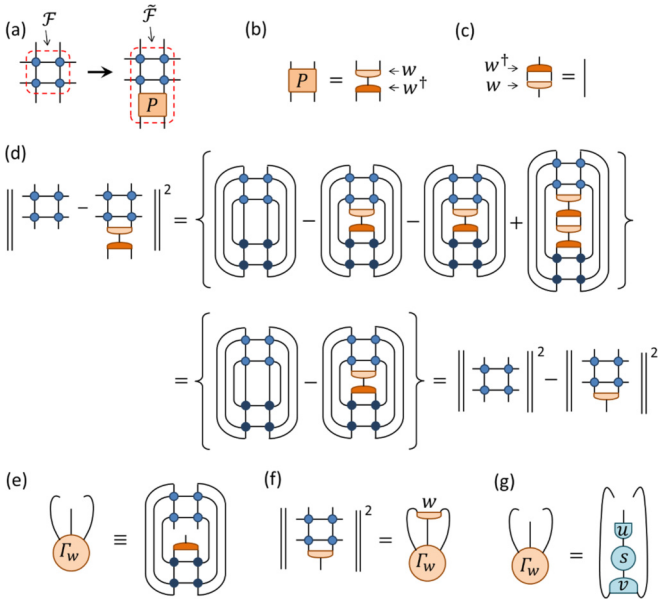

FIG. 4. (a) In a *projective truncation* a subnetwork $\mathcal{F}$ is replaced by a new subnetwork $\tilde{\mathcal{F}}$, which consists of a projector P applied to the original subnetwork, i.e., $\tilde{\mathcal{F}} = \mathcal{F}P$. (b) Here we assume that P is decomposed as a product of an isometric tensor w and its conjugate, $P = ww^\dagger$. (c) By definition, isometry w contracts to identity with its conjugate, $w^\dagger w = \mathbb{I}$. (d) The square of the error in a projective truncation is expanded as a sum of four terms; however given that $P^2 = P$, two of the terms cancel; see also Eq. (18). (e) The environment Γ_w of isometry w is defined as the network that results by removing a single instance of w from $\|\mathcal{F}w\|^2$; see also Eq. (20). (f) By construction, the contraction of w and its environment Γ_w is equal to $\|\mathcal{F}w\|^2$. (g) Environment Γ_w is decomposed, via singular value decomposition (SVD), into a product of isometric tensors u, v, and diagonal matrix s.

Here we have assumed that the tensor A was normalized, $\|A\| = 1$, or equivalently that the singular values were normalized as $\sqrt{\sum_m (\lambda_m)^2} = 1$. The SVD is known to provide the optimally accurate decomposition of a tensor A into the product of a pair of tensors (connected by an index of some rank χ'); thus it has proved vitally useful as the foundation for many previous schemes for the renormalization of tensor networks.

C. Projective truncations

The TNR approach requires use of a broader class of local approximation than those based on the SVD, which we term *projective truncations*. In a projective truncation, a local subnetwork $\mathcal{F}$ is replaced by a new local network $\tilde{\mathcal{F}}$ that consists of a projector P, which satisfies $PP^\dagger = P^2 = P$, acting on some or all of the open indices of $\mathcal{F}$,

$$\tilde{\mathcal{F}} = \mathcal{F}P = \mathcal{F}ww^\dagger; \tag{16}$$

see Fig. 4(a) for an example. Here we have expanded the projector P as the product of an isometric tensor w and its conjugate, $P = ww^\dagger$, where the isometry satisfies $w^\dagger w = \mathbb{I}$; see also Figs. 4(b) and 4(c). [Note that, as part of the TNR algorithm, we shall also consider cases where projector is decomposed as a more complicated product of many different

isometries; see for example Fig. 6(c)]. Projective truncations are a particularly useful class of local approximation, as the figure of merit to optimize projector P takes a very simple form. To see this, we first expand the terms in Eq. (11),

$$\varepsilon^2 = \|\mathcal{F}\| + \|\tilde{\mathcal{F}}\| - \mathrm{tTr}(\mathcal{F} \otimes \tilde{\mathcal{F}}^*) - \mathrm{tTr}(\tilde{\mathcal{F}} \otimes \mathcal{F}^*). \quad (17)$$

Notice that, as $PP^\dagger = P$, for a projective truncation we have

$$\|\tilde{\mathcal{F}}\| = \mathrm{tTr}(\mathcal{F} \otimes \tilde{\mathcal{F}}^*) = \mathrm{tTr}(\tilde{\mathcal{F}} \otimes \mathcal{F}^*). \quad (18)$$

It follows that the expression for the replacement error ε can be simplified,

$$\begin{aligned} \varepsilon &= \sqrt{\|\mathcal{F}\|^2 - \|\mathcal{F}P\|^2} \\ &= \sqrt{\|\mathcal{F}\|^2 - \|\mathcal{F}w\|^2}, \end{aligned} \quad (19)$$

where we have again made use of $PP^\dagger = P$ in reaching the second line of working; see also Fig. 4(d).

Let us now turn to the problem of optimizing the projector P such that the error ε of Eq. (19) is minimized. For simplicity we consider the case where the projector P decomposes as a product of a single isometry w and its conjugate, $P = ww^\dagger$, although a key feature of the method we discuss is that it can be applied to the more complicated case, an in Fig. 6(c), where P is represented as a product of several different isometric tensors. Notice that the error ε of Eq. (19) is minimized when $\|\mathcal{F}w\|$ is maximized, which follows as $\|\mathcal{F}w\| \leqslant \|\mathcal{F}\|$; we now discuss an iterative strategy for optimizing isometry w to maximize the expression $\|\mathcal{F}w\|$.

The strategy we employ is based on *linearization* of the cost function. Given that the expression $\|\mathcal{F}w\|^2$ has quadratic dependence on isometry w (or, more specifically, it depends on both w and $w^\dagger$), we simplify the problem by temporarily holding $w^\dagger$ fixed and then solving the resulting linear optimization for w, and iterating these steps until w is sufficiently converged. Note that this follows the same strategy employed to optimize tensors in a MERA as described in Ref. [19], to which we refer the interested reader for more details. We begin by expressing the closed network $\|\mathcal{F}w\|^2$ in a factorized form,

$$\|\mathcal{F}w\|^2 = \mathrm{tTr}(\Gamma_w \otimes w), \quad (20)$$

where Γ_w, referred to as the environment of w, represents the contraction of everything in $\|\mathcal{F}w\|^2$ excluding tensor w; see also Figs. 4(e) and 4(f). The singular value decomposition of the environment Γ_w, when considered as a matrix according to the same partition of indices for which tensor w is isomeric, is then taken,

$$\Gamma_w = usv^\dagger, \quad (21)$$

as shown in Fig. 4(g). The isometry w is then updated to become

$$w = vu^\dagger, \quad (22)$$

where it can be argued that this choice of updated isometry maximizes Eq. (20). However, as the cost function was linearized, such that $w^\dagger$ in the environment Γ_w was held fixed, these steps of (i) computing the environment Γ_w and (ii) obtaining an updated w through the SVD of the environment Γ_w must be iterated many times, until the solution converges.

IV. TENSOR NETWORK RENORMALIZATION

Tensor network renormalization is a class of coarse-graining scheme designed to be compatible with proper removal of all short-ranged correlations at each RG step [1]. For any given lattice geometry there are many potential TNR schemes the fulfill this requirement. In this paper we focus on a particular implementation of TNR for a $2D$ square lattice network that we call the *binary* TNR scheme, as introduced in Ref. [1], which reduces the linear dimension of the network by a factor of 2 with each RG step. In Appendix B we discuss a *ternary* TNR scheme, which reduces the linear dimension of the network by a factor of 3 with each RG step, and in Appendix C we discuss an *isotropic* binary TNR scheme that treats both dimensions of the tensor network equally while also reducing the linear dimension of the network by a factor of 2 at each RG step. Similarly TNR can also be implemented in other lattice geometries besides the square lattice, including those in higher dimensions. The majority of the algorithmic details we present for implementation of the binary TNR scheme carry over to other TNR schemes.

A. Coarse-graining step of the binary TNR scheme

The starting point for an iteration of the binary TNR scheme is a square lattice tensor network $\mathcal{G}$ composed of four-index tensors A_{ijkl}, where indices are assumed to be of some dimension χ. As discussed in Sec. II such a network could represent the partition function of a $2D$ classical statistical system or the Euclidean path integral of a $1D$ quantum system. For simplicity we assume that network $\mathcal{G}$ is spatially homogeneous, i.e., that all its tensors are copies of a unique tensor A_{ijkl}, while noting that the algorithm can easily be extended to deal with nonhomogeneous networks (special examples, including networks with an open boundary or a defect line, can be handled using similar methods to those developed in the context of MERAs [28–32], and are discussed separately in Ref. [33]). We also assume that $\mathcal{G}$ is invariant under complex conjugation plus reflection on the horizontal axis, as discussed further in Appendix D. Again, this assumption is not strictly necessary, but is useful in simplifying the TNR algorithm. For the case in which $\mathcal{G}$ represents a Euclidean path integral of a quantum Hamiltonian H the presence of this symmetry follows from H being Hermitian, while for the case in which $\mathcal{G}$ represents the partition function of a classical system the symmetry is present if the underlying $2D$ classical statistical model has an axis with which it is invariant under spatial reflection. We now describe the coarse-graining steps involved in an iteration of the binary TNR scheme, which maps network $\mathcal{G}$ to the coarser network $\mathcal{G}'$ whose linear dimension has been reduced by a factor of 2, before discussing the optimization of the tensors involved and other algorithmic components in more detail.

The first step of the iteration is to apply a particular gauge change on the horizontal indices on every second row of tensors in $\mathcal{G}$, as discussed in Appendix D. Here the gauge change is chosen such that it is equivalent to flipping top-bottom

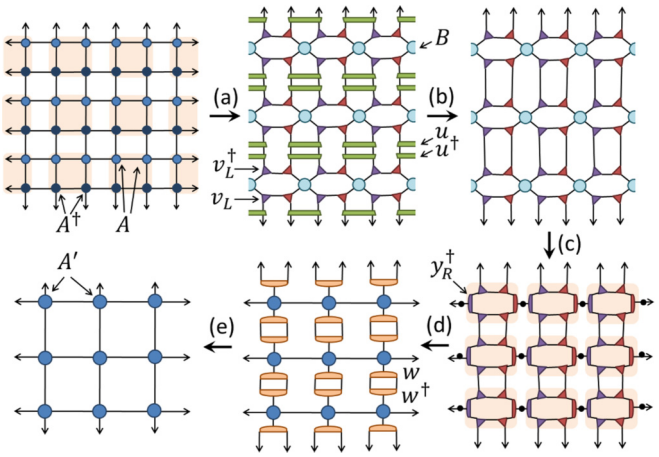

FIG. 5. The sequence of coarse-graining steps used in the binary TNR scheme in order to map an initial square lattice of tensors A, where every second row of tensors has been conjugated as described in Appendix D, to a coarser square lattice composed of tensors A'. (a) A projective truncation is made on all 2×2 blocks of tensors; see Figs. 6(a)–6(c). (b) Conjugate pairs of disentanglers u are contracted to identity. (c) A projective truncation is made on all B tensors; see Figs. 6(d)–6(e). (d) A final projective truncation is made; see Figs. 6(f) and 6(g) for details. (e) Conjugate pairs of isometries w are contracted to identity.

indices of tensors A and taking the complex conjugation; as such we denote the transformed tensors $A^\dagger$. That such a gauge transformation exists follows from the assumed reflection symmetry. Next, Fig. 5 depicts the remaining steps in transforming network $\mathcal{G}$ into the coarser network $\mathcal{G}'$. In Fig. 5(a), a projective truncation is enacted on 2×2 blocks of tensors A (where two of the tensors have undergone the aforementioned change of gauge as $A^\dagger$), the details of which are shown in Fig. 6(a). The projector P_u used at this step is represented as a product of two isometries v_L and v_R and a unitary tensor u (and their conjugates) as shown in Fig. 6(c). The unitary tensors u, which we call *disentanglers*, act on two neighboring indices such that, if we regard each index of the network as hosting a χ-dimensional complex vector space $\mathbb{V}_\chi$, they describe a mapping between vector spaces,

$$u : \mathbb{V}_\chi \otimes \mathbb{V}_\chi \to \mathbb{V}_\chi \otimes \mathbb{V}_\chi. \tag{23}$$

By virtue of being unitary, the disentanglers satisfy $u^\dagger u = \mathbb{I}^{\otimes 2}$ where $\mathbb{I}$ is the identity operator on $\mathbb{V}_\chi$. Conceptually, the role of disentanglers is to remove short-range correlations that would otherwise be missed, as discussed in greater detail in Ref. [1], and they constitute the key difference between TNR and previous tensor renormalization schemes. Isometries v_L and v_R each map two indices in the network, one horizontal and one vertical, to a new index of some chosen dimension $\chi' \leqslant \chi^2$,

$$v_L : \mathbb{V}_{\chi'} \to \mathbb{V}_\chi \otimes \mathbb{V}_\chi, \quad v_R : \mathbb{V}_{\chi'} \to \mathbb{V}_\chi \otimes \mathbb{V}_\chi, \tag{24}$$

where the new index has been regarded as hosting a χ'-dimensional complex vector space $\mathbb{V}_{\chi'}$. By definition, isometries satisfy $v_L^\dagger v_L = v_R^\dagger v_R = \mathbb{I}'$, with $\mathbb{I}'$ the identity operator on $\mathbb{V}_{\chi'}$. After the coarse-graining step of Fig. 5(a), it is useful to define a new four-index tensor B, which is defined from

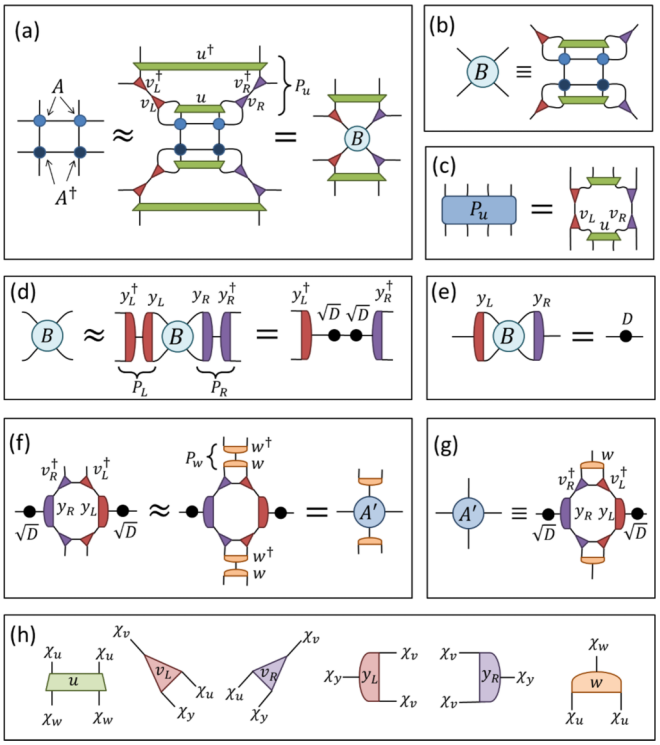

FIG. 6. (a) Details of the projective truncation made at the first step of the TNR iteration; here two copies of a projector P_u, which is composed of a product isometric and unitary tensors, are applied to a 2×2 block of A tensors. (b) Definition of four-index tensor B. (c) Projector P_u is formed from isometries v_L, v_R and disentangler u (and their conjugates). (d) Details of the projective truncation made at the second step of the TNR iteration. (e) Definition of matrix D. (f) Details of the projective truncation made at the third step of the TNR iteration. (g) Definition of new four-index tensor A', copies of which comprise the coarse-grained square-lattice tensor network. (h) Delineation of the different dimensions $\{\chi_u, \chi_v, \chi_w, \chi_y\}$ of indices on tensors $\{u, v_L, v_R, y_L, y_R, w\}$.

the block of A tensors and from u, v_L, and v_R, as depicted in Fig. 6(b).

After the projective truncation implemented by projector P_u is enacted on all 2×2 blocks of tensors, pairs of disentanglers from neighboring blocks can annihilate to identity, see Fig. 5(b), leaving a network of B tensors interspersed with groups of isometries v_L and v_R. Next, as depicted in Fig. 5(b) and further detailed in Fig. 6(d) and 6(e), a projective truncation is made on B tensors. Two projectors P_L and P_R are used at this step, acting on the left or right indices of the B tensor, respectively, each formed as a product of isometries, $P_L \equiv y_L y_L^\dagger$ and $P_R \equiv y_R y_R^\dagger$. Isometries y_L and y_R, which satisfy $y_L^\dagger y_L = y_R^\dagger y_R = \mathbb{I}'$, each map two indices to a single index also assumed to be of dimension χ',

$$y_L : \mathbb{V}_{\chi'} \to \mathbb{V}_{\chi'} \otimes \mathbb{V}_{\chi'}, \quad y_R : \mathbb{V}_{\chi'} \to \mathbb{V}_{\chi'} \otimes \mathbb{V}_{\chi'}. \tag{25}$$

We then define matrix D from enacting isometries y_L and y_R on tensor B, as shown Fig. 6(e). It is useful, though not strictly necessarily, to work in a gauge where matrix D is diagonal and positive, which can always be achieved through proper choice

of gauge on isometries y_L and y_R, such that the matrix can easily be decomposed as $D = \sqrt{D}\sqrt{D}$.

After projective truncations have been made on B tensors a final projective truncation, as shown Fig. 5(d) and further detailed in Figs. 6(f) and 6(g), is made using projector $P_w \equiv ww^\dagger$ with isometry w mapping two χ-dimensional indices to a single index of dimension χ',

$$w : \mathbb{V}_{\chi'} \to \mathbb{V}_\chi \otimes \mathbb{V}_\chi, \tag{26}$$

where the isometry satisfies $w^\dagger w = \mathbb{I}'$. After projector P_w has been implemented throughout the network, pairs of isometries w from neighboring cells can annihilate to identity with their conjugates, as depicted in Fig. 5(e). This final step yields a coarse-grained square lattice $\mathcal{G}'$ of four index tensors A', whose indices are of some specified dimension $\chi' \leqslant \chi^2$. Notice that the new four-index tensor A' is defined from a product of various tensors obtained throughout the coarse-graining step, namely from $\{v_L, v_R, y_L, y_R, \sqrt{D}, w\}$, as shown Fig. 6(g).

B. Optimization of tensors

Each coarse graining iteration of the binary TNR scheme follows from a series of projective truncations. The projectors involved can be optimized using the iterative SVD update strategy, described in Sec. III C, as we now discuss in more detail.

The first step of the TNR iteration, as depicted Fig. 5(a), requires optimization of a projector P_u composed of isometries v_L and v_R and disentangler u, as shown Fig. 6(c). To update one of these tensors one first computes its environment, where the environments Γ_{v_L}, Γ_{v_R}, and Γ_u are shown in Figs. 7(a)–7(c), then updates the tensor from the SVD of the environment as discussed in Sec. III C. Each of the tensors v_L, v_R, u should be updated in turn and the process iterated until all tensors are sufficiently converged (which typically requires of order a few hundred iterations). The computational cost of computing each of the environments scales as $O(\chi^7)$, assuming the indices involved in the network are all χ-dimensional.

The second projective truncation step of the TNR iteration, as depicted Fig. 5(c), requires the optimization of isometries y_L and y_R, which again are optimized through alternating, iterative SVD updates based on calculation of their environments Γ_{y_L} and Γ_{y_L}, as shown in Figs. 7(d) and 7(e). Note that, in order to ensure that the reflection symmetry (which was assumed to be present in the initial tensor network) is preserved, it is necessary to symmetrize the environments before performing the SVD, as described in Appendix D. The computational cost of computing each of the environments is $O(\chi^5)$ assuming that the indices involved in the network are χ-dimensional.

The third and final projective truncation of the TNR iteration, as depicted Fig. 5(d), requires optimization of isometry w which can be achieved through iterative SVD updates of the environment Γ_w, depicted in Fig. 7(f). The computational cost of computing the environment Γ_w is $O(\chi^6)$ assuming that the indices involved in the network are χ-dimensional.

C. RG flow of tensors

The single iteration of the TNR approach described in Sec. IV A, which mapped the initial tensor network $\mathcal{G}$ to the coarser network $\mathcal{G}'$, can be iterated many times to generate a sequence of increasing coarse-grained networks,

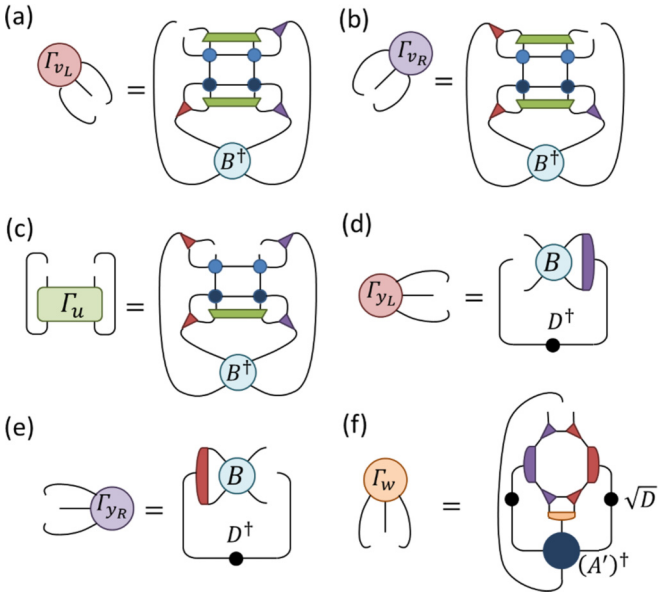

FIG. 7. The linearized environments of tensors $\{v_L, v_R, u, y_L, y_R, w\}$ involved in an iteration of the binary TNR scheme. (a)–(c) Environments Γ_{v_L}, Γ_{v_R}, and Γ_u of the isometries v_L, v_R and disentangler u involved in the first projective truncation of the TNR iteration, as detailed in Fig. 6(a). (d) and (e) Environments Γ_{y_L} and Γ_{y_R} of isometries y_L and y_R from the second projective truncation of the TNR iteration, as detailed in Fig. 6(d). (f) Environment Γ_w of isometry w from the third projective truncation of the TNR iteration, as detailed in Fig. 6(f).

$$\mathcal{G}^{(0)} \to \mathcal{G}^{(1)} \to \mathcal{G}^{(2)} \to \cdots \to \mathcal{G}^{(s)} \to \cdots \tag{27}$$

with $\mathcal{G}^{(0)} \equiv \mathcal{G}$ now as the initial network and network $\mathcal{G}^{(s)}$ as tensor network after s iterations of TNR, where the linear dimension of the lattice $\mathcal{G}^{(s)}$ has been reduced by a factor of 2 compared to the previous network $\mathcal{G}^{(s-1)}$. Each network $\mathcal{G}^{(s)}$ consists of copies of a four-index tensor $A^{(s)}_{ijkl}$, whose indices are of dimensions $\chi^{(s)}$, arranged in a square lattice configuration. Here the initial dimension $\chi^{(0)}$ is fixed from the starting tensors $A^{(0)}_{ijkl}$, while the dimensions $\chi^{(s)}$ at later steps are user specified, and can be increased to improve the accuracy of the calculation at the cost of increasing the computational expense (aspects of computational efficiency will further be discussed in Sec. IV D). Tensors $A^{(s+1)}$ are defined from (four copies of) the previous tensor $A^{(s)}$ under coarse-graining via a product of optimized isometric tensors,

$$A^{(s)} \xrightarrow{\left\{v_L^{(s)}, v_R^{(s)}, u^{(s)}, y_L^{(s)}, y_R^{(s)}, w^{(s)}\right\}} A^{(s+1)}, \tag{28}$$

as depicted in Fig. 6(g), such that we can consider TNR as generating an RG flow in the space of (four-index) tensors,

$$A^{(0)} \to A^{(1)} \to A^{(2)} \to \cdots \to A^{(s)} \to \cdots . \tag{29}$$

Properties of the system under consideration, such as the expectation values of local observables, can be computed from both the coarse-grained tensors $A^{(s)}$ and the tensors

$\{u^{(s)}, v_L^{(s)}, v_R^{(s)}, y_L^{(s)}, y_R^{(s)}, w^{(s)}\}$ involved in the coarse-graining, as further discussed in Sec. V.

D. Algorithmic details

While the TNR algorithm discussed thus far is a viable implementation of approach, we now detail how this basic TNR algorithm can be improved in a number of ways.

In most calculations it is convenient to use the same bond dimensions $\chi^{(s)}$ for all RG steps s, i.e., such that $\chi^{(s)} = \chi$ for $s > 0$ (where $\chi^{(0)}$ is fixed by the local dimension of the model under consideration). However, even when using the same bond dimension throughout different coarse-graining iterations, it can still be useful, in order to maximize the efficiency of the TNR approach, to use different bond dimensions on vertical and horizontal indices of tensors $A^{(s)}$, as well as different dimensions on tensors that appear at intermediate steps of each coarse-graining iteration. Following this idea, four different refinement parameters $\{\chi_u, \chi_v, \chi_w, \chi_y\}$ can be defined, each denoting a dimension of an outgoing index on different isometric tensors as indicated in Fig. 6(h). Here χ_w and χ_y denote the dimensions of vertical and horizontal indices on A tensors, respectively, while χ_u and χ_v denote dimensions of indices that only appear at intermediate steps of each TNR iteration. The ratio of dimensions should be adjusted such that the truncation errors ε given at different intermediate steps become roughly equal; in practice such dimension can be determined heuristically. For several test models it has been observed that the different optimal bond dimensions are, to good approximation, linearly related to one another (i.e., the ratios of different optimal bond dimensions remain roughly constant for a given model). This implies that the cost scaling of an algorithmic step can be specified unambiguously, for instance as having cost $O(\chi^7)$, without detailing the dependence on the different dimensions involved [as they are linearly related]. Note that in the benchmark results presented in Sec. VI we label results of a TNR simulation by a single dimension χ, by which we mean the *largest* of the bond dimensions $\{\chi_u, \chi_v, \chi_w, \chi_y\}$ that was used in the simulation.

The computational cost of the binary TNR can be reduced by implementing an additional projective truncation on the 2×2 blocks of A tensors at the start of the TNR iteration, as described in Appendix E. This additional step reduces the cost of computing the environments Γ_{v_L}, Γ_{v_R}, and Γ_u, see Figs. 7(a)–7(c), from $O(\chi^7)$ to $O(\chi^6)$, thus also reducing the cost of optimizing tensors v_L, v_R, and u. When utilizing the ideas of Appendix E the tensor contractions required to implement TNR are all of cost $O(\chi^6)$ or less; thus the efficiency of the algorithm is greatly improved.

The accuracy of TNR, for a fixed bond dimension χ, can be improved by taking a larger environment into account for optimization of the tensors, using an approach similar to the approach of Refs. [5,7] for improving standard TRG by taking the environment into consideration. Appendix F describes how the tensors involved in the TNR coarse-graining iteration can be optimized using a larger (though still local) environment, and the benefits of doing so. Note that, by also incorporating the ideas of Appendix E, the leading order computational cost of optimizing using the larger environments remains unchanged at $O(\chi^6)$. It is also possible to modify TNR to take account of the full environment from the network, similarly to how the second renormalization group (SRG) method [5] modifies the TRG to take account of the full environment. This modification, which we shall not detail in the present paper, requires sweeping the optimization back and forth over different scales of coarse-graining, functioning similarly to the energy minimization algorithm for optimizing a MERA [19].

V. CALCULATION OF OBSERVABLES

In this section we discuss how TNR can be applied to compute expectation values of local observables in classical statistical or quantum many-body systems. There are many different ways of performing this calculation; one approach could be to construct a MERA from the TNR coarse-graining transformations, as described in Ref. [18], from which expectation values could then be computed using standard MERA techniques [19]. In this paper we describe a different approach based on directly coarse-graining the network with the addition of an impurity tensor.

Let $\mathcal{G}^{(0)}$ be a homogeneous square lattice tensor network with periodic boundaries that consists of an $L \times L$ array of copies of a four-index tensor $A^{(0)}$. Assume that the sequence of $T = \log_2(L/2)$ TNR transformations have been optimized as to generate a sequence of coarser lattices, see Eq. (27), where $\mathcal{G}^{(T)}$ is a 2×2 lattice of tensors $A^{(T)}$ that can be exactly contacted. In order to evaluate the expectation value of a local observable, we must now evaluate the tensor network $\mathcal{G}^{(0)}$ with the addition of an impurity tensor representing the local observable under consideration. Here we consider an impurity tensor $M^{(0)}$ that replaces a 2×2 block of A tensors from $\mathcal{G}^{(0)}$. To evaluate the impurity network we use the same projective truncations as was used to coarse-grain the homogeneous network $\mathcal{G}^{(0)}$ everywhere except in the immediate vicinity of the impurity, the presence of which is incompatible with the coarse-graining steps used for the homogeneous system; see Figs. 8(a)–8(e). After an iteration of TNR to the impurity network, one obtains a new impurity network, equal to the homogeneous network $\mathcal{G}^{(1)}$ except where a new impurity $M^{(1)}$ replaces a 2×2 block of tensors $A^{(1)}$; see Fig. 8(f). The coarse-grained impurity $M^{(1)}$ is defined from applying a set of two-body gates, $\{G_R, G_L, G_U, G_Y\}$, to the initial impurity tensor $M^{(0)}$, as depicted in Fig. 8(g), where the gates are functions of $A^{(0)}$ and the isometries $\{v_L, v_R, u, y_L, y_R, w\}$ used in the TNR iteration, as depicted in Fig. 8(h). The coarse-graining transformation can be applied to the impurity network multiple times as to generate a sequence of impurity tensors,

$$M^{(0)} \to M^{(1)} \to \cdots \to M^{(T)}, \tag{30}$$

each imbedded in an increasingly coarse-grained square-lattice network. We call the transfer operator that maps impurity tensors from one length scale to the next, which was introduced in Ref. [15], the *ascending superoperator* $\mathcal{R}$,

$$M^{(s+1)} = \mathcal{R}(M^{(s)}), \tag{31}$$

as shown in Fig. 8(g). After $T = \log_2(L/2)$ RG steps, the impurity network only contains a single tensor $M^{(T)}$, in place of the 2×2 block of tensors $A^{(T)}$ that would otherwise have been in the homogeneous network $\mathcal{G}^{(T)}$, from which the expectation

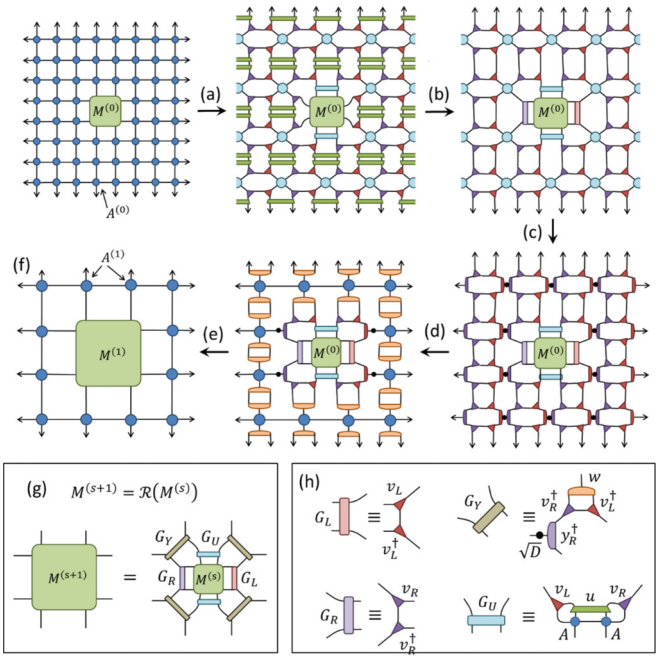

FIG. 8. (a)–(e) An iteration of the TNR coarse-graining transformation for a square-lattice tensor network with an impurity tensor $M^{(0)}$. The projective truncations of the TNR iteration are applied as usual, see Fig. 5, neglecting those which are incompatible impurity tensor. (f) The coarse-grained square lattice is homogeneous everywhere except for a 2×2 region occupied by new impurity tensor $M^{(1)}$. (g) The coarse-grained impurity tensor $M^{(s+1)}$ is given by enacting ascending superoperator $\mathcal{R}$, defined from two-body gates $\{G_R, G_L, G_U, G_Y\}$, on the impurity tensor $M^{(s)}$. (h) Definition of two-body gates $\{G_R, G_L, G_U, G_Y\}$.

value is evaluated through the appropriate trace of $M^{(T)}$ as depicted in Fig. 9.

Note that the evaluation of two-point correlators, which corresponds to contracting a network with two local impurities, can be handled similarly. In this case each of the impurities will transform individually in the same way as Eq. (31) until they become adjacent to one another in the network, where they will then fuse into a single impurity after the next TNR iteration.

The evaluation of local observables can also be formulated in a more general way, using TNR to map the network on the punctured plane to an open cylinder, analogous to the logarithmic transformation in CFT [12,13]. This mapping using TNR was originally formulated in Ref. [15], to which we

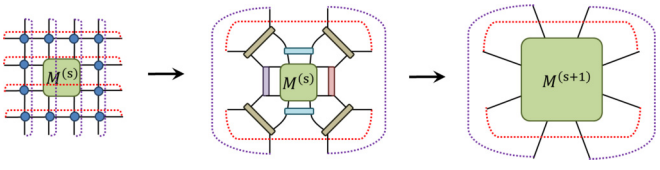

FIG. 9. A finite network with periodic boundaries and a local impurity $M^{(s)}$ is coarse-grained into a single-impurity tensor $M^{(s+1)}$ under the transformation depicted in Fig. 8. The expectation value of the local observable corresponding to the impurity is given from the trace of $M^{(s+1)}$ as depicted.

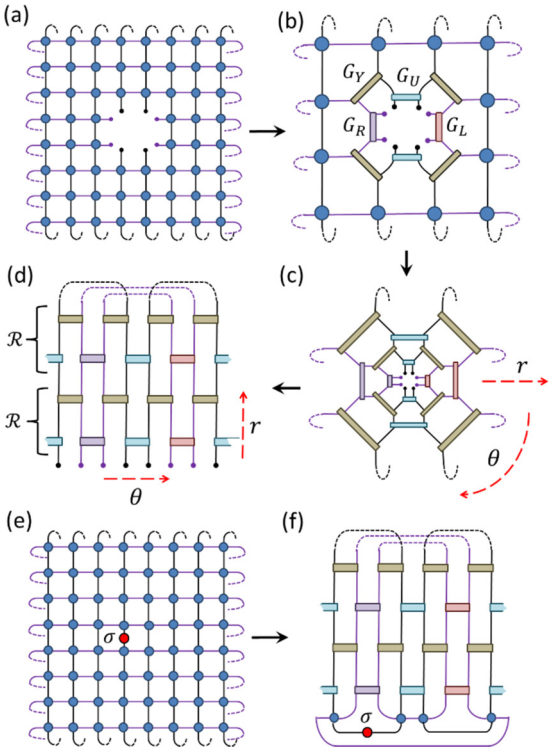

FIG. 10. (a) An 8×8 square lattice of tensors that has had a 2×2 block of tensors removed, leaving 8 open indices. (b) The lattice after one iteration of coarse-graining with TNR. (c) The lattice after a second iteration of coarse-graining with TNR. (d) The network from (c) is redrawn on the cylinder. Each double layer of the network corresponds to the ascending superoperator $\mathcal{R}$ as defined in Fig. 8(g). (e) An 8×8 square lattice of tensors with an impurity σ positioned on a link. (f) The network from (e) after two iterations of coarse-graining with TNR.

refer the interested reader for more details. A brief summary is included here for completeness. In Fig. 10(a) we consider a finite square lattice network that has a 2×2 block of tensors removed, such that the indices connecting to the block are left open. As with the case of the impurity network considered previously, one can coarse-grain this network with TNR, performing the same projective truncations except in the immediate vicinity of the open indices, which must remain untouched. The result of this is shown in Figs. 10(a)–10(d); the square lattice network (with an open "hole") is mapped to a tensor network on a finite-width cylinder. Here one end of the cylinder has free indices, which correspond to those of the open "hole," and moving along the length of the cylinder corresponds to a change of *scale* in the original system, where each double row of tensors in the cylinder corresponds to an application of the ascending superoperator $\mathcal{R}$ as defined in Fig. 8(g).

Using the same transformations, the tensor network with local impurity, as depicted in Fig. 10(e), is mapped to a closed cylindrical network after coarse-graining with TNR, as depicted in Fig. 10(f). Contracting this network from bottom to top would be equivalent to evaluating the expectation value by coarse-graining the observable as discussed previously; see Eq. (30). However, one could also evaluate the observable by

contracting the network from top to bottom, or in some other desired order. In practice, exact contraction of the network shown in Fig. 10(f) is computationally expensive for large bond dimension χ; thus it may be necessary to use an approximate method for this evaluation. The approximate method we employ in this paper is based on contracting the network layer by layer from top to bottom, where we approximate the boundary state as a matrix product state [34,35] (MPS) and use the TEBD algorithm [36,37] to apply each layer of gates from the cylinder while maintaining an MPS representation.

VI. BENCHMARK RESULTS

A. 2D classical Ising model

In this section we provide benchmark calculations for the binary TNR algorithm, comparing against TRG, for the partition function of the $2D$ classical Ising model, as defined in Eqs. (1) and (2). We begin by encoding the partition function as a tensor network as discussed in Sec. II A: a four-index tensor A_{ijkl} is used to encode the four Boltzmann weights on the edge of a plaquette, as per Eq. (3), which corresponds to having one tensor A for every two spins and a tensor network with a 45° tilt with respect to the spin lattice; see Figs. 1(a) and 1(b). For convenience we then contract a 4×4 square of tensors A to form a new tensor $A^{(0)}$ of bond dimension $\chi = 16$, which serves as the starting point for both the TNR and TRG approaches. We apply up to 20 RG steps of either TNR or TRG, which corresponds to lattices of Ising spins with linear dimension up to $L = 4 \times 2^{20} \approx 4 \times 10^6$ spins. Here we define an RG step as that which maps the square lattice to a new square lattice of the same orientation, but of half the linear dimension; note that this corresponds to two steps of TRG as defined in Ref. [2]. Each of the approaches generates a sequence of coarse-grained tensors, as per Eq. (29), where copies of tensor $A^{(s)}$ comprise the coarse-grained network after s RG steps.

In the implementation of TNR we enforce reflection symmetry, as discussed in Appendix D, on both spatial axes, and also employ the ideas discussed in Appendix E to reduce the leading order cost of the TNR algorithm from scaling as $O(\chi^7)$ to scaling as $O(\chi^6)$ in terms of the bond dimension χ. Furthermore we optimize tensors using the larger environment as discussed in Appendix F, and also use Z_2-invariant tensors (recall that the Ising model has a global Z_2 symmetry: it is invariant under the simultaneous flip $\sigma_k \to -\sigma_k$ of all the spins), which are employed using standard methods for incorporating global symmetries in tensor networks [38]. The TRG results have been calculated using the square-lattice TRG algorithm as presented in Ref. [2], the cost of which also scales as $O(\chi^6)$ in terms of the bond dimension χ. While the cost of TRG and TNR both scale as $O(\chi^6)$, the overall cost of a TNR calculation is greater than a TRG calculation of the same bond dimension by a constant factor, $k \approx 200$, stemming from the iterative nature of the TNR optimization.

Figure 11(a) explores the truncation error ε, as defined in Eq. (11), incurred in RG step s of the either the TRG or TNR approach applied to the Ising model at critical temperature, $T_c = 2/\ln(1+\sqrt{2}) \approx 2.269$. It is seen that, although increasing the bond dimension χ of TRG reduces the initial truncation error ε, the error always increases quickly as a function of RG step s regardless of the bond dimension in use, a phenomenon described as the breakdown of TRG at criticality in the original work by Levin and Nave [2]. In comparison it is seen that TNR avoids such a breakdown; the truncation errors ϵ remain constant over many RG steps s when coarse-graining with TNR, beyond a slight increase after the initial RG step.

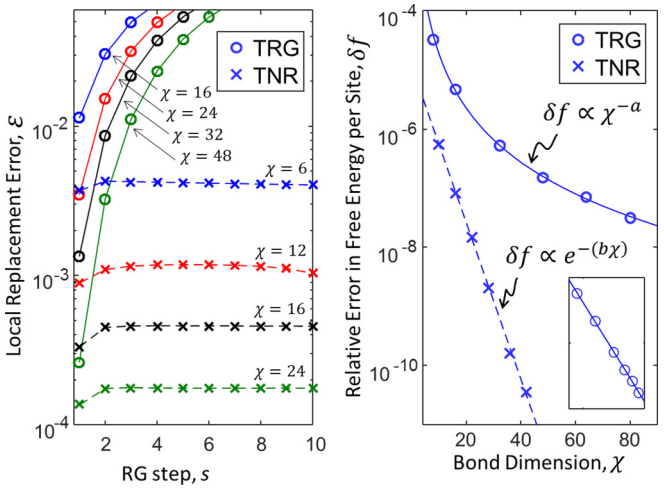

FIG. 11. (a) Comparison between TRG and TNR of the truncation error ε, as defined in Eq. (11), as a function of RG step s in the $2D$ classical Ising model at critical temperature T_c. While increasing the bond dimension χ gives smaller truncation errors, the truncation errors still grow quickly as a function of RG step s under TRG. Conversely, truncation errors remain stable under coarse-graining with TNR. (b) Relative error in the free energy per site δf at the critical temperature T_c, comparing TRG and TNR over a range of bond dimensions χ. The error from TRG is seen to diminish polynomially with bond dimension, with fit $\delta f \propto \chi^{-3.02}$ (where the inset displays the same TRG data with logarithmic scales on both axes), while the error from TNR diminishes exponentially with bond dimension, with fit $\delta f \propto \exp(-0.305\chi)$. Extrapolation suggests that TRG would need bond dimension $\chi \approx 750$ to match the accuracy of the $\chi = 42$ TNR result.

The relative error in the free energy per site, $f = -T\ln(Z)/N$, at the critical temperature T_c is compared for TRG and TNR over a range of bond dimensions χ in Fig. 11(b). Evident is a qualitative difference in the convergence of the free energy between the approaches. In TRG the free energy f is seen to converge polynomially with χ, with fit $\delta f \propto \chi^{-3.02}$, while in TNR f is seen to converge exponentially fast with χ, with fit $\delta f \propto \exp(-0.305\chi)$. Given that the cost of implementing the two approaches differs only by a constant factor, $k \approx 200$, for equivalent bond dimension χ, it is evident that TNR can produce a significantly more accurate free energy than would be computationally viable with standard TRG. For instance, extrapolation suggests that in order to match the accuracy in the free energy for $\chi = 42$ TNR, which required approximately 12 hours computation time on a laptop computer, one would need to implement TRG with $\chi \approx 750$, far beyond what is considered feasible with the approach. Figure 12 shows the spontaneous magnetization $M(T)$ and specific heat $c(T) = -T\frac{\partial^2 f}{\partial T^2}$ obtained with TNR for $\chi = 6$,

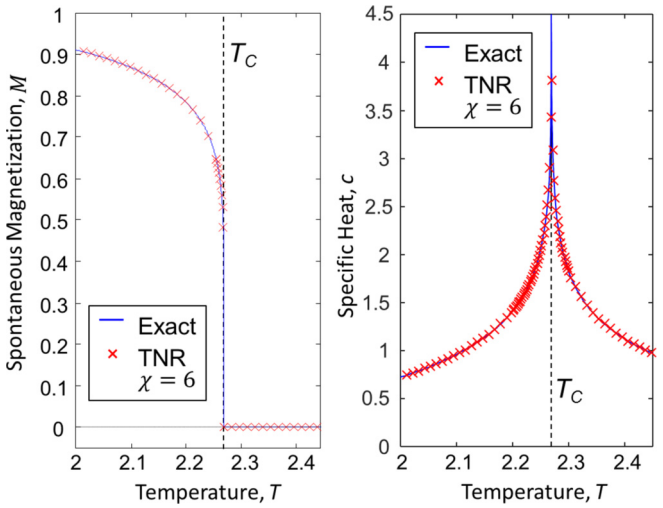

FIG. 12. (a) Spontaneous magnetization $M(T)$ of the $2D$ classical Ising model near critical temperature T_c, both exact and obtained with TNR with $\chi = 6$. Even very close to the critical temperature, $T = 0.9994\ T_c$, the magnetization $M \approx 0.48$ is reproduced to within 1% accuracy. (b) Specific heat, $c(T) = -T\frac{\partial^2 f}{\partial T^2}$, both exact and obtained using TNR with $\chi = 6$.

over a range of temperatures T near the critical temperature T_c. Remarkable agreement with the exact results is achieved throughout, even very close to the critical point T_c.

Next we explore the ability of TNR to produce a scale-invariant RG flow in the space of tensors $A^{(s)}$ for the Ising model at critical temperature T_c. We apply up to 20 RG steps of TNR to the partition function, employing the gauge-fixing strategy discussed in Appendix G from the third RG step onwards. The difference $\delta^{(s)} \equiv \|A^{(s)} - A^{(s-1)}\|$ between tensors at successive RG steps (where tensors have been normalized such that $\|A^{(s)}\| = 1$) is displayed in Fig. 13. For the larger χ calculations, the difference $\delta^{(s)}$ reduces after the initial RG steps, before increasing again in the limit of many RG steps. This behavior is to be expected. The main limitation to realizing scale-invariance exactly in the initial RG steps is physical: the lattice system includes RG-irrelevant terms that break scale invariance at short-distance scales, but are suppressed at larger distances. On the other hand, after many RG steps the main obstruction to scale invariance is the numerical truncation errors, which can be thought of as introducing RG-relevant terms, effectively shifting the flow away from criticality and thus away from scale invariance. However, use of a larger bond dimension χ reduces truncation errors, allowing TNR to not only achieve a more precise approximation to scale invariance, but also to hold it for more RG steps.

In order to demonstrate that the (approximate) fixed-point map given by TNR is representative of the $2D$ Ising universality class we extract the critical data from this map. This calculation was previously carried out in Ref. [15], to which we refer the interested reader for more details. Scaling operators, with their corresponding scaling dimensions, are obtained from diagonalization of the ascending superoperator $\mathcal{R}$, see Fig. 8(a), associated with the $s = 4$ coarse-graining iteration with TNR. At this level of coarse-graining the RG-irrelevant

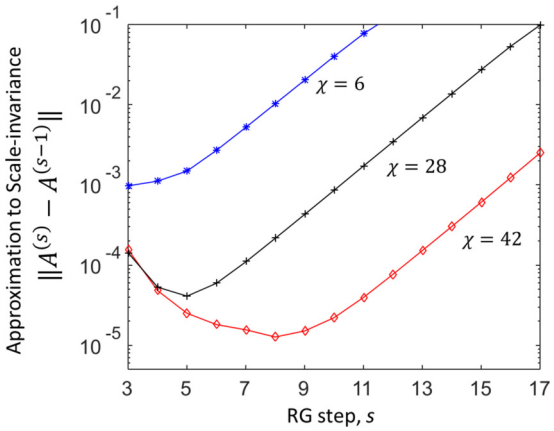

FIG. 13. The precision with which TNR approximates a scale-invariant fixed-point tensor for the $2D$ classical Ising model at critical temperature T_c is examined by comparing the difference between tensors produced by successive TNR iterations $\delta^{(s)} \equiv \|A^{(s)} - A^{(s-1)}\|$, where tensors have been normalized such that $\|A^{(s)}\| = 1$. The precision with which scale invariance is approximated in the initial RG steps (small s) is limited by the presence of RG-irrelevant terms in the lattice Hamiltonian that break scale invariance at short-distance scales, while numerical truncation errors, which can be thought of as introducing RG-relevant terms, shift the system from criticality (and thus scale invariance) in the limit of many RG steps s.

terms in the system have been sufficiently suppressed, such that a good approximation to scale-invariance is realized. The smallest 101 scaling dimensions obtained from a $\chi = 6$ TNR calculation are displayed in Fig. 14, all of which are within 2% of their exact values. Also computed were the operator product expansion (OPE) coefficients [12,13] for the primary fields, see again Ref. [15] for details, which were found to match their correct values from CFT to within 0.2%. These results indeed confirm that the fixed-point RG map given by TNR is accurately characterizing the Ising CFT.

B. 1D quantum Ising model

In this section we provide benchmark calculations for the binary TNR algorithm applied to the Euclidean path integral

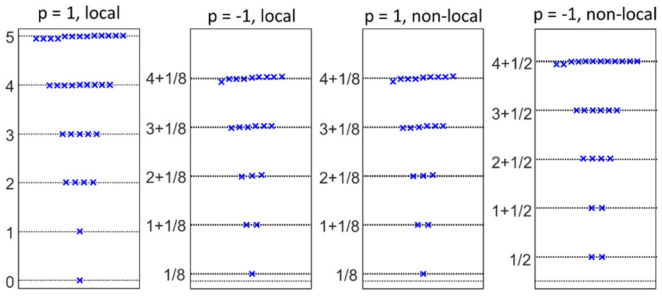

FIG. 14. The smallest 101 scaling dimensions of the $2D$ classical Ising model at critical temperature T_c, obtained by diagonalizing the ascending superoperator $\mathcal{R}$, see Fig. 8(g), using TNR with bond dimension $\chi = 6$. The scaling dimensions are organized according to the parity $p = \pm 1$ (even/odd) and locality of the corresponding scaling operators. All scaling dimensions shown are reproduced to within 2% of their exact values.

of the $1D$ quantum Ising model, which has Hamiltonian $H_{\rm Is}$ defined

$$H_{\rm Is} = \sum_k \left(\sigma_k^x \sigma_{k+1}^x + \lambda \sigma_k^z\right), \tag{32}$$

where σ^x and σ^z are Pauli matrices, and λ represents the magnetic field strength. For convenience we perform an initial blocking step, in which blocks of four spins are combined into effective sites of local dimension $d = 16$, and then generate the tensor network representation of the Euclidean path integral as explain in Sec. II B. We typically use a time step of $\tau = 0.002$ and coarse-grain in the Euclidean time direction using 10 iterations of the single-dimension coarse-graining scheme explained in Appendix A, before beginning the TNR calculation.

To start with, we compare the performance of the TNR algorithm as a means to optimize a MERA [17] versus the energy minimization approach of Ref. [19]. The ground state of the Ising Hamiltonian $H_{\rm Is}$ at critical magnetic field strength, $\lambda = 1$, is represented using a scale-invariant MERA [19,21–24], here consisting of three transitional layers followed by infinitely many copies of a scale-invariant layer, obtained in two different ways. In the first approach, we apply four coarse-graining iterations of TNR to the Euclidean path integral of $H_{\rm Is}$, after which the flow in the space of tensors has reached an approximate scale-invariant fixed point. A scale-invariant MERA is then built from certain tensors that were produced from the TNR calculation, specifically the disentanglers u and isometries w, as described in Ref. [18]. Here the first three iterations of TNR generate the three transitional layers, while the fourth iteration of TNR is used to build the scale-invariant layer. In the second approach, we use the energy minimization algorithm of Refs. [19,23] to directly optimize a scale-invariant MERA (with three transitional layers preceding the scale-invariant layers) by iteratively minimizing the expectation value of the H. Presented in Fig. 15(a) is the comparison between the ground state energy error δE obtained by the two methods. For an equivalent bond dimension χ, energy minimization produces a MERA with smaller error δE than TNR, by a factor $k \approx 10$ that is roughly independent of χ. However, optimization using TNR is also computationally cheaper than optimization using the energy minimization algorithm; the cost of TNR scales as $O(\chi^6)$ in terms of bond dimension χ, while the energy minimization scales as $O(\chi^9)$. In addition, the energy minimization algorithm, which iteratively sweeps over all MERA layers, requires more iterations to converge than does TNR. Taking these considerations into account, TNR can be seen to be the more efficient approach to obtain a ground state MERA to within a given level of accuracy in the energy δE; see Ref. [18] for additional details. Thus the TNR approach represents a useful, alternative means for optimizing a MERA and, by extension, is promising as a tool for the exploration of ground states of quantum systems.

In Fig. 15(b) the low-energy excitation spectra of $H_{\rm Is}$ at criticality, $\lambda = 1$, are plotted as a function of system size L. This was computed using TNR to coarse-grain the Hamiltonian $H_{\rm Is}$, as explained in Ref. [18], which was then diagonalized on a finite lattice with periodic boundaries. The excitation spectra reproduce the expected $1/L$ scaling with system size and match the predictions from CFT [12,13], demonstrating that, in addition to the ground state, TNR can accurately approximate the low-energy eigenstates of $H_{\rm Is}$.

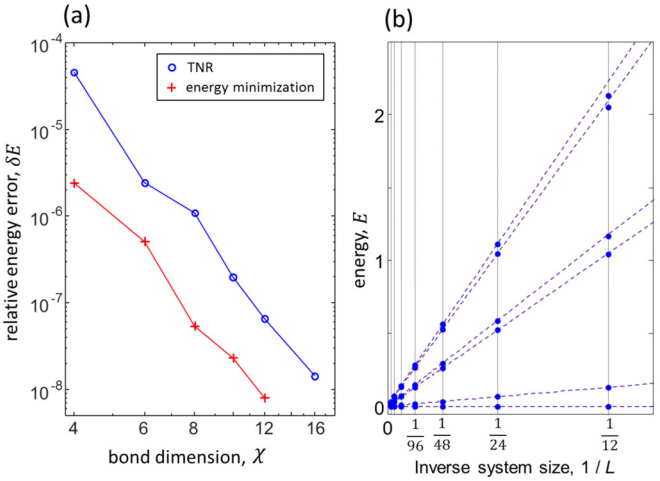

FIG. 15. (a) Relative error in the energy of scale-invariant MERAs optimized for the ground state of the $1D$ quantum Ising model at criticality in terms of bond dimension χ, comparing MERAs optimized using TNR to those optimized using variational energy minimization. Energy minimization produces MERAs with a more accurate approximation to the ground state energy, but is significantly more computationally expensive [with a computational cost that scales as $O(\chi^9)$ versus as $O(\chi^6)$ for TNR]. (b) Low-energy eigenvalues of the $1D$ quantum Ising model at criticality as a function of $1/L$, computed with $\chi = 12$ TNR. Discontinuous lines correspond to the finite-size CFT prediction, which ignores corrections of order L^{-2}.

Next we explore the use of TNR for computing expectation values of finite-temperature thermal states. This is achieved by applying TNR to the tensor network corresponding to $e^{-\beta H_{\rm Is}}$ for inverse temperature β, as discussed in Sec. II B, which is then used to generate a thermal state MERA as explained in Ref. [18]. The thermal energy per site as a function of β is displayed in Fig. 16(a) for several different magnetic field strengths λ. In the gapped regime, $\lambda > 1$, the thermal energy converges exponentially quickly to the ground state energy in the limit of large β (or small temperature T), while at the gapless critical point, $\lambda = 1$, the thermal energy converges polynomially quickly to the ground state energy with β. The results from $\chi = 12$ TNR accurately reproduce the exact energies over the range of parameters considered. Two-point correlation functions of the critical, $\lambda = 1$, system are examined in Fig. 16(b) over a range of inverse temperatures β, where correlations are seen to decay faster at smaller β (or equivalently at higher temperature T) due to thermal fluctuations. Again, the results from $\chi = 12$ TNR accurately reproduce the exact correlations, indicating that TNR can be used to approximate thermal states of quantum systems over a wide range of temperatures.

VII. DISCUSSION

After reviewing the conceptual foundations for real-space renormalization of partition functions and Euclidean path integrals, when expressed as tensor networks, a self-contained

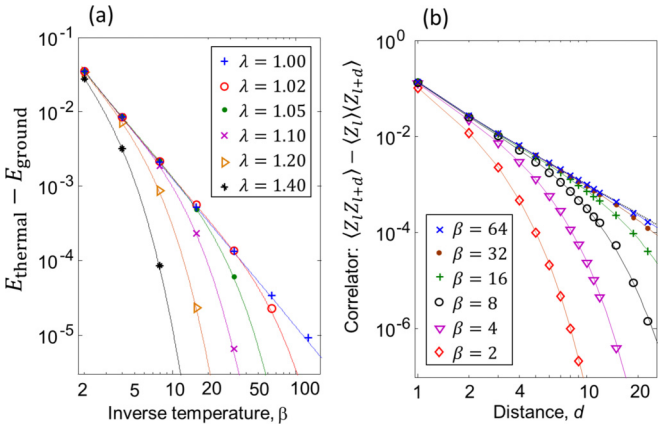

FIG. 16. (a) Thermal energy per site (above the ground state energy) as a function of the inverse temperature β, for the $1D$ quantum Ising model in an infinite chain, for different values of magnetic field λ. Data points are computed with $\chi = 12$ TNR while continuous lines correspond to the exact solution. (b) Connected two-point correlators at the critical magnetic field $\lambda = 1$, as a function of the distance d, for several values of β. Data points are computed with $\chi = 12$ TNR while continuous lines again correspond to the exact solution.

description of the algorithm for employing TNR to study properties of classical and quantum many-body systems was provided.

Benchmark results, provided in Sec. VI, demonstrated some of the advantages of TNR. These include (i) providing a computationally sustainable coarse-graining transformation even for systems at or near critical point (i.e., one that can be iterated many times without increase in truncation error), (ii) convergence in the RG flow of tensors to a scale-invariant fixed point for a critical system (which then allows calculation of the critical data from the fixed-point RG map directly), (iii) providing an alternate, more efficient means for optimizing a MERA for the ground state of a quantum system, and (iv) providing a means to accurately study properties of thermal states of quantum systems over a wide range of temperatures T. With regards to (i) above, TNR was demonstrated to overcome a major obstacle of previous schemes for renormalization of tensor networks, such as TRG, which exhibit a computational breakdown when near or at a critical point. As a consequence, the free energy per site f of the $2D$ classical Ising model at criticality was seen to converge to the exact value exponentially faster in bond dimension χ with TNR than with the previous TRG approach.

Future work shall include the development and implementation of TNR schemes for the coarse-graining of networks on $3D$ lattices, which could be applied to study $3D$ classical statistical and $2D$ quantum many-body systems. A TNR scheme for $3D$ lattices would offer an alternative, potentially more efficient, means to optimize a $2D$ MERA over the previous (often prohibitively expensive) strategies based on energy minimization [39,40]. The TNR approach presented in this paper can also be used to compute the norm $\langle\Psi|\Psi\rangle$ of a $2D$ quantum many-body state encoded in a projected entangled pair state (PEPS) [41,42]; thus it could be incorporated as a key part of an algorithm for simulation of $2D$ quantum many-body systems using PEPS [43].

ACKNOWLEDGMENTS

The author thanks Markus Hauru and Guifre Vidal for insightful comments. The author acknowledges support by the Sherman-Fairchild Foundation and by the Simons Foundation (Many Electron Collaboration).

APPENDIX A: COARSE-GRAINING ALONG A SINGLE DIMENSION

In this Appendix we describe a scheme that coarse-grains a square lattice along one dimension only, which is similar in effect to the higher-order tensor renormalization group (HOTRG) method introduced in Ref. [8]. It is useful to perform this coarse-graining to rescale one dimension before applying the TNR algorithm if the initial tensor network is highly anisotropic in the strength of its correlations (which occurs, for instance, when the network represents the Euclidean path integral of a quantum system expanded with a very small time step τ). The preliminary coarse-graining can generate a network that is more isotropic in the strength of its correlations and thus more suitable as a starting point for the TNR approach, which then rescales both lattice dimensions equally.

An iteration of the coarse-graining, which acts to compress the vertical dimension of the square-lattice tensor network by a factor of two, is depicted in Fig. 17. The first step, as shown Fig. 17(a), involves a projective truncation that acts on a pair of A tensors, as further detailed in Fig. 17(b). The isometries w that comprise the projector can be optimized in the standard way, as discussed in Sec. III C. The second step, as shown in Fig. 17(c), involves contraction of a pair of A tensors with isometries w, as detailed in Fig. 17(d), to generate the tensors A' of the coarse-grained network.

APPENDIX B: TERNARY TNR SCHEME

The binary TNR scheme presented in Sec. IV A is one of many possible TNR schemes for the square lattice; here

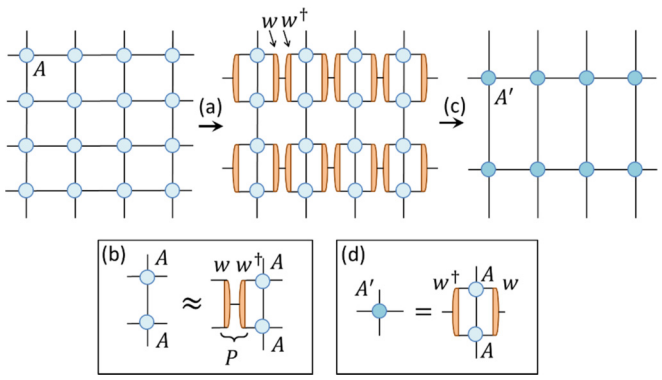

FIG. 17. An iteration of a coarse-graining scheme that acts to compress only the vertical dimension of the square-lattice tensor network. (a) A projective truncation, involving isometries w, is employed. (b) Details of the projective truncation from (a); here a projector P, composed of an isometry w and its conjugate, is applied to a pair of A tensors. (c) The coarse-grained network, composed of tensors A', is given via tensor contractions. (d) Definition of the coarse-grained tensor A'.

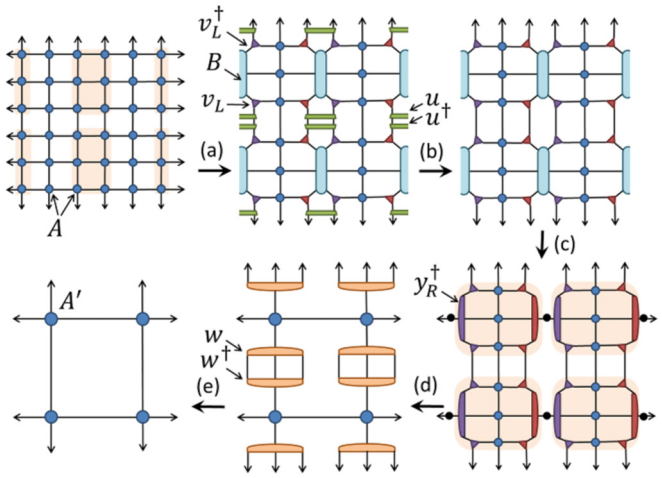

FIG. 18. A depicted of an iteration of the ternary TNR scheme, which maps a square-lattice tensor network to a coarse-grained square lattice of one-third the linear dimension of the original. (a) A projective truncation, detailed in Fig. 19(a). (b) Pairs of disentanglers u annihilate to identity. (c) A projective truncation, detailed in Fig. 19(b). (d) A projective truncation, detailed in Fig. 19(c). (e) Pairs of isometries w annihilate to identity, yielding the coarse-grained square lattice network.

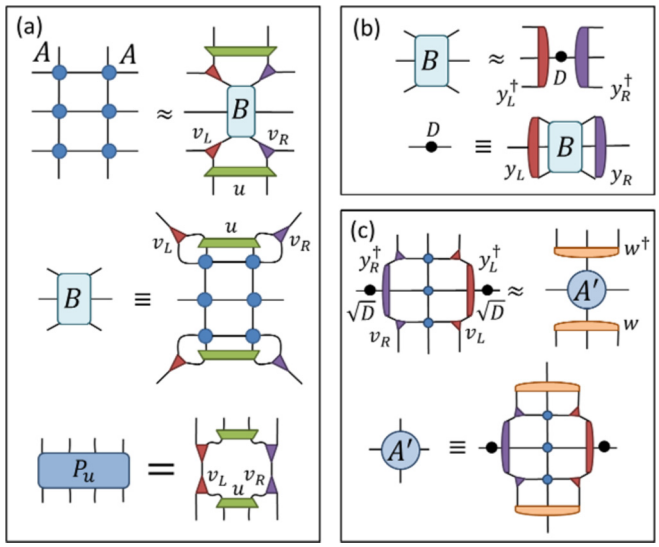

FIG. 19. Overview of the projective truncations involved in the ternary TNR scheme. (a) A projective truncation, implemented by projector P_u composed of isometries v_L, v_R and disentangler u, acts upon a 3×2 block of A tensors. (b) A projective truncation, formed from isometries y_L, y_R and their conjugates, acts upon B tensors. (c) A projective truncation, formed from isometry w and its conjugate, acts to give the new four-index tensor A' of the coarse-grained network.

we describe a ternary TNR scheme which reduces the linear dimension of the lattice by a factor of 3 with each iteration (or, equivalently, coarse-grains a 3×3 block of tensors into a new tensor with each iteration). One advantage of the ternary TNR scheme is that single-tensor impurities can remain single-tensor impurities under coarse-graining, in contrast to the binary TNR scheme in which impurities spread to 2×2 blocks as depicted Fig. 8. This property could be convenient for certain types of calculation, such as in the evaluation of conformal data from a critical system. Through a similar derivation as presented in Ref. [18], where the binary TNR scheme was shown to yield a binary MERA, the ternary TNR scheme can be shown to yield a ternary MERA [19].

The steps of an iteration of the ternary TNR scheme are shown in Figs. 18(a)–18(e). These steps are as follows: (a) a projective truncation involving isometries v_L, v_R and disentangler u as detailed in Fig. 19(a), (b) annihilation of conjugate pairs of disentanglers u to identity, (c) a projective truncation involving isometries y_L and y_R as detailed in Fig. 19(b), (d) a projective truncation involving isometry w as detailed in Fig. 19(c), (e) annihilation of conjugate pairs of the isometry w to identity. The four-index tensor A' of the coarse-grained network, as defined in Fig. 19(c), now accounts for a 3×3 block of tensors A from the original square-lattice network.

APPENDIX C: ISOTROPIC TNR SCHEME

In this appendix we present another TNR scheme for the square lattice. Like the binary TNR scheme presented in the main text, this scheme also reduces the linear dimension of the lattice by a factor of 2 with each iteration. However, unlike the binary TNR scheme, this scheme treats both dimensions of the lattice equally, for which we refer to it as the *isotropic* TNR scheme. Thus, if the starting network that is invariant under 90° rotations, such as that corresponding to the partition function of an isotropic $2D$ classical statistical model, this TNR scheme can preserve the rotational symmetry under coarse-graining.

The isotropic TNR scheme is applied to a square-lattice network with a four-site unit cell, where it is assumed that the network consists of three types of four-index tensor, A_b, A_p, and A_r. As shown in Fig. 20 the lattice has a 2×2 unit cell where each A_b tensor connects with four A_p tensors in the network, and likewise each A_r tensor also connects with four A_p. It is assumed that A_p and A_r tensors are invariant with respect to 90° rotations, while A_p tensors are invariant with respect to 180° rotations; under this assumption the square-network itself is invariant with respect to 90° rotations centered about either an A_p or A_r tensor. Notice that a uniform (one-site unit cell) square lattice is just a special case of this four-site unit cell lattice, where indices on one of the sublattices, such as indices connected to A_r tensors, are fixed at trivial dimension, i.e., bond dimension $\chi = 1$. Thus this isotropic TNR scheme can be directly applied to the partition function of an isotropic $2D$ classical statistical model, when it is represented as a uniform square-lattice network of tensors A that are invariant with respect to 90° rotations.

The steps of a coarse-graining iteration of the isotropic TNR scheme are shown in Figs. 20(a)–20(d). The initial step, shown in Fig. 20(a), involves two different projective truncations. One of the projective truncations, as detailed in Fig. 21(a), acts upon a block of four A_p tensors and a A_b tensor, while the other acts upon individual A_r tensors as detailed in Fig. 21(b). In the second step of the iteration, shown in Fig. 20(b), conjugate pairs of disentanglers u annihilate to the identity. A pair of projective truncations are used in the third step of the iteration,

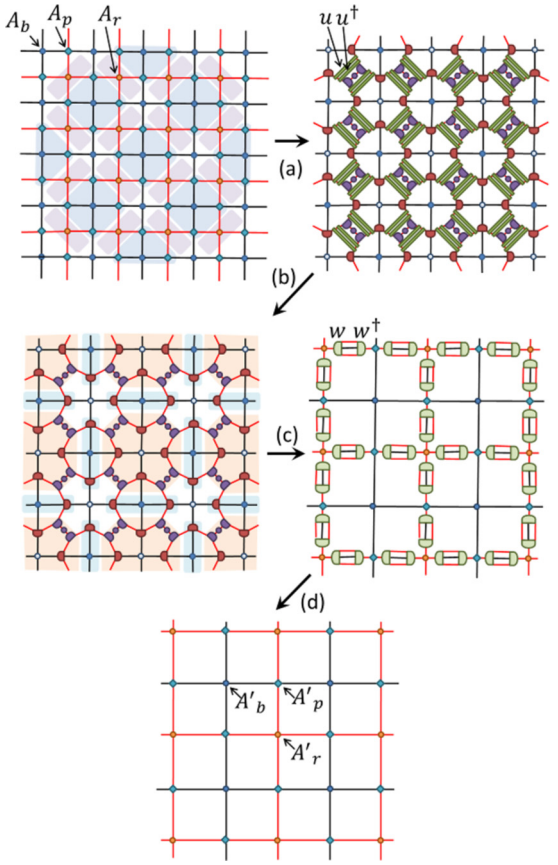

FIG. 20. Depiction of a single iteration of the isotropic TNR scheme, which maps a square-lattice network, composed of tensors A_b, A_p, and A_r, to a coarser lattice of the same type. (a) Two types of projective truncation are made, as detailed in Figs. 21(a) and 21(b). (b) Conjugate pairs of disentanglers u annihilate to identity. (c) Two types of projective truncation are made, as detailed in Figs. 21(c) and 21(d). (d) Conjugate pairs of isometries w annihilate to identity, yielding a coarse-grained network composed of tensors A'_b, A'_p, and A'_r.

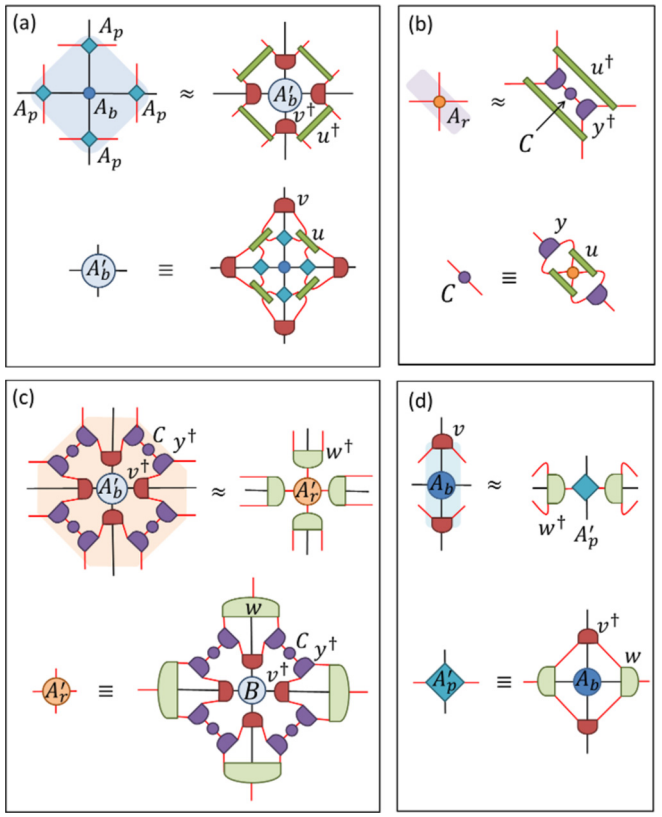

FIG. 21. Overview of the projective truncations involved in an iteration of the isotropic TNR scheme. (a) A projective truncation, involving isometries v and disentanglers u, is enacted upon a product of four A_p tensors and a A_b tensor. The coarse-grained tensor A'_b is also defined. (b) A projective truncation, involving isometries y and disentanglers u, is enacted upon A_r tensors. (c) A projective truncation, involving isometries w, is enacted, yielding coarse-grained tensors A'_r. (d) A final projective truncation, again involving isometries w, is applied to a product of A_b and v tensors, yielding coarse-grained tensors A'_p.

seen in Fig. 20(c). These projective truncations, which involve a projector composed of an isometry w and its conjugate, are detailed in Fig. 21(c)–21(d). In the final step of the iteration, as shown in Fig. 20(d), certain isometries w annihilate to identity with their conjugates, yielding the coarse-grained network. Notice that the tensors $A_b{}'$, $A_r{}'$, and $A_p{}'$ of the coarse-grained network, as defined in Fig. 21, possess the same rotational symmetry as the corresponding initial tensors A_b, A_r, and A_p, and also that the coarse-grained network has the same unit cell (in terms of the coarse-grained tensors) as the initial network, but is reduced by a factor of 2 in linear dimension.

APPENDIX D: REFLECTION SYMMETRY

In this Appendix we describe the details, given a tensor network $\mathcal{G}$ that is symmetric with respect to spatial reflections (perhaps in conjunction with complex conjugation) along one axis, for how the symmetry can be preserved under coarse-graining with TNR. For the case in which $\mathcal{G}$ represents a Euclidean path integral the presence of this symmetry follows from the Hermiticity of the Hamiltonian, and is thus always present, while for the case in which $\mathcal{G}$ represents a partition function the symmetry is present if the underlying $2D$ classical statistical model is invariant under spatial reflection along an axis. Proper exploitation of this symmetry is desirable as it can significantly simplify the TNR algorithm.

We say the homogeneous tensor network $\mathcal{G}$ is Hermitian symmetric with respect to the horizontal axis if a row of tensors, which form a matrix product operator (MPO), is invariant with respect to permutation of top-bottom indices in conjunction with complex conjugation; see also Fig. 22. If a row of tensors satisfies the above definition of reflection symmetry, then it can be shown that permutation of top-bottom indices and complex conjugation of tensor A is equivalent to enacting a unitary gauge change on its horizontal indices,

$$A^\dagger \equiv (A_{ilkj})^* = \sum_{j',l'} \mathbf{x}^*_{j,j'} A_{ijkl} \mathbf{x}_{l',l}; \tag{D1}$$

see also Figs. 22(b) and 22(c). Here the indices of tensor A_{ijkl} are labeled clockwise from the top, as per Fig. 1(a), and $\mathbf{x}$ is some unitary matrix.

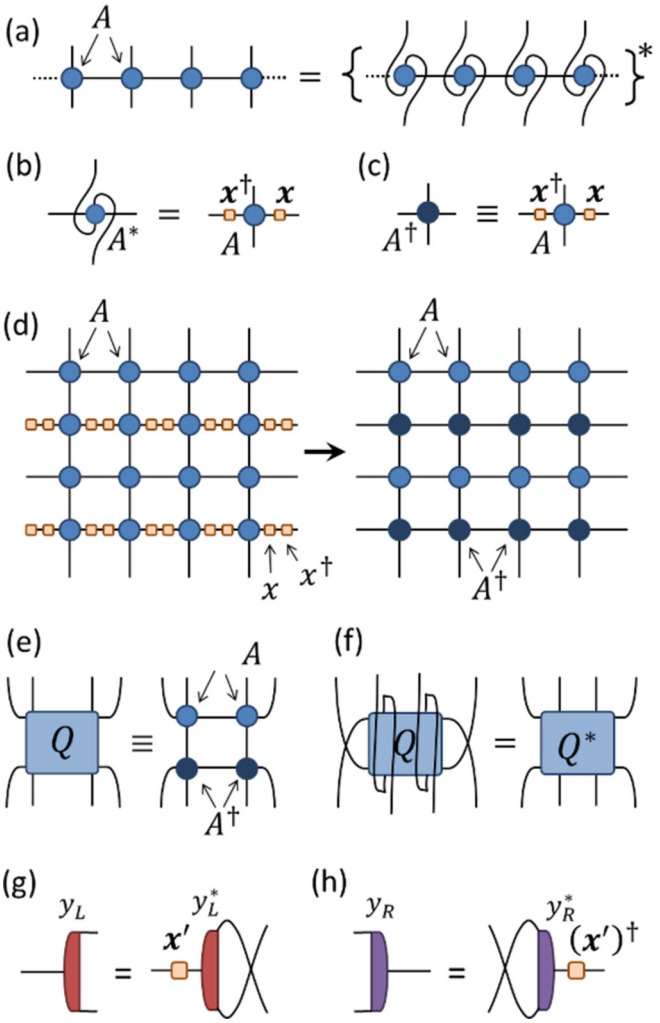

FIG. 22. (a) The tensor A is Hermitian symmetric if a row of such tensors is invariant with respect to permutation of top-bottom indices in conjunction with complex conjugation. (b) Permutation and complex conjugation of Hermitian symmetric A is equivalent to a gauge change, enacted by some unitary matrix $\mathbf{x}$, on its horizontal indices. (c) Definition of $A^\dagger$. (d) A gauge change is performed on the tensors of every second row of the square-lattice network. (e) Definition of tensor Q. (f) Tensor Q is seen to be Hermitian. (g) and (h) Constraints for isometries y_L and y_R necessary such that the tensors of the coarse-grained network remain Hermitian symmetric.

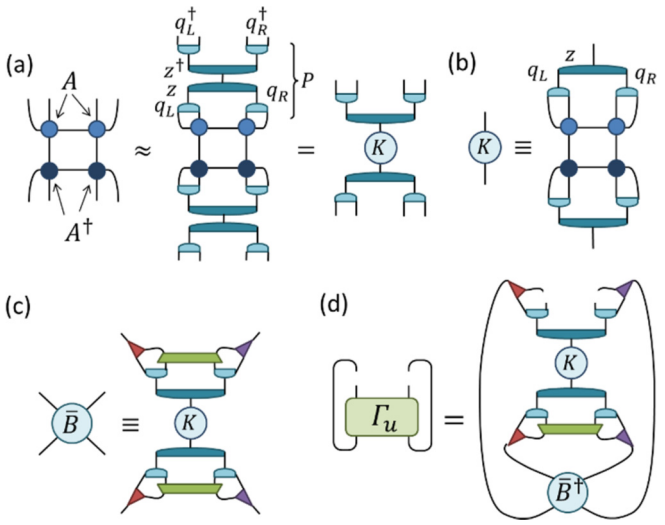

FIG. 23. (a) Details of an optional, preliminary projective truncation for the binary TNR scheme where two copies of a projector P, which is formed from a product isometric tensors q_L, q_R, z and their conjugates, are enacted upon a 2×2 block of A tensors. (b) Definition of tensor K. (c) Definition of tensor $\bar{B}$, which differs from the previous B, see Fig. 6(b), only by an amount related to the truncation error of P in (a). (d) Detail of the environment Γ_u computed from $\bar{B}$. The cost of contracting this network scales as $O(\chi^6)$ in terms of the bond dimension χ, as opposed to $O(\chi^7)$ for the previous environment of Fig. 7(c).

In the first step of the TNR iteration, as discussed in Sec. IV A, it is useful to enact the unitary gauge change $\mathbf{x}$ on every second row of tensors in the network $\mathcal{G}$ as depicted in Fig. 2(b). Let us define tensor Q as the tensor formed from contracting two copies of A and two copies of $A^\dagger$ together as depicted in Fig. 2(d). The reason that the initial gauge transformation on $\mathcal{G}$ is useful is that it allows tensor Q to be Hermitian under exchange of top-bottom indices; thus at the first step of the TNR iteration the same projector P_u may be used both on the top and bottom of Q; see Figs. 6(a)–6(c). The reflection symmetry can be preserved under the TNR iteration, such that the tensors A' of the coarse-grained network satisfy Eq. (D1) for some unitary matrix $\mathbf{x}'$, if the isometries y_L and y_R used in the second step of the TNR iteration satisfy the relation shown in Figs. 22(e) and 22(f). This relation states that the isometries should be invariant under complex conjugation in conjunction with permutation of their incoming indices and a unitary gauge change, enacted by unitary matrix $\mathbf{x}'$, on their outgoing index. Isometries y_L and y_R satisfying this relation can be obtained by symmetrizing the environments Γ_{y_L} and Γ_{y_R} of the isometries during their optimization, similarly to previously strategies for preserving reflection symmetry in MERAs described in Ref. [31].

Finally, we remark these ideas can be extended such that reflection symmetry along the vertical axis can be preserved (simultaneously with that on the horizontal axis), if both symmetries are present in the initial network.

APPENDIX E: REDUCTION OF COMPUTATIONAL COST

The cost of the TNR algorithm as described in Sec. IV scales as $O(\chi^7)$ in terms of the bond dimension χ; in this Appendix here we describe how this cost scaling can be reduced to $O(\chi^6)$. This reduction in cost is achieved by doing an additional projection truncation at the start of each TNR iteration; specifically this projective truncation is enacted on 2×2 blocks of A tensors, with two of the tensors conjugated as discussed in Appendix D, before the projective truncation step of Fig. 6(a)–6(c). The projector P involved in this step is composed of isometries q_l, q_r, and z (and their complex conjugates), as depicted in Fig. 23, and can be optimized with the standard iterative SVD approach as described in Sec. III C.

After this initial projective truncation the cost of the subsequent step of the TNR iteration is reduced. Figure 23(c) depicts the new tensor $\bar{B}$, which differs from the tensor B in

Fig. 6(b) only by an amount related to the truncation error ε of the initial projection step, but can be computed with a cost that scales $O(\chi^6)$, as opposed to $O(\chi^7)$ for computing B. Likewise the environment Γ_u of disentangler u when expressed in terms of $\bar{B}$, see Fig. 23(d), can also be computed with cost $O(\chi^6)$ instead of the cost $O(\chi^7)$ associated with computing the previous environment of Fig. 7(c). Similarly, the environments of isometries v_L and v_R can also be computed with cost $O(\chi^6)$ when using $\bar{B}$. Thus, when using the results of this Appendix, no operation required to implement the binary TNR scheme has a cost scaling of greater than $O(\chi^6)$.

APPENDIX F: OPTIMIZATION USING A LARGER ENVIRONMENT

In this Appendix we discuss how the accuracy of the binary TNR scheme, for given bond dimension χ, can be improved by taking a larger environment into account at each truncation step. This follows similar ideas introduced in Ref. [7] to improve TRG by taking the local environment into consideration at each truncation step.

The TNR approach is based on the use *projective truncations* to implement coarse-graining transformations, as discussed in Sec. III C. A projective truncation involves application of a projector P to a local subnetwork of tensors $\mathcal{F}$, where it is desired that P acts on $\mathcal{F}$ as an approximate resolution of the identity; see Eq. (19). Use of a larger subnetwork $\mathcal{F}$ typically allows a more accurate truncation, as the projector P can take into account correlations from a larger region of the network. Figure 24(a) depicts a larger subnetwork, consisting of two copies of the B tensor in addition to v_L and v_R tensors, that can be used in the determination of the projectors P_L, P_R, and P_w in the second and third step of the TNR iteration. The condition that projectors P_L, P_R, and P_w act with small truncation error on this subnetwork, as shown in Fig. 24(a), is less restrictive than the condition previously imposed on the projectors in Sec. IV (which used a smaller subnetwork), thus potentially allowing for more accurate projectors to be chosen. The isometric tensors $\{y_L, y_R, w\}$ that compose these projectors can be chosen to minimize the truncation error ε by optimizing them to maximize $\|\tilde{A}\|^2$, with tensor $\tilde{A}$ as defined in Fig. 24(b), which effectively replaces the two separate optimizations depicted previously in Figs. 6(d) and 6(f).

We would also like to use the larger subnetwork in the optimization of the disentangler u. Given that the B tensors depend on the disentanglers u, as depicted in Fig. 6(b), one could likewise optimize u to maximize $\|\tilde{A}\|^2$. However, the dependance of $\tilde{A}$ on disentanglers u is not in a form that is directly compatible with the optimization problem discussed in Sec. III C. To this end, we use a modified environment $\tilde{\Gamma}_u$ of u generated from $\|\tilde{A}\|^2$, as depicted in Fig. 24(c), where the modified environment results from the removal of either a pair tensors, $v_L v_L^\dagger$ or $v_R v_R^\dagger$, from $\tilde{A}$. The modified environment $\tilde{\Gamma}_u$ is now compatible with the previous optimization strategy of iterative SVD updates, and thus can be used to directly replace the previous environment Γ_u of u from Fig. 7(c). Note that $\tilde{\Gamma}_u$ can be computed with cost that scales as $O(\chi^7)$ in terms of bond dimension χ [or $O(\chi^6)$ when employing the ideas of Appendix E], which is the same scaling with χ as the basic TNR algorithm discussed in Sec. IV.

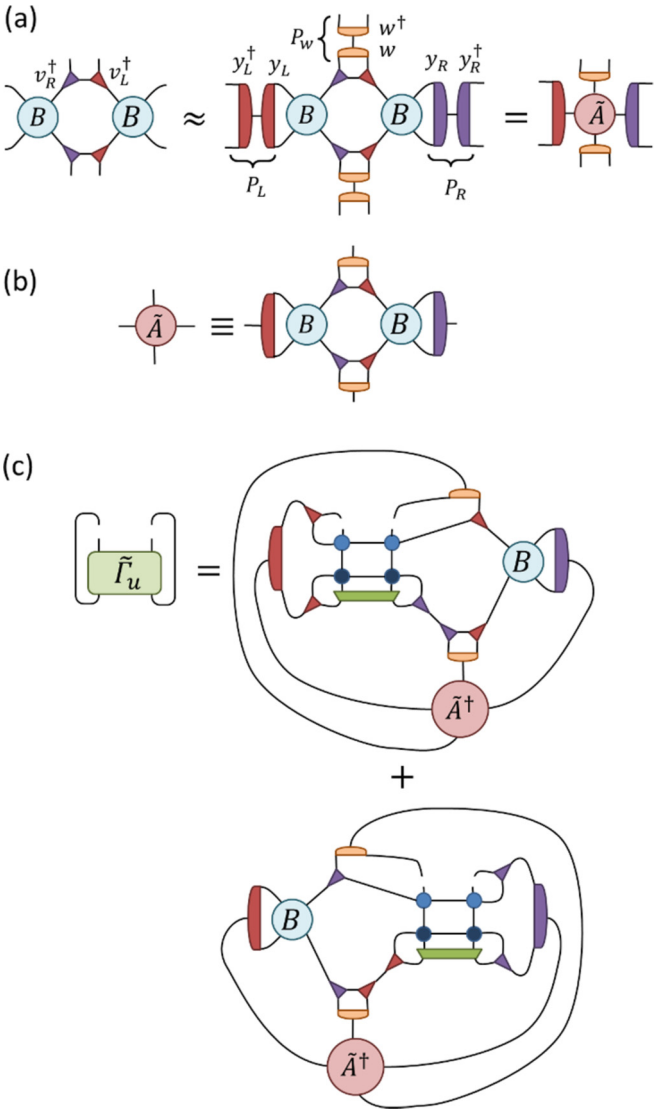

FIG. 24. (a) Isometric tensors y_L, y_R, w yielding a more accurate coarse-graining transformation can found by optimizing them to minimize the truncation error when applied to a region of the network, here containing two copies of the B tensor, that is larger than was previously considered in Figs. 6(d) and 6(f). Notice that minimizing the truncation error is equivalent to maximizing the norm of $\tilde{A}$. (b) Definition of tensor $\tilde{A}$. (c) The two contributions to the modified environment $\tilde{\Gamma}_u$ for disentangler u, as generated from $\|\tilde{A}\|^2$.

APPENDIX G: ACHIEVING A SCALE-INVARIANT RG FLOW

A novel and useful feature of the TNR approach is that, when applied to a scale-invariant critical system, it can generate an explicitly scale-invariant RG flow, such that tensors at different scales s of coarse-graining are equal,

$$A^{(s+1)} \approx A^{(s)}, \tag{G1}$$

up to small differences stemming from truncation errors. However, realization of scale invariance requires fixing the gauge degree of freedom, which we now discuss.

Given a square lattice network $\mathcal{G}$ composed of identical tensors A_{ijkl} there exists a local gauge freedom in the network relating to a local change of basis on individual indices of the tensor, implemented by unitary matrices $\mathbf{x}$ and $\mathbf{y}$,

$$A_{ijkl} \to \sum_{i'j'k'l'} (A)_{i'j'k'l'} \mathbf{x}_{ii'} (\mathbf{x}^\dagger)_{k'k} \mathbf{y}_{ll'} (\mathbf{y}^\dagger)_{j'j}, \qquad \text{(G2)}$$

under which the tensor network remains unchanged. (Note that in general the tensor network is invariant under changes of gauge implemented by invertible matrices; however the more restrictive class of unitary changes of gauge is sufficient to consider whether reflection symmetry is exploited; see Appendix D). If the gauge degree of freedom is not given proper consideration, application of TNR to a scale-invariant critical system will not, in general, produce an *explicitly* scale-invariant RG flow, as defined by Eq. (G1). Instead an *implicitly* scale-invariant RG flow may be given, where the tensors $A^{(s+1)}$ and $A^{(s)}$ differ by choice of gauge, even though they are representative of the same critical fixed point. (Conversely, it can be demonstrated that TRG does not generate an implicitly scale-invariant RG flow when applied to a scale-invariant system, as certain gauge-invariant properties of tensors $A^{(s)}$ diverge with RG step s; see Ref. [1]). We now explain how the gauge freedom in TNR can be fixed such that an otherwise implicitly scale-invariant RG flow becomes explicitly scale-invariant.

There are many strategies one could employ to fix the choice of gauge on tensor $A^{(s+1)}$ to be compatible with that on the previous tensor $A^{(s)}$. One possibility is to include a separate gauge-fixing step after each coarse-graining iteration that minimizes the difference $\|A^{(s+1)} - A^{(s)}\|$ through optimization of unitary matrices $\mathbf{x}$ and $\mathbf{y}$ that implement a change of gauge on $A^{(s+1)}$, as per Eq. (G2). A different possibility, one that we find more convenient, is to include the gauge-fixing as part of the tensor optimization described in Sec. IV B. Let $B^{(s)}$ be the tensor obtained after the first step of the TNR iteration on $A^{(s)}$, as per Fig. 6(b), and assume we wish to choose a gauge on isometries $v_L^{(s)}$, $v_R^{(s)}$ and disentangler $u^{(s)}$ consistent with the previous TNR iteration, i.e., such that the choice of gauge on $B^{(s)}$ takes it as close to possible $B^{(s-1)}$. Let us define

$$\tilde{B} = B^{(s)} + \delta B^{(s-1)}, \qquad \text{(G3)}$$

for some $\delta > 0$. Then, during the optimization of $v_L^{(s)}$, $v_R^{(s)}$, and $u^{(s)}$, if $\tilde{B}$ is used instead of $B^{(s)}$ in the calculation of the tensor environments, Figs. 7(a)–7(c), the optimization is biased towards ensuring that the tensors are chosen in the same gauge as those at the previous TNR iteration. Typically we take $\delta = 1$ for the early stages of the optimization of tensors $v_L^{(s)}$, $v_R^{(s)}$, and $u^{(s)}$, but reduce δ smaller as the tensors converge. The same strategy can then be employed during the optimization of tensors $y_L^{(s)}$, $y_R^{(s)}$, and $w^{(s)}$ in the other intermediate steps of the TNR iteration to ensure that the gauge on $A^{(s+1)}$ is fixed in a way compatible with $A^{(s)}$.

[1] G. Evenbly and G. Vidal, Phys. Rev. Lett. **115**, 180405 (2015).
[2] M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007).
[3] H. C. Jiang, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **101**, 090603 (2008).
[4] Z.-C. Gu, M. Levin, and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008).
[5] Z.-Y. Xie, H.-C. Jiang, Q.-N. Chen, Z.-Y. Weng, and T. Xiang, Phys. Rev. Lett. **103**, 160601 (2009).
[6] Z.-C. Gu and X.-G. Wen, Phys. Rev. B **80**, 155131 (2009).
[7] H.-H. Zhao, Z.-Y. Xie, Q.-N. Chen, Z.-C. Wei, J. W. Cai, and T. Xiang, Phys. Rev. B **81**, 174411 (2010).
[8] Z.-Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang, Phys. Rev. B **86**, 045139 (2012).
[9] B. Dittrich, F. C. Eckert, and M. Martin-Benito, New J. Phys. **14**, 035008 (2012).
[10] A. Garcia-Saez and J. I. Latorre, Phys. Rev. B **87**, 085130 (2013).
[11] For a review of the renormalization group, see M. E. Fisher, Rev. Mod. Phys. **70**, 653 (1998).
[12] P. Di Francesco, P. Mathieu, and D. Senechal, *Conformal Field Theory* (Springer-Verlag, New York, 1997).
[13] M. Henkel, *Conformal Invariance and Critical Phenomena* (Springer-Verlag, Berlin, Heidelberg, 1999).
[14] K. G. Wilson, Phys. Rev. B **4**, 3174 (1971); **4**, 3184 (1971); Rev. Mod. Phys. **47**, 773 (1975).
[15] G. Evenbly and G. Vidal, Phys. Rev. Lett. **116**, 040401 (2016).
[16] G. Vidal, Phys. Rev. Lett. **99**, 220405 (2007).
[17] G. Vidal, Phys. Rev. Lett. **101**, 110501 (2008).
[18] G. Evenbly and G. Vidal, Phys. Rev. Lett. **115**, 200401 (2015).
[19] G. Evenbly and G. Vidal, Phys. Rev. B **79**, 144108 (2009).
[20] V. Giovannetti, S. Montangero, and R. Fazio, Phys. Rev. Lett. **101**, 180503 (2008).
[21] R. N. C. Pfeifer, G. Evenbly, and G. Vidal, Phys. Rev. A **79**, 040301(R) (2009).
[22] G. Evenbly, P. Corboz, and G. Vidal, Rev. A **81**, 010303(R) (2010).
[23] G. Evenbly and G. Vidal, in *Strongly Correlated Systems: Numerical Methods*, edited by A. Avella and F. Mancini, Springer Series in Solid-State Sciences, Vol. 176 (Springer-Verlag, Berlin, Heidelberg, 2013), Chap. 4.
[24] J. C. Bridgeman, A. O'Brien, S. D. Bartlett, and A. C. Doherty, Phys. Rev. B **91**, 165129 (2015).
[25] M. Suzuki, Phys. Lett. A **146**, 319 (1990); J. Math. Phys. **32**, 400 (1991).
[26] A. T. Sornborger and E. D. Stewart, Phys. Rev. A **60**, 1956 (1999).
[27] L. de Lathauwer, B. de Moor, and J. Vandewalle, SIAM J. Matrix Anal. Appl. **21**, 1253 (2000).
[28] G. Evenbly, R. N. C. Pfeifer, V. Pico, S. Iblisdir, L. Tagliacozzo, I. P. McCulloch, and G. Vidal, Phys. Rev. B **82**, 161107(R) (2010).

[29] P. Silvi, V. Giovannetti, P. Calabrese, G. E. Santoro, and R. Fazio, J. Stat. Mech. (2010) L03001.
[30] G. Evenbly and G. Vidal, Phys. Rev. B **91**, 205119 (2015).
[31] G. Evenbly and G. Vidal, J. Stat. Phys. **157**, 931 (2014).
[32] Y.-L. Lo, Y.-D. Hsieh, C.-Y. Hou, P. Chen, and Y.-J. Kao, Phys. Rev. B **90**, 235124 (2014).
[33] M. Hauru, G. Evenbly, W. W. Ho, D. Gaiotto, and G. Vidal, Phys. Rev. B **94**, 115125 (2016).
[34] M. Fannes, B. Nachtergaele, and R. F. Werner, Commun. Math. Phys. **144**, 443 (1992).
[35] S. Ostlund and S. Rommer, Phys. Rev. Lett. **75**, 3537 (1995).
[36] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).
[37] G. Vidal, Phys. Rev. Lett. **93**, 040502 (2004).
[38] S. Singh, R. N. C. Pfeifer, and G. Vidal, Phys. Rev. A **82**, 050301 (2010).
[39] L. Cincio, J. Dziarmaga, and M. M. Rams, Phys. Rev. Lett. **100**, 240603 (2008).
[40] G. Evenbly and G. Vidal, Phys. Rev. Lett. **102**, 180406 (2009).
[41] F. Verstraete and J. I. Cirac, arXiv:cond-mat/0407066.
[42] F. Verstraete, J. I. Cirac, and V. Murg, Adv. Phys. **57**, 143 (2008).
[43] J. Jordan, R. Orus, G. Vidal, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **101**, 250602 (2008).