We discuss in detail algorithms for implementing tensor network renormalization (TNR) for the study of classical statistical and quantum many-body systems. Firstly, 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.

PACS numbers: 05.30.-d, 02.70.-c, 03.67.Mn, 75.10.Jm

CONTENTS

I. Introduction

II. Tensor network representations of many-body systems
   A. Classical many-body systems
   B. Quantum many-body systems

III. Coarse-graining tensor networks
   A. Local approximations
   B. Truncated singular value decomposition
   C. Projective truncations

IV. Tensor network renormalization
   A. Coarse-graining step of the binary TNR scheme
   B. Optimization of tensors
   C. RG flow of tensors
   D. Algorithmic details

V. Calculation of observables

VI. Benchmark results
   A. 2D classical Ising model
   B. 1D quantum Ising model

VII. Discussion
   A. Coarse-graining along a single dimension
   B. Ternary TNR scheme
   C. Isotropic TNR scheme
   D. Reflection symmetry
   E. Reduction of computational cost
   F. Optimization using a larger environment
   G. Achieving a scale-invariant RG flow

References

I. INTRODUCTION

Tensor network renormalization (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 (TRG) as well as other subsequently developed approaches, 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 prevent this accumulation, such that TNR can maintain an accurate 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 (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 (TEFR) in Ref. which reproduces the proper structure of gapped RG fixed points. On the other hand, TNR fully resolves these problems, reproducing the proper structure of gapped fixed points as well as producing scale-invariant fixed points when applied to critical systems corresponding to discrete ver-

References
sions of conformal field theories (CFT), thus correctly realizing Wilsons RG ideas on tensor networks. By capturing scale-invariance, and producing a rescaling transformation for the lattice consistent with conformal transformations of the field theory, 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 desentanglers 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 desentanglers in entanglement renormalization (ER) and in the multi-scale entanglement renormalization ansatz (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 based upon variational energy minimization for obtaining optimized MERA, and also allows methods developed for extracting scale-invariant data from quantum critical systems using MERA to be generalized to classical statistical system,

\[
Z = \sum_{\{\sigma\}} e^{-H(\{\sigma\})/T},
\]

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

\[
H(\{\sigma\}) = -\sum_{\langle i,j \rangle} \sigma_i \sigma_j,
\]

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 Ising spin interactions on the edges of the plaquette on which it sits,

\[
A_{ijkl} = e^{(\sigma_i \sigma_j + \sigma_k \sigma_l + \sigma_i \sigma_l + \sigma_i \sigma_k)} / T,
\]

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

\[
Z = \sum_{ijk\ldots} A_{ijkl} A_{mnop} A_{qrst} A_{opqr} \ldots
\]

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.

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 many-body system can each be expressed as a square lattice network, which is the starting point for 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,
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. We assume, for simplicity, that Hamiltonian $H$ is a sum of identical nearest-neighbor terms $h_r$,

$$H = \sum_r h_{r,r+1}. \quad (5)$$

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

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

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

$$e^{-\tau H} \approx e^{-\tau H_{\text{odd}}} e^{-\tau H_{\text{even}}} \quad (7)$$

where $H_{\text{odd}}$ and $H_{\text{even}}$ represent the contribution to $H$ given from sites $r$ odd or $r$ even respectively, and an error of order $O(\tau^n)$ has been introduced. [Note that one can obtain an error $O(\tau^n)$, $n > 1$, by using a higher order Suzuki-Trotter decomposition]. Since $H_{\text{odd}}$ 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 $e^{-\tau H_{\text{even}}}$,

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

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

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 $\tau$.

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 $\tau$, 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 $e^{-\tau h}$,

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

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 $e^{-\tau h}$,

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

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 $\tau$]. 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 Fig.2(b-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 similar to the HOTRG 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 $G$ 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 $\langle G \rangle$, associated to network. As an exact contraction of the network $G$ 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.
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 $A$.

A. Local approximations

Let $\mathcal{F}$ denote a sub-network of tensors, for example a $2 \times 2$ 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 sub-network $\mathcal{F}$ with a different network of tensors $\tilde{\mathcal{F}}$ if they differ by a small amount $\varepsilon$, 

$$\varepsilon \equiv \|[\mathcal{F} - \tilde{\mathcal{F}}]\|,$$

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

$$\|A\| = \sqrt{\text{tr}(A \otimes A^\dagger)},$$

where ‘tr’ 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 manuscript, TNR, employ a pattern 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 $2 \times 2$ 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 $\varepsilon$ in Eq.8 could be viable as part of a coarse-graining scheme. In the original TRG algo-
rithm proposed by Levin and Nave, and in many of the
generalizations and improvements to TRG, the local
approximations that are used are based upon a truncated
singular value decomposition (SVD) of single tensors [or
a generalized form of the SVD known as the higher-order
singular value decomposition (HOSVD)], which we now
discuss.

Consider a four index tensor $A_{ijkl}$ where each index is
of dimension $\chi$. If the tensor is viewed as a $\chi^2 \times \chi^2$
matrix corresponding to the pairing of indices $A_{i[j][k]l}$ then
the SVD can be performed,

$$A_{ijkl} = \sum_m u_{ijm} s_{mm} v_{mkl},$$

(13)

where $u$ and $v$ are unitary according to grouping of in-
dices $u_{ij[m]}$ and $v_{[m][kl]}$ respectively, and $s$ is a positive
diagonal matrix of singular values, i.e. $s_{mm} = \delta_{nm} \lambda_m$,
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 approxi-
mate,

$$A_{ijkl} \approx \sum_m \chi' u_{ijm} s_{mm} v_{mkl},$$

(14)

where the truncation error $\varepsilon$, as defined in Eq.11 is seen
to equal to the square-root of the sum of the squares of the
discarded singular values,

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

(15)

Here we have assumed that the tensor $A$ was normal-
ized, $\|A\| = 1$, 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 ten-
sor $A$ into the product of a pair of tensors (connected by
an index of some rank $\chi'$), 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 lo-
cal approximation than those based upon the SVD, which
we term projective truncations. In a projective trunca-
tion, a local sub-network $F$ is replaced by a new local
network $\tilde{F}$ that consists of a projector $P$, which satisfies
$PP^\dagger = P^2 = P$, acting on some or all of the open indices
of $F$,

$$\tilde{F} = FP = Fw w^\dagger,$$

(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, $P = w w^\dagger$, where the isometry satis-
ifies $w^\dagger w = 1$, see also Fig.4(b-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.4(c)].
Projective truncations are a particularly useful class of
local approximation, as the figure of merits to optimize
projector $P$ takes a very simple form. To see this, we
first expand the terms in Eq.11

$$\varepsilon^2 = \|F\|^2 + \|\tilde{F}\|^2 - t \text{Tr} (F \otimes \tilde{F}^* ) - t \text{Tr} (\tilde{F} \otimes F^* ).$$

(17)

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

$$\|\tilde{F}\| = t \text{Tr} (F \otimes \tilde{F}^* ) = t \text{Tr} (\tilde{F} \otimes F^* ).$$

(18)

It follows that the expression for the replacement error $\varepsilon$ can be simplified,

$$\varepsilon = \sqrt{\|F\|^2 - \|FP\|^2}
= \sqrt{\|F\|^2 - \|Fw\|^2},$$

(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 pro-
jector $P$ such that the error $\varepsilon$ of Eq.19 is minimized.
For simplicity we consider the case where the projector $P$
composes as a product of a single isometry $w$ and
its conjugate, $P = w w^\dagger$, 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 $\varepsilon$ of Eq.19 is minimized when $\|Fw\|$ is
maximized, which follows as $\|Fw\| \leq \|F\|$; we now dis-
cuss an iterative strategy for optimizing isometry $w$ to
maximize the expression $\|Fw\|$.

The strategy we employ is based upon linearization of
the cost function. Given that the expression $\|Fw\|^2$ has
quadratic dependence on isometry $w$ [or, more specifi-
cally, in depends on both $w$ and $w^\dagger$], we simplify the
problem by temporarily holding $w^\dagger$ fixed and then solv-
ing 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 ten-
sors in a MERA as described in Ref.19, to which we refer
the interested reader for more details. We begin by ex-
pressing the closed network $\|Fw\|^2$ in a factorized form,

$$\|Fw\|^2 = t \text{Tr} (\Gamma_w \otimes w ),$$

(20)

where $\Gamma_w$, referred to as the environment of $w$, represents
the contraction of everything in $\|Fw\|^2$ excluding tensor
$w$, see also Fig.4(e-f). The singular value decomposition
of the environment $\Gamma_w$, when considered as a matrix ac-
cording to the same partition of indices for which tensor
$w$ is isometric, is then taken,

$$\Gamma_w = u v^\dagger,$$

(21)
FIG. 4. (a) In a projective truncation a sub-network $F$ is replaced by a new sub-network $\tilde{F}$, which consists of a projector $P$ applied to the original sub-network, i.e. $\tilde{F} = FP$. (b) Here we assume that $P$ is decomposed as a product of an isometric tensor $w$ and its conjugate, $P = w w^\dagger$. (c) By definition, isometry $w$ contracts to identity with its conjugate, $w w^\dagger = I$. (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. 13. (e) The environment $\Gamma_w$ of isometry $w$ is defined as the network that results by removing a single instance of $w$ from $\|Fw\|^2$, see also Eq. 20. (f) By construction, the contraction of $w$ and its environment $\Gamma_w$ is equal to $\|Fw\|^2$. (g) Environment $\Gamma_w$ is decomposed, via singular value decomposition (SVD), into a product of isometric tensors $u$, $v$ and diagonal matrix $s$.

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

$$w = vu^\dagger,$$

(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 $\Gamma_w$ was held fixed, these steps of (i) computing the environment $\Gamma_w$ and (ii) obtaining an updated $w$ through the SVD of the environment $\Gamma_w$ 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. For any given lattice geometry there are many potential TNR schemes that fulfill this requirement. In this manuscript 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 [2] 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 [3] 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 $G$ composed of four index tensors $A_{ijkl}$, where indices are assumed to be of some dimension $\chi$. As discussed in Sect.[4] 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 $G$ 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 non-homogeneous networks [special examples, including
networks with an open boundary or a defect line, can be handled using similar methods as those developed in the context of MERA\textsuperscript{28–32}, and are discussed separately in Ref\textsuperscript{33}. We also assume that $G$ 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. In the case that $G$ represents a Euclidean path integral of a quantum Hamiltonian $H$, the presence of this symmetry follows from $H$ being Hermitian, while in the case that $G$ 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 $G$ to the coarser network $G'$ 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 $G$, as discussed in Appendix D. Here the gauge change is chosen such that it is equivalent to flipping top-bottom 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 6 depicts the remaining steps in transforming network $G$ into the coarser network $G'$. In Fig 6(a), a projective truncation is enacted on $2 \times 2$ blocks of tensors $A$ [where two of the tensors are 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 $\chi$-dimensional complex vector space $V_\chi$, they describe a mapping between vector spaces,

$$u : V_\chi \otimes V_\chi \rightarrow V_\chi \otimes V_\chi.$$  \hspace{1cm} (23)

By virtue of being unitary, the disentanglers satisfy $u^\dagger u = \mathbb{1} \otimes \mathbb{1}$ where $\mathbb{1}$ is the identity operator on $V_\chi$. Conceptually, the role of disentanglers is to remove short-range correlations that would otherwise be missed, as discussed in greater detail in Ref\textsuperscript{28}, 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 $\chi' \leq \chi^2$,

$$v_L : \mathcal{V}_{\chi'} \rightarrow \mathcal{V}_{\chi'} \otimes \mathcal{V}_{\chi'}, \quad v_R : \mathcal{V}_{\chi'} \rightarrow \mathcal{V}_{\chi'} \otimes \mathcal{V}_{\chi'},$$  \hspace{1cm} (24)

where the new index has been regarded as hosting a $\chi'$-dimensional complex vector space $\mathcal{V}_{\chi'}$. By definition, isometries satisfy $v_L^\dagger v_L = v_R^\dagger v_R = I'$, with $I'$ the identity operator on $\mathcal{V}_{\chi'}$. After the coarse-graining step of Fig 6(a), it is useful to define a new vector index $B$, which is defined from 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 $2 \times 2$ blocks of tensors, pairs of disentanglers from neighboring blocks can annihilate to identity, see Fig 6(b), leaving a network of $B$ tensors interspersed with groups of isometries $v_L$ and $v_R$. Next, as depicted in Fig 6(b) and further detailed in Fig 6(d-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 $B$ tensor respectively, each formed as a product of isometries, $P_L = y_L y_L^\dagger$ and $P_R = y_R y_R^\dagger$. Isometries $y_L$ and $y_R$, which satisfy $y_L^\dagger y_L = y_R^\dagger y_R = I'$, each map two indices to a single index also assumed to be of dimension $\chi'$,

$$y_L : \mathcal{V}_{\chi'} \rightarrow \mathcal{V}_{\chi'} \otimes \mathcal{V}_{\chi'}, \quad y_R : \mathcal{V}_{\chi'} \rightarrow \mathcal{V}_{\chi'} \otimes \mathcal{V}_{\chi'}.$$  \hspace{1cm} (25)
where the isometry satisfies \( w \) is made using projectorors a final projective truncation, as shown Fig.5(d) and This final step yields a coarse-grained square lattice identity with their conjugates, as depicted in Fig.5(e). of isometries \( \chi \) A of four index tensors \( \{ t \) obtained throughout the coarse-graining step, namely from \( \text{Dim} \), the second projective truncation of the TNR iteration, as detailed in Fig.6(g). We then define matrix \( D \) from enacting isometries \( y_L \) and \( y_R \) on tensor \( B \), as shown Fig6(c). 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 Fig6(d) and further detailed in Fig6(f-g), is made using projector\( P_w = w w^\dagger \) with isometry \( w \) mapping two \( \chi \)-dimensional indices to a single index of dimension \( \chi' \),

\[
\chi' = \sqrt{\chi^2},
\]

where the isometry satisfies \( w^\dagger w = 1 \). 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 Fig5(e). This final step yields a coarse-grained square lattice \( G' \) of four index tensors \( A' \), whose indices are of some specified dimension \( \chi' \leq \chi^2 \). 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 Fig6(g).

We now discuss in more detail.

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 \( \Gamma_{v_L}, \Gamma_{v_R} \) and \( \Gamma_u \) of the isometries \( v_L, v_R \) and disentangler \( u \) involved in the first projective truncation of the TNR iteration, as detailed in Fig6(a). (d-e) Environments \( \Gamma_{y_L} \) and \( \Gamma_{y_R} \) of isometries \( y_L \) and \( y_R \) from the second projective truncation of the TNR iteration, as detailed in Fig6(d). (f) Environment \( \Gamma_w \) of isometry \( w \) from the third projective truncation of the TNR iteration, as detailed in Fig6(f).

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 Sect III C as we now discuss in more detail.

The first step of the TNR iteration, as depicted Fig5(a), requires optimization of a projector \( P_u \) composed of isometries \( v_L \) and \( v_R \) and disentangler \( u \), as shown Fig5(c). To update one of these tensors one first computes its environment, where the environments \( \Gamma_{v_L}, \Gamma_{v_R}, \Gamma_u \) are shown in Fig7(a-c), then updates the tensor from the SVD of the environment as discussed in Sect 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 \( \chi \)-dimensional.

The second projective truncation step of the TNR iteration, as depicted Fig5(c), requires the optimization of isometries \( y_L \) and \( y_R \), which again are optimized through alternating, iterative SVD updates based upon calculation of their environments \( \Gamma_{y_L} \) and \( \Gamma_{y_R} \), as shown in Fig7(d,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 B. The computational cost of computing each of the environments is \( O(\chi^5) \) assuming the indices involved in the network are \( \chi \)-dimensional.

The third and final projective truncation of the TNR iteration, as depicted Fig5(d), requires optimization of isometry \( w \) which can be achieved through iterative SVD updates of the environment \( \Gamma_w \), depicted in Fig7(f). The computational cost of computing the environment \( \Gamma_w \) is \( O(\chi^8) \) assuming the indices involved in the network are \( \chi \)-dimensional.

C. RG flow of tensors

The single iteration of the TNR approach described in Sect III A which mapped the initial tensor network \( G \) to the coarser network \( G' \), can be iterated many times to generate a sequence of increasing coarse-grained networks,

\[
G^{(0)} \rightarrow G^{(1)} \rightarrow G^{(2)} \rightarrow \cdots \rightarrow G^{(s)} \rightarrow \cdots
\]

with \( G^{(0)} = G \) now as the initial network and network \( G^{(s)} \) as tensor network after \( s \) iterations of TNR, where the linear dimension of the lattice \( G^{(s)} \) has been reduced by a factor of 2 compared to the previous network \( G^{(s-1)} \). Each network \( G^{(s)} \) consists of copies of a four index tensor \( A_i^{(s)}_{ijkl} \), 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 $A_{ijkl}^{(0)}$, 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 Sect. 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,

$$A^{(s)} \rightarrow A^{(s+1)},$$

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

$$A^{(0)} \rightarrow A^{(1)} \rightarrow A^{(2)} \rightarrow \ldots \rightarrow A^{(s)} \rightarrow \ldots .$$

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 $\{v^{(s)}, v^{(s)}_L, v^{(s)}_R, y^{(s)}_L, y^{(s)}_R, w^{(s)}\}$ involved in the coarse-graining, as further discussed in Sect. IV D.

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 $\chi_w$ and $\chi_y$ denote the dimensions of vertical and horizontal indices on $A$ tensors respectively, while $\chi_u$ and $\chi_v$ 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 $\varepsilon$ given at different intermediate steps become roughly equal: in practice such dimensions 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 Sect. IV E. We label results of a TNR simulation by a single dimension $\chi$, 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 $2 \times 2$ blocks of $A$ tensors at the start of the TNR iteration, as described in Appendix F. This additional step reduces the cost of computing the environments $\Gamma_v L$, $\Gamma_v R$ and $\Gamma_u$, see Fig. 7(a-c), from $O(\chi^7)$ to $O(\chi^6)$, thus also reduces 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 $\chi$, 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 and 7 for improving standard TRG by taking the environment into consideration. Appendix E 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, similar to how SRG modifies the TRG to take account of the full environment. This modification, which we shall not detail in the present manuscript, requires sweeping the optimization back and forth over different scales of coarse-graining, functioning similarly to the energy minimization algorithm for optimizing a MERA.

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. In this manuscript we describe a different approach based upon directly coarse-graining the network with the addition of an impurity tensor.

Let $G^{(0)}$ 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 $G^{(7)}$ is a $2 \times 2$ lattice of tensors $A^{(7)}$ that can be exactly contacted. In order to evalu-
ate the expectation value of a local observable, we must now evaluate the tensor network $G^{(0)}$ with the addition of an impurity tensor representing the local observable under consideration. Here we consider an impurity tensor $M^{(0)}$ that replaces a $2 \times 2$ block of $A$ tensors from $G^{(0)}$. To evaluate the impurity network we use the same projective truncations as was used to coarse-grain the homogeneous network $G^{(0)}$ 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 Fig.8(a-e). After an iteration of TNR to the impurity network, one obtains a new impurity network, equal to the homogeneous network $G^{(1)}$ except where a new impurity $M^{(1)}$ replaces a $2 \times 2$ 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(b). The coarse-graining transformation can be applied to the impurity network multiple times as to generate a sequence of impurity tensors,

$$M^{(0)} \rightarrow M^{(1)} \rightarrow \cdots \rightarrow M^{(T)},$$

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

$$M^{(s+1)} = R \left( M^{(s)} \right),$$

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 $2 \times 2$ block of tensors $A^{(T)}$ that would otherwise have been in the homogeneous network $G^{(T)}$, from which the expectation value is evaluated through the appropriate trace of $M^{(T)}$ as depicted in Fig.8(h).

Note that the evaluation of two-point correlators, which corresponds to contracting a network with two local impurities, can be handled in a 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\cite{12,13}. This mapping using TNR was originally formulated in Ref.15, to which we refer the interested reader for more details. A brief summary is included here for completeness. In Fig.8(a) we consider a finite square lattice network that has a $2 \times 2$ 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 Fig.8(a-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

![FIG. 8. (a-e) An iteration of the TNR coarse-graining transformation for a square-lattice tensor network that is homogeneous everywhere except for a $2 \times 2$ region that has been replaced with an impurity tensor $M^{(0)}$. The projective truncations of the TNR iteration are applied as usual, see Fig.8(b) neglecting those which are incompatible impurity tensor. (f) The coarse-grained square lattice is homogeneous everywhere except for a $2 \times 2$ region occupied by new impurity tensor $M^{(1)}$. (g) The coarse-grained impurity tensor $M^{(s+1)}$ is given by enacting ascending superoperator $R$, 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 $\{G_R, G_L, G_U, G_Y\}$.

![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.](image)
superoperator $R$ 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 it 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 $\chi$, thus it may be necessary to use an approximate method for this evaluation. The approximate method we employ in this manuscript is based upon contracting the network layer by layer from top-to-bottom, where we approximate the boundary state as a matrix product state (MPS) and use the TEBD algorithm\(^3\) to apply each layer of gates from 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 2D classical Ising model, as defined in Eqs. 1 and 2. We begin by encoding the partition which corresponds to having one tensor $A$ for every two spins and a tensor network with a 45 degree tilt with respect to the spin lattice, see Fig. 10(a-b). For convenience we then contract a $4 \times 4$ square of tensors $A$ to form a new tensor $A^{(0)}$ of bond dimension $\chi = 16$, which serves as the starting point for the 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 $\chi$. Furthermore we optimize tensors using the larger environment as discussed in Appendix E 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 $\sigma_k \rightarrow -\sigma_k$ of all the spins], which are employed using standard methods for incorporating global symmetries in tensor networks.\(^3\) 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 $\chi$. 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.

Fig. 11(a) explores the truncation error $\varepsilon$, 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, $T_c = 2/\log (1 + \sqrt{2}) \approx 2.269$. It is seen that, although increasing the bond dimension $\chi$ of TRG reduces the initial truncation error $\varepsilon$, the error always increases quickly as a function of RG step $s$ regardless of the bond dimension in use, a phenomenon described.
as the break-down of TRG at criticality in the original work by Levin and Nave. In comparison it is seen that TNR avoids such a break-down; the truncation errors \( \epsilon \) remain constant over many RG steps \( s \) when coarse-graining with TNR, beyond a slight increase after the initial RG step.

The relative error in the free energy per site, \( f = -T \log(Z)/N \), at the critical temperature \( T_c \) is compared for TRG and TNR over a range of bond dimensions \( \chi \) in Fig.11b. 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 \( \chi \), with fit \( \delta f \propto \chi^{-0.02} \), while in TNR \( f \) is seen to converge exponentially fast with \( \chi \), 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 \( \chi \), 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 \( c(T) = -T \frac{\partial^2 M}{\partial T^2} \) obtained with TNR for \( \chi = 6 \), 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 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 \( \delta(s) = \|A^{(s)} - A^{(s-1)}\| \) between tensors at successive RG steps [where tensors have been normalized such that \( \|A^{(s)}\| = 1 \)] is displayed in Fig.13. For the larger \( \chi \) 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, effecting shifting the flow away from criticality and thus away from scale invariance. However, use of a larger bond dimension \( \chi \) 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...
B. 1D quantum Ising model

In this section we provide benchmark calculations for the binary TNR algorithm applied to the Euclidean path integral of the 1D quantum Ising model, which has

\[ H_{\text{Is.}} = \sum_{k} (\sigma_{k}^{x} \sigma_{k+1}^{x} + \lambda \sigma_{k}^{z}) \]  

(32)

where \( \sigma^{x} \) and \( \sigma^{z} \) are Pauli matrices, and \( \lambda \) 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 Sect. 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 versus the energy minimization approach of Ref. 19. The ground state of the Ising Hamiltonian \( H_{\text{Is.}} \) at critical magnetic field strength, \( \lambda = 1 \), is represented using a scale-invariant MERA, 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 \( H_{\text{Is.}} \), 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 the 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 and 24 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. 13 are the comparison between the ground energy error \( \delta E \) obtained by

![Graph](image)

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 \( \delta^{(s)} = ||A^{(s)} - A^{(s-1)}|| \), 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 \( s \).

![Graph](image)

FIG. 14. The smallest 101 scaling dimensions of the 2D classical Ising at critical temperature \( T_{c} \), obtained by diagonalizing the ascending superoperator \( R \), 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.
energy minimization scales as $O(\chi^6)$ in terms of bond dimension $\chi$, while the energy minimization scales as $O(\chi^3)$. 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 $\delta E$, 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 $H_{Is}$. at criticality, $\lambda = 1$, is plotted as a function of system size $L$. This was computed using TNR to coarse-grain the Hamiltonian $H_{Is}$, 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 CFT12, demonstrating that, in addition to the ground state, TNR can accurately approximate the low-energy eigenstates of $H_{Is}$. 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 $e^{-\beta H_{Is}}$ for inverse temperature $\beta$, as discussed in Sect.11 which is then used to generate a thermal state MERA as explained in Ref.18. The thermal energy per site as a function of $\beta$ is displayed in Fig.16(a) for several different magnetic field strengths $\lambda$. In the gapped regime, $\lambda > 1$, the thermal energy converges exponentially quickly to the ground energy in the limit of large $\beta$ [or small temperature $T$], while at the gapless critical point, $\lambda = 1$, the thermal energy converges polynomially quickly to the ground energy with $\beta$. The results from $\lambda = 12$ 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 $\beta$, where correlations are seen to decay faster at smaller $\beta$ [or equivalently at higher temperature $T$] due to thermal fluctuations. Again, the results from $\lambda = 12$ 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 net-
works, a self-contained description of the algorithm for employing TNR to study properties of classical and quantum many-body systems was provided.

Benchmark results, provided in Sect. VII, demonstrated some of the advantages of TNR. These include (i) providing a computationally sustainably 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 upon energy minimization. The TNR approach presented in this manuscript can also be used to compute the norm ⟨Ψ|Ψ⟩ of a 2D quantum many-body state encoded in a PEPS, thus it could be incorporated as a key part of an algorithm for simulation of 2D quantum many-body systems using PEPS.

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 in Fig. 17(a), involves a projective truncation that acts upon 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 Sect. 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 Sect. IV A is one of many possible TNR schemes for the square-lattice; here 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. 15, 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 Fig. 18(a-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 $3 \times 3$ 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 $2 \times 2$ 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 $A_b$ 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 sub-lattices, 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 Fig.21(a-d). The initial step, shown in Fig.21(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.21(b), conjugate pairs of disentanglers annihilate to the identity. A pair of projective truncations are used in the third step of the iteration, seen in Fig.21(c). These projective truncations, which involve a projector composed of an isometry $w$ and its conjugate, are detailed in Fig.21(c-d). In the final step of the iteration, as shown in Fig.21(d), certain isometries $w$ annihilate to identity with their conjugates, yielding the coarse-grained network. Notice that the tensors $A_b'$, 

\[ A_{\text{projective truncation involving isometries } v_L, b_p, v_R, p, y_L, y_R, y_L, y_R, w, w' \text{ annihilate to identity with their conjugates, yielding the coarse-grained network.} \]
FIG. 20. Depiction of a single iteration of the isotropic TNR scheme, which maps a square-lattice network, comprised 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 Fig. 21(a-b). (b) Conjugate pairs of disentanglers $u$ annihilate to identity. (c) Two types of projective truncation are made, as detailed in Fig. 21(c-d). (d) Conjugate pairs of isometries $w$ annihilate to identity, yielding a coarse-grained network comprised of tensors $A'_b$, $A'_p$, and $A'_r$.

$A'_r$ 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 $G$ 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. In the case that $G$ represents a Euclidean path integral the presence of this symmetry follows from the Hermicity of the Hamiltonian, and is thus always present, while in the case that $G$ 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 $G$ 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 conjuga-
tion of tensor $A$ is equivalent to enacting a unitary gauge change on its horizontal indices,

$$A^\dagger \equiv (A_{ikj})^* = \sum_{j',k'} x_{ij,j'}^* A_{ikj} x_{k',l}', \quad (D1)$$

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

In the first step of the TNR iteration, as discussed in Sect.IVA it is useful to enact the unitary gauge change $x$ on every second row of tensors in the network $G$ 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 $G$ 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 $P_0$ may be used both on the top and bottom of $Q$, see Fig.2(a-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 $x'$, if the isometries $y_L$ and $y_R$ used in the second step of the TNR iteration satisfy the relation shown in Fig.22(e-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 $x'$, on their outgoing index. Isometries $y_L$ and $y_R$ satisfying this relation can be obtained by symmetrizing the environments $\Gamma_{yL}$ and $\Gamma_{yR}$ of the isometries during their optimization, similar to previously strategies for preserving reflection symmetry in MERA 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 Sect.IV scales as $O(x^7)$ in terms of the bond dimension $x$; in this appendix here we describe how this cost scaling can be reduced to $O(x^6)$. 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 $2 \times 2$ blocks of $A$ tensors, with two of the tensors conjugated as discussed in Appendix D before the projective truncation step of Fig.2(a-c). The projector $P$ involved in this step is composed of isometries $y_L$, $y_R$, and $z$ [and their complex conjugates], as depicted in Fig.23 and can be optimized with the standard iterative SVD approach as described in Sect.III.C.

After this initial projective truncation the cost of the subsequent step of the TNR iteration is reduced. Fig.23(c) depicts the new tensor $\bar{B}$, which differs from the tensor $B$ in Fig.23(b) only by an amount related to the truncation error $\varepsilon$ of the initial projection step, but can be computed with a cost that scales $O(x^6)$, as opposed to $O(x^7)$ for computing $B$. Likewise the environment $\Gamma_u$ of disentangler $u$ when expressed in terms of $B$, see Fig.23(d), can also be computed with cost $O(x^6)$ instead of the cost $O(x^7)$ associated to computing the previous environment of Fig.23(c). Similarly, the environments of isometries $v_L$ and $v_R$ can also be computed with cost $O(x^6)$ 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 $O(x^6)$.
Appendix F: Optimization using a larger environment

In this appendix we discuss how the accuracy of the binary TNR scheme, for given bond dimension $\chi$, 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 upon the use projective truncations to implement coarse-graining transformations, as discussed in Sect. III C. A projective truncation involves application of a projector $P$ to a local sub-network of tensors $F$, where it is desired that $P$ acts on $F$ as an approximate resolution of the identity, see Eq.19. Use of a larger sub-network $F$ typically allows a more accurate truncation, as the projector $P$ can take into account correlations from a larger region of the network. Fig.22(a) depicts a larger sub-network, 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 sub-network, as shown in Fig.22(a), is less restrictive than the condition previously imposed on the projectors in Sect.IV [which used a smaller sub-network], thus potentially allows 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 $\varepsilon$ by optimizing them to maximize $||\tilde{A}||^2$, with tensor $\tilde{A}$ as defined in Fig.23(b), which effectively replaces the two separate optimizations depicted previously in Fig.23(d,f).

We would also like to use the larger sub-network in the optimization of the disentangler $u$. Given that the $B$ tensors depend on the disentanglers $u$, as depicted in Fig.23(b), one could likewise optimize $u$ to maximize $||\tilde{A}||^2$. However, the dependance of $A$ on disentanglers $u$ is not in a form that is directly compatible with the optimization problem discussed in Sect. III C. To this end, we use a modified environment $\tilde{\Gamma}_u$ of $u$ generated from $||\tilde{A}||^2$, as depicted in Fig.23(c), where the modified environments 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, thus can be used to directly replace the previous environment $\Gamma_u$ of $u$ from Fig.23(c). Note that $\tilde{\Gamma}_u$ can be computed with cost that scales as $O(\chi^6)$ in terms of bond dimension $\chi$ [or $O(\chi^6)$ when employing the ideas of Appendix E], which is the same scaling with $\chi$ as the basic TNR algorithm discussed in Sect. IV.

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,

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

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

Give a square lattice network $G$ 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 $x$ and $y$,

$$A_{ijkl} \rightarrow \sum_{i'j'k'l'} (A)_{ij'k'l'} x_i y_j \tilde{A}_{ikl} (x_i^\dagger y_j^\dagger)_{i'j'}, \quad (G2)$$

under which the tensor network remains unchanged. [Note, 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 if 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 will not system will not, in general, produce an explicitly scale-invariant RG flow, as defined by Eq.(G1) Instead an implicitly scale-invariant RG flow.
The two contributions to the modified environment \( \tilde{\Gamma} \), disentangler \( \tilde{A} \) maximizes the norm of \( \tilde{r} \). Notice that minimizing the truncation error is equivalent to choosing 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 tensor, that is larger than was previously considered in Fig.6(d, f). Therefore, 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 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 Sect. 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,

\[
\tilde{B} = B^{(s)} + \delta B^{(s-1)},
\]

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, Fig.6(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 \( \delta \) 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 \( A^{(s)} \).

---

*gevenbly@uci.edu
1 G. Evenbly and G. Vidal, arXiv:1412.0732
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, 1997).
13 M. Henkel, Conformal Invariance and Critical Phenomena
14. K.G. Wilson, Phys. Rev. B 4, 3174 (1971). K.G. Wilson, Phys. Rev. B 4, 3184 (1971). K.G. Wilson, Rev. Mod. Phys. 47, 773 (1975).

15. G. Evenbly, R. Myers, and G. Vidal, *Local scale transformations on the lattice with tensor network renormalization*, in preparation.

16. G. Vidal, Phys. Rev. Lett. 99, 220405 (2007).

17. G. Vidal, Phys. Rev. Lett. 101, 110501 (2008).

18. G. Evenbly and G. Vidal, arXiv:1502.05385.

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, *Quantum Criticality with the Multi-scale Entanglement Renormalization Ansatz*, chapter 4 in the book *Strongly Correlated Systems. Numerical Methods*, edited by A. Avella and F. Mancini (Springer Series in Solid-State Sciences, Vol. 176 2013), arXiv:1109.5334.

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, 6, (1990) 319-323. M. Suzuki, J. Math. Phys. 32, 2 (1991) 400-407.

26. A. T. Sornborger and E. D. Stewart, Phys. Rev. A, 60 (1999) 156-1965.

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 (2014) 157:931-978.

32. Y.-L. Lo, Y.-D. Hsieh, C.-Y. Hou, P. Chen, and Y.-J. Kao, Phys. Rev. B 90, 235124 (2014).

33. G. Evenbly and G. Vidal, in preparation.

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).