

Forward-mode automatic differentiation for the tensor renormalization group and its relation to the impurity method

Yuto Sugimoto^{*}*Department of Physics, Tohoku University, Sendai 980-8578, Japan*

(Received 11 February 2026; accepted 10 April 2026; published 8 May 2026)

We propose a forward-mode automatic differentiation (AD) framework for tensor renormalization group methods. In this approach, evaluating the derivatives of the partition function up to the order of k increases the matrix-multiplication cost by a factor of $(k+1)(k+2)/2$ compared to computing the free energy alone, and the memory footprint is only $k+1$ times that of the original calculation. In the limit where the derivatives of the singular value decomposition are neglected, we establish a theoretical correspondence between our forward-mode AD and conventional impurity methods. Numerically, we find that the proposed AD algorithm can calculate internal energy and specific heat significantly higher accuracy than the impurity method at comparable computational cost. We also provide a practical procedure to extract critical exponents from derivatives of the renormalized tensor in tensor renormalization group calculations in both two and three dimensions. In addition, we discuss how to efficiently differentiate an arbitrary tensor network.

DOI: [10.1103/9wb1-rrcn](https://doi.org/10.1103/9wb1-rrcn)

I. INTRODUCTION

The tensor renormalization group (TRG) [1] is a numerical method for studying classical and quantum many-body systems. A wide range of applications has been developed, including tensor network formulations of path integrals for fermionic and bosonic theories as well as lattice gauge theories [2–47]. There have been several algorithmic developments aimed at improving computational efficiency [48–50] or achieving higher accuracy [51–57]. In the TRG framework, the partition function is approximated via a truncation based on singular value decomposition (SVD), which retains the dominant contributions while discarding subleading degrees of freedom. Beyond the partition function, the computation of physical quantities is equally essential. Three common approaches are available: numerical differentiation, the impurity-tensor method [58], and automatic differentiation [59]. However, numerical differentiation based on finite differences often suffers from numerical instability, since its accuracy strongly depends on the choice of the step size. The impurity method evaluates observables by inserting an impurity tensor into the tensor network. Depending on the target quantity, this approach yields smoother data than

numerical differentiation. Extensions of the method allow for systematic summations that directly access higher-order moments, enabling the evaluation of quantities such as the Binder cumulant or magnetization for the higher-order TRG (HOTRG) and bond-weighted TRG (BWTRG) [58,60]. However, in the impurity method, one typically employs the same projectors as those optimized for the bulk tensor which can be an additional source of systematic error. Automatic differentiation (AD) [61] offers a powerful alternative for differentiation, enabling machine precision evaluation of derivatives. AD treats the computational process as a graph, iteratively applying the chain rule to propagate derivatives. This methodology has proven highly successful in the field of deep learning [62]. Notably, Ref. [59] demonstrated a straightforward extension of AD to tensor network algorithms including TRG by interpreting the tensor network computation as a differentiable computational graph. Several studies have explored the incorporation of AD techniques within the context of the TRG [38,51,63–65]. Reference [59] uses backpropagation (reverse-mode AD), which evaluates gradients of a given target function via a backward pass, to compute the internal energy and the specific heat within the TRG framework. While reverse-mode AD is well suited to problems with many input parameters such as neural networks, its memory cost typically scales with the depth of the computational graph, because it requires storing intermediate quantities for the backward pass that propagates sensitivities from the output back to the inputs. In the TRG setting, the depth of the computational graph grows logarithmically with the system size, and, hence, the

^{*}Contact author: sugimoto@nucl.phys.tohoku.ac.jp

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.

reverse-mode memory footprint increases accordingly. This becomes particularly restrictive when one requires large volumes or considers higher-dimensional tensor networks, where even the storage of individual tensors is already demanding. Moreover, the TRG scheme is typically designed to evaluate the partition function over a range of system sizes within a single forward coarse-graining procedure. In such cases, the additional memory and computational overhead associated with backpropagation can be prohibitive, especially when analyzing the volume dependence of observables, for example, in finite-size scaling studies. Furthermore, existing implementations in [66,67] rely on high-level AD packages [68,69], which may limit their applicability to specific programming languages or environments.

Motivated by these limitations, we develop an alternative strategy to incorporate automatic differentiation into the TRG scheme by explicitly deriving the forward-mode chain rule. Our approach can be viewed as a generalization of the impurity method. It propagates derivative information along the renormalization flow in a controlled manner, without requiring backpropagation through a deep computational graph. Consequently, both the memory footprint and the computational cost remain only a small constant factor larger than those of the original TRG computation. The proposed method also achieves substantially higher accuracy while preserving the same asymptotic computational scaling as conventional impurity-based calculations in the HOTRG. Furthermore, the method is readily applicable to a broad class of TRG algorithms and can be extended straightforwardly to higher-dimensional tensor networks. Finally, we present an efficient procedure to extract critical exponents by applying the proposed AD scheme to finite-size scaling analyses.

In Sec. II, we review automatic differentiation and present our proposed algorithm, together with its theoretical connection to the impurity method of the HOTRG. We also extend our scheme to the BWTRG [51]. In Sec. III, we provide numerical results that demonstrate the efficiency of the proposed approach. We show that it yields significantly more accurate estimates while preserving the same order of computational cost as the conventional impurity method. We further perform a finite-size scaling analysis enabled by our AD-based evaluation and present additional results for the three-dimensional isotropic lattice case. Section IV is devoted to discussion and summary.

II. METHODS

A. About tensor renormalization group

We first briefly review the tensor renormalization group method [1,3]. For a translationally invariant system, the partition function Z in two dimensions can be written as a uniform tensor network:

$$Z = \text{tTr} \left[\begin{array}{ccc} \text{---} T \text{---} & \text{---} T \text{---} & \text{---} T \text{---} \\ | & | & | \\ \text{---} T \text{---} & \text{---} T \text{---} & \text{---} T \text{---} \\ | & | & | \\ \text{---} T \text{---} & \text{---} T \text{---} & \text{---} T \text{---} \end{array} \right], \quad (1)$$

where tTr denotes the contraction of all tensor indices with periodic boundary conditions. The local tensor T derived from the action is placed on each lattice site and connected according to the square (hypercube) lattice geometry. However, a direct contraction of the tensor network representation of Z is impractical, because the computational cost grows exponentially with the system volume V .

TRG methods address this difficulty by employing coarse-graining of the network, typically using a truncated singular value decomposition to control the bond dimension. In TRG schemes such as the Levin-Nave-type TRG [1,51,52] or the HOTRG type [3,49,50], the local tensor T is approximated or decomposed and contracted with its neighbors recursively, so that the number of tensors in the network decreases at each iteration. Importantly, for a translationally invariant system, the network remains uniform under coarse-graining; namely, the same tensor or unit cell is retained on every coarse-grained lattice site.

Typically, the coarse-graining map R for the HOTRG on a two-dimensional lattice along the x direction is based on the contraction:

$$\begin{array}{c} \text{---} T^{(i+1)} \text{---} \\ | \end{array} := R(T^{(i)}, T^{(i)}, E^{(i)}, F^{(i)}) \quad (2)$$

$$= \begin{array}{c} \text{---} F^{(i)} \text{---} \text{---} T^{(i)} \text{---} E^{(i)} \text{---} \\ | \quad \quad \quad | \\ \text{---} T^{(i)} \text{---} \end{array}$$

Here, $T^{(i)}$ and $T^{(i+1)}$ denote the tensors before and after coarse-graining at steps i and $i+1$, respectively, and $E^{(i)}$ and $F^{(i)}$ are projectors inserted into every horizontal link of lattice, which effectively performs a truncated SVD of $T^{(i)}$ or a larger subnetwork. In the above contraction, two adjacent tensors $T^{(i)}$ are approximated into a single tensor $T^{(i+1)}$, which we call coarse-grained tensor. By iterating the above procedure $O(\log V)$ times, the original network is reduced to a single tensor, and the approximated partition function under the periodic boundary condition is obtained by taking its tensor trace.

B. Automatic differentiation and tensor renormalization group

Consider a scalar function

$$f(\theta_1, \dots, \theta_n), \quad \theta \in \mathbb{R}^n, \quad (3)$$

where $\theta = (\theta_1, \dots, \theta_n)^\top$. We assume that f is given as a composition of sequentially defined maps

$$f(\theta) = (w_J \circ w_{J-1} \circ \dots \circ w_0)(\theta). \quad (4)$$

Here, f denotes a certain algorithm, while each w_k represents a computational routine that maps an intermediate variable to the next one. By introducing intermediate variables $\{x_k\}_{k=0}^J$ as

$$x_0 = \theta, \quad (5)$$

$$x_{j+1} = w_j(x_j) (j = 0, 1, \dots, J), \quad (6)$$

where $x_{J+1} = f$. Our goal is to evaluate the derivatives of f with respect to the parameters $\{\theta_i\}_{i=1}^n$. By applying the chain rule, we obtain

$$\frac{\partial f}{\partial \theta_i} = \frac{\partial f}{\partial x_{J+1}} \frac{\partial x_{J+1}}{\partial x_J} \dots \frac{\partial x_0}{\partial \theta_i}. \quad (7)$$

AD computes these derivatives exactly up to machine precision by decomposing f into a computational graph composed of the elementary maps $\{w_j\}$ and automatically evaluating the derivatives through the chain rule. The forward-mode AD evaluates Eq. (7) from right to left along the computational graph:

$$\frac{\partial x_j}{\partial \theta_i} = \frac{\partial x_j}{\partial x_{j-1}} \frac{\partial x_{j-1}}{\partial \theta_i}, \quad (8)$$

for $j = 1, \dots, J+1$. This strategy is useful when the number of input parameters is small while the output f is high dimensional, such as a vector or a tensor.

Another strategy evaluates Eq. (7) from left to right. This is called reverse-mode AD. Defining the adjoint variables

$$\bar{x}_j \equiv \frac{\partial f}{\partial x_j}, \quad (9)$$

reverse-mode AD propagates them backward as

$$\bar{x}_{j-1} = \bar{x}_j \frac{\partial x_j}{\partial x_{j-1}}, \quad (10)$$

for $j = J+1, \dots, 1$, yielding $\bar{x}_0 = \frac{\partial f}{\partial \theta_i}$. Since we do not have to distinguish the dependence on each θ_i until the final step, we can compute the derivatives with respect to θ_i for

$i = 1, \dots, n$ in a single backward pass. This is particularly effective when the number of input parameters is much larger than the dimension of the output, as in neural networks.

Let us discuss the TRG case. Throughout this paper, we assume that the initial tensor $T^{(0)}$ depends on the parameter β . The function f corresponds to the free energy or the partition function of the system, and the intermediate variables x_j correspond to the renormalized tensors $T^{(j)}$. In Ref. [59], a reverse-mode implementation was introduced to compute the internal energy and the specific heat for the two-dimensional Ising model. However, reverse-mode AD faces practical difficulties in TRG calculations. Each TRG step relies on a truncated SVD of $T^{(j)}$, so the map w_j depends strongly on $T^{(j)}$ through the associated projectors. Consequently, to carry out the backward pass, one has to store all projectors or the tensors $T^{(j)}$ generated at each renormalization step. This leads to significant memory and disk usage.

Another limitation is that reverse-mode AD computes the gradient of a specific target function. Denoting one TRG iteration by

$$T^{(i+1)} = R(T^{(i)}, E_1^{(i)}, E_2^{(i)}, \dots). \quad (11)$$

Here, $\{E_k^{(i)}\}$ denotes the projectors used in the i th step. Assuming that the renormalization map R increases the system size by a factor of 2, the partition function for different system sizes can be written as

$$\begin{aligned} Z(V = 2^1, \beta) &= \text{tTr}[R(T^{(0)}(\beta))], \\ Z(V = 2^2, \beta) &= \text{tTr}[(R \circ R)(T^{(0)}(\beta))], \\ &\vdots \\ Z(V = 2^n, \beta) &= \text{tTr}\left[\underbrace{R \circ \dots \circ R}_n(T^{(0)}(\beta))\right]. \end{aligned} \quad (12)$$

To obtain derivatives of Z for arbitrary system sizes with reverse-mode AD, one would, in principle, need to apply reverse-mode AD separately to R , $R \circ R$, ..., and $\underbrace{R \circ \dots \circ R}_n$. This is inefficient, because a single TRG run naturally produces the sequence of tensors $\{T^{(i)}\}$, and, thus, all system sizes in Eq. (12), within one calculation. Furthermore, the renormalized tensor $T^{(i)}$ itself contains physical information beyond the partition function, such as scaling dimensions [2] and energy spectrum [70], so TRG should be viewed as a multioutput algorithm whose outputs are the entire sequence $\{T^{(i)}\}$. From this perspective, the forward-mode AD is particularly suitable, since it propagates derivative information together with the forward renormalization flow and can provide derivatives associated with all outputs $\{T^{(i)}\}$ in a single run.

C. Differentiation of the HOTRG

We introduce the forward-mode AD of the HOTRG here. First, we consider first derivatives of the partition function Z . The partition function $Z^{(n)}$ at n th iteration is defined as

$$Z^{(n)} = \text{tTr}[T^{(n)}]. \quad (13)$$

Assume the initial tensor or action has parameter dependence with respect to β . The first derivative of $\ln Z$ with respect to β at the n th iteration is simply given by

$$\frac{\partial \ln Z^{(n)}}{\partial \beta} = \frac{\text{tTr}\left[\frac{\partial T^{(n)}}{\partial \beta}\right]}{\text{tTr}[T^{(n)}]}. \quad (14)$$

Therefore, the entire procedure requires computing only two objects at each renormalization step: the coarse-grained tensor $T^{(n)}$ and its derivative $\dot{T}^{(n)} := \frac{\partial T^{(n)}}{\partial \beta}$.

To derive the chain rule, let us consider the RG map of 2D HOTRG in Eq. (2). To calculate $\dot{T}^{(n)}$ at each step, we begin with the initial tensor $T^{(0)}$ and its derivative $\dot{T}^{(0)}$. The chain rule for the HOTRG update is expressed as the derivative of RG map R . By differentiating Eq. (2) from $i = 1, \dots, n$, sequentially, we get

$$\begin{aligned} \dot{T}^{(n+1)} &= R(\dot{T}^{(n)}, T^{(n)}, E^{(n)}, F^{(n)}) + R(T^{(n)}, \dot{T}^{(n)}, E^{(n)}, F^{(n)}) \\ &\quad + R(T^{(n)}, T^{(n)}, \dot{E}^{(n)}, F^{(n)}) + R(T^{(n)}, T^{(n)}, E^{(n)}, \dot{F}^{(n)}). \end{aligned} \quad (15)$$

The derivatives of E and F can be obtained through a differentiation of the SVD formula, which we leave for Appendix B. In the practical calculation, we adopt a Lorentzian broadening $1/x \rightarrow x/(x^2 + \eta)$ for derivatives of isometries following Ref. [59] to avoid divergence when there are degenerate singular values. In this work, we will use improved projectors refereed as squeezers [49,71,72] in the HOTRG calculations. In the Ising model which we use in this work, $\eta \rightarrow \infty$ limit corresponds to the vanishing limit of the derivatives of squeezers.¹ The above procedure completes the formulation of the first-order response within

the forward-mode scheme. The extension to second-order derivatives is completely analogous. By differentiating Eq. (15) once more with respect to β , we obtain the evolution equation for the second derivative $\ddot{T}^{(n+1)}$ as follows with $T^{(n)}$, $\dot{T}^{(n)}$, and $\ddot{T}^{(n)}$:

$$\begin{aligned} \ddot{T}^{(n+1)} &= R(\ddot{T}^{(n)}, T^{(n)}, E^{(n)}, F^{(n)}) + R(T^{(n)}, \ddot{T}^{(n)}, E^{(n)}, F^{(n)}) \\ &\quad + R(T^{(n)}, T^{(n)}, \ddot{E}^{(n)}, F^{(n)}) + R(T^{(n)}, T^{(n)}, E^{(n)}, \ddot{F}^{(n)}) \\ &\quad + 2[R(\dot{T}^{(n)}, \dot{T}^{(n)}, E^{(n)}, F^{(n)}) + R(T^{(n)}, T^{(n)}, \dot{E}^{(n)}, \dot{F}^{(n)})] \\ &\quad + R(\dot{T}^{(n)}, T^{(n)}, \dot{E}^{(n)}, F^{(n)}) + R(\dot{T}^{(n)}, T^{(n)}, E^{(n)}, \dot{F}^{(n)}) \\ &\quad + R(T^{(n)}, \dot{T}^{(n)}, \dot{E}^{(n)}, F^{(n)}) + R(T^{(n)}, \dot{T}^{(n)}, E^{(n)}, \dot{F}^{(n)}). \end{aligned} \quad (16)$$

Taking the derivative of Eq. (14) with respect to β , we obtain

$$\frac{\partial^2 \ln Z^{(n)}}{\partial \beta^2} = \frac{\text{tTr}[\ddot{T}^{(n)}] \text{tTr}[T^{(n)}] - (\text{tTr}[\dot{T}^{(n)}])^2}{(\text{tTr}[T^{(n)}])^2}. \quad (17)$$

Higher-order derivatives follow straightforwardly by iterating the same procedure. The update rule for $\dot{T}$ in Eq. (15) naively consists of four HOTRG-like contractions. However, we do not need to evaluate these four contractions separately. Instead, we use the fact that each contraction can be decomposed into a sequence of matrix multiplications along a contraction tree [73]. Rather than evaluating the four contractions independently, we differentiate the intermediate tensors that appear along this tree. With this procedure, the total matrix-multiplication cost of computing derivatives up to first order is exactly 3 times that of the original HOTRG.

The same idea extends to higher orders. Although the algorithm involves a series of HOTRG-like contractions, contraction-tree-based differentiation reduces the total numerical cost of evaluating all derivatives up to the order of k to only $\frac{(k+1)(k+2)}{2}$ times that of computing the free energy alone. For example, by explicitly rewriting Eq. (15) as

$$\begin{aligned} \dot{T}^{(n+1)} &= \text{Diagram 1} + \text{Diagram 2} + \text{Diagram 3} + \text{Diagram 4} \\ &= \left(\text{Diagram 1} + \text{Diagram 2} \right) + \left(\text{Diagram 3} + \text{Diagram 4} \right) \end{aligned} \quad (18)$$

¹This is because the initial tensor of Ising model has reflection symmetry, which yields squeezers to be isometry itself. In general squeezers, $\eta \rightarrow \infty$ does not mean $\dot{E} = \dot{F} = 0$ without taking derivatives of singular values to be zero.

where the diamond tensor represents first derivative tensor. In Eq. (18), the dominant matrix-multiplication part of the contraction is evaluated only twice.

For the second derivative, Eq. (16) can be written as

$$\begin{aligned}
 \ddot{T}^{(n+1)} = & \left(\text{Diagram 1} \right) + \left(\text{Diagram 2} \right) + 2 \times \left(\text{Diagram 3} \right) \\
 & + \left(\text{Diagram 4} \right) + \left(\text{Diagram 5} \right) + 2 \times \left(\text{Diagram 6} \right) \\
 & + 2 \times \left(\text{Diagram 7} \right) \times \left(\text{Diagram 8} \right).
 \end{aligned} \tag{19}$$

The pentagon tensors indicate its second derivative. In this contraction, the bottleneck takes just 3 times as the HOTRG. In practice, it is not necessary to execute all contractions shown in the figures independently. Instead, by performing the bulk, first-order, and second-order contractions sequentially for each intermediate tensor, one can reuse the intermediate tensors that are common to Eqs. (2), (18), and (19) rather than recomputing them multiple times. As a result, the total cost of evaluating derivatives up to first and second order is exactly 3 and 6 times that of the original HOTRG, respectively. This scaling is unchanged for general tensor network contractions. See Appendix C 1 for details. We also note that the memory footprint is only $k + 1$ times that of the original HOTRG, since we only propagate $T^{(i)}$ and its derivatives at every coarse-graining step; importantly, it does not depend on the graph depth $\log V$. Note that, to obtain the derivatives of the squeezers (isometries), we differentiate the SVD and apply the resulting rules to all contractions appearing in the original algorithm. This approach is applicable to a wide range of programming languages without relying on external AD packages, as it requires only implementing a simple update rule.

D. Differentiation of the BWTRG

Here, we employ AD rules for the BWTRG [51]. Suppose we have fundamental tensor $T^{(n)}$ and tensor $A^{(n)}, B^{(n)}, C^{(n)}, D^{(n)}$ constructed from SVD of $T^{(n)}$ and bond weight $E^{(n)}, F^{(n)}$. In the coarse-graining step of BWTRG, we consider the following contraction:

$$\text{Diagram 9} := \begin{array}{c} E^{(n)} \\ \text{---} \text{Diagram 10} \text{---} \\ D^{(n)} \text{---} A^{(n)} \\ F^{(n)} \text{---} C^{(n)} \\ B^{(n)} \text{---} E^{(n)} \end{array} \tag{20}$$

By considering the SVD of $T^{(n)}$ and its derivative, we can easily obtain $\dot{A}^{(n)}, \dot{B}^{(n)}, \dot{C}^{(n)}, \dot{D}^{(n)}$ or higher-order derivatives. Before the contraction, we introduce the following intermediate tensor

$$\text{Diagram 11} = \begin{array}{c} E^{(n)} \\ \text{---} \text{Diagram 12} \text{---} \\ A^{(n)} \\ F^{(n)} \end{array}, \tag{21}$$

$$\text{Diagram 13} = \begin{array}{c} F^{(n)} \\ \text{---} \text{Diagram 14} \text{---} \\ B^{(n)} \text{---} E^{(n)} \end{array}, \tag{22}$$

and their derivative $\dot{A}', \dot{B}'$ by chain rule. The coarse-graining rule for the BWTRG is now written as

$$\begin{aligned}
 \text{Diagram: a circle with four external lines} \quad T^{(n+1)} &:= R_{\text{BWTRG}}(A', B', C^{(n)}, D^{(n)}) \\
 &= \text{Diagram: a square with vertices } D^{(n)}, A'^{(n)}, B'^{(n)}, C^{(n)} \text{ and external lines} \quad .
 \end{aligned} \tag{23}$$

The first derivatives can be obtained by differentiating Eq. (23):

$$\begin{aligned}
 \text{Diagram: a diamond with four external lines} \quad \dot{T}^{(n+1)} &= R_{\text{BWTRG}}(\dot{A}', B', C^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(A', \dot{B}', C^{(n)}, D^{(n)}) \\
 &\quad + R_{\text{BWTRG}}(A', B', \dot{C}^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(A', B', C^{(n)}, \dot{D}^{(n)})
 \end{aligned} \tag{24}$$

$$= \left(\begin{array}{c} \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: circle with } D^{(n)} \text{ and } \dot{B}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: circle with } D^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \end{array} \right) \begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: circle with } C^{(n)} \end{array} + \begin{array}{c} \text{Diagram: circle with } D^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \end{array} \left(\begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: diamond with } \dot{A}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: circle with } C^{(n)} \end{array} \right), \tag{25}$$

$$\begin{aligned}
 \text{Diagram: a pentagon with four external lines} \quad \ddot{T}^{(n+1)} &= R_{\text{BWTRG}}(\ddot{A}', B', C^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(A', \ddot{B}', C^{(n)}, D^{(n)}) \\
 &\quad + R_{\text{BWTRG}}(A', B', \ddot{C}^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(A', B', C^{(n)}, \ddot{D}^{(n)}) \\
 &\quad + 2 \left[R_{\text{BWTRG}}(\dot{A}', \dot{B}', C^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(\dot{A}', B', \dot{C}^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(\dot{A}', B', C^{(n)}, \dot{D}^{(n)}) \right. \\
 &\quad \left. + R_{\text{BWTRG}}(A', \dot{B}', \dot{C}^{(n)}, D^{(n)}) + R_{\text{BWTRG}}(A', \dot{B}', C^{(n)}, \dot{D}^{(n)}) + R_{\text{BWTRG}}(A', B', \dot{C}^{(n)}, \dot{D}^{(n)}) \right]
 \end{aligned} \tag{26}$$

$$\begin{aligned}
 &= \left(\begin{array}{c} \text{Diagram: pentagon with } \ddot{D}^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: circle with } D^{(n)} \text{ and } \ddot{B}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: circle with } D^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: pentagon with } \ddot{D}^{(n)} \text{ and } B'^{(n)} \end{array} + 2 \times \begin{array}{c} \text{Diagram: pentagon with } \ddot{D}^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \end{array} \right) \begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: circle with } C^{(n)} \end{array} \\
 &\quad + \begin{array}{c} \text{Diagram: circle with } D^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: pentagon with } \ddot{D}^{(n)} \text{ and } B'^{(n)} \end{array} \left(\begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: pentagon with } \ddot{A}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: pentagon with } \ddot{A}'^{(n)} \\ \text{Diagram: circle with } C^{(n)} \end{array} + 2 \times \begin{array}{c} \text{Diagram: diamond with } \dot{A}'^{(n)} \\ \text{Diagram: diamond with } \dot{C}^{(n)} \end{array} \right) \\
 &\quad + 2 \times \left(\begin{array}{c} \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: circle with } D^{(n)} \text{ and } \dot{B}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: circle with } D^{(n)} \text{ and } B'^{(n)} \\ \text{Diagram: diamond with } \dot{D}^{(n)} \text{ and } B'^{(n)} \end{array} \right) \times \left(\begin{array}{c} \text{Diagram: circle with } A'^{(n)} \\ \text{Diagram: diamond with } \dot{A}'^{(n)} \end{array} + \begin{array}{c} \text{Diagram: diamond with } \dot{A}'^{(n)} \\ \text{Diagram: circle with } C^{(n)} \end{array} \right).
 \end{aligned} \tag{27}$$

Again, the cost of computing derivatives up to second order is exactly 6 times the cost of the original contraction. More generally, when derivatives up to the order of k are propagated, the leading contraction cost is multiplied by $(k+1)(k+2)/2$. In practical calculations, differentiating the SVD introduces an additional $O(D^6)$ overhead, while the remaining tensor contraction operations required to evaluate the derivatives are also multiplied by $(k+1)(k+2)/2$.

E. A connection to impurity method

Here, we explain how the forward-AD scheme connects to impurity methods. Let us consider the case evaluating the first derivatives of $\ln Z$. This value can be expressed as the summation of all possible configurations including a single $\dot{T}$ [74]:

$$\frac{\partial Z}{\partial \beta} = \text{tTr} \left[\begin{array}{c} \text{Diagram 1: } \dot{T} \text{ at top-left} \\ \text{Diagram 2: } \dot{T} \text{ at top-middle} \\ \text{Diagram 3: } \dot{T} \text{ at top-right} \\ \vdots \\ \text{Diagram N: } \dot{T} \text{ at bottom-right} \end{array} \right], \quad (28)$$

where $\dot{T}$ is called a single-impurity tensor. If there is translational invariance, Eq. (28) can be treated as a single-impurity tensor network localized at a specific lattice site. More generally, we can consider higher-order derivatives or operator insertions. In both cases, the impurity-tensor network can be systematically generated by introducing external sources and the corresponding generating function. The second derivative of Z is also expressed as a summation of all possible networks containing any two $\dot{T}$ and the sum of networks containing one second derivative, $\ddot{T}$, which is called a two-impurity tensor.

In the impurity method, tensor networks with impurities are coarse-grained by a given TRG scheme. There are two typical strategies: (i) use the same isometries as those obtained from the bulk tensor T and (ii) introduce an independent truncation for the impurity tensor itself. The former approach is often employed in HOTRG calculations of higher-order moments of physical observables and can be straightforwardly extended to higher-dimensional lattices. The latter approach is mainly practical in two-dimensional systems, because under TRG iterations impurities tend to spread over higher-dimensional lattices and for multiple impurities it becomes difficult to keep track of and sum over all possible impurity configurations. Therefore, we will examine the former case here.

Let us now discuss the relationship between our forward-AD scheme and the conventional impurity method.

1. Impurity method of the HOTRG

In the impurity method in the HOTRG, the k th impurity tensor is updated as systematic summation of networks [37,58]:

$$S_1^{(n+1)} = \frac{1}{2} [R(S_1^{(n)}, T^{(n)}, E^{(n)}, F^{(n)}) + R(T^{(n)}, S_1^{(n)}, E^{(n)}, F^{(n)})], \quad (29)$$

$$S_2^{(n+1)} = \frac{1}{4} [R(S_2^{(n)}, T^{(n)}, E^{(n)}, F^{(n)}) + R(T^{(n)}, S_2^{(n)}, E^{(n)}, F^{(n)}) + 2R(S_1^{(n)}, S_1^{(n)}, E^{(n)}, F^{(n)})], \quad (30)$$

$$\vdots \quad (31)$$

$$S_k^{(n+1)} = \frac{1}{2^k} \sum_{m=0}^k \binom{k}{m} R(S_{k-m}^{(n)}, S_m^{(n)}, E^{(n)}, F^{(n)}). \quad (32)$$

where $S_m^{(n)}$ denote the k -impurity tensors, with the initial conditions

$$S_m^{(0)} = \frac{\partial^m T}{\partial \beta^m} \quad (33)$$

and we have defined $S_0^{(i)} = T^{(i)}$ for all i . Thus, we can calculate up to k th derivatives at the cost of $\frac{(k+1)(k+2)}{2}$ times bottleneck contraction in the HOTRG. The key observation is that the usual impurity method can be regarded as the limit of our forward-mode AD formulation in which the derivatives of the SVD are set to zero up to a normalization constant. Let us look at Eq. (15). By setting $\dot{E} = \dot{F} = 0$, or $\eta = \infty$, we recover the HOTRG contraction with a single derivative tensor, where the projectors are those obtained from the bulk tensor T . This is essentially the standard impurity-tensor update scheme in the HOTRG method,

once we identify $\tilde{T}$ with an impurity tensor. Likewise, for the second derivative, by interpreting $\tilde{T}$ as a two-impurity tensor, we obtain an analogous correspondence. As discussed above, since an impurity tensor can be expressed in terms of derivatives of the initial tensor, the forward-mode AD formulation can be regarded as an extension of the impurity method. This relationship naturally extends to higher-order derivatives. Therefore, even for higher-order derivatives, one can directly derive the update rules of the impurity-tensor method corresponding to arbitrary higher-order moments by applying the chain rule step by step, without explicitly accounting for the systematic summation.

2. Impurity method of the BWTRG

To clarify the correspondence with the multi-impurity method for the BWTRG [60], we rewrite the BWTRG and its differentiation rules in a projector formalism [75,76]. This can be achieved by a simple modification: Replace the truncated SVD and the bond-weight splitting by an insertion of projectors and singular values:

$$T^{(n)} \approx U \Sigma^{\frac{1-k}{2}} \Sigma^k \Sigma^{\frac{1-k}{2}} V^\dagger \quad (34)$$

$$= U \Sigma^{\frac{1-k}{2}} (\Sigma^{-\frac{1-k}{2}} U^\dagger T^{(n)} V \Sigma^{-\frac{1-k}{2}}) \Sigma^{\frac{1-k}{2}} V^\dagger, \quad (35)$$

where k denotes the hyperparameter of the BWTRG, the projectors U , $V^\dagger$, and Σ are constructed from the truncated singular vectors and singular values of $T^{(n)}$, and the term inside the parentheses plays the role of the bond weight $E^{(n)}$ or $F^{(n)}$. As in the HOTRG case, in the limit where the SVD differentiation is ignored and only the derivative of $T^{(n)}$ is retained, the chain rule in Eqs. (23) and (25) reproduces the update rule for the squared-lattice version of the multi-impurity method introduced in Ref. [60] up to a normalization constant. In this impurity limit, the derivatives of $E^{(n)}$ and $F^{(n)}$ play the role of spatial averages of higher-order derivatives. Note that, since the derivative of $T^{(n)}$ does not appear in the update rule in the BWTRG, the initial tensor in Eq. (28) is inadequate in this limit. Therefore, we must incorporate the temperature dependence into the initial bond weight via diagonalization of the Boltzmann weight and take $T^{(0)}$ to be constant with respect to β . We also note that, once derivatives of the SVD are neglected, the derivatives of $E^{(n)}$ and $F^{(n)}$ are no longer constrained to be diagonal matrices. The difference between AD and the impurity method was already pointed out in Ref. [60], where the impurity method effectively ignores the β dependence of the isometric tensors. In our formulation, this distinction becomes clearer. Finally, we emphasize that, even if a systematic impurity method cannot be defined, our approach enables the computation of higher-order moments up to the order of k with machine precision, with a matrix-multiplication cost that is

multiplied by a factor of $\frac{(k+1)(k+2)}{2}$ and additional $O(D^6)$ to construct derivatives of the SVD.

III. NUMERICAL RESULTS

In this section, we demonstrate the power of forward AD through numerical calculations for the Ising model using both HOTRG and BWTRG.

A. 2D case

We first consider the two-dimensional classical Ising model on a square lattice. The Hamiltonian and the partition function are given in terms of the spin variable $\sigma \in \{-1, 1\}$:

$$H(\{\sigma\}) = -\sum_{\langle i,j \rangle} \sigma_i \sigma_j, \quad Z(\beta) = \sum_{\{\sigma\}} e^{-\beta H(\{\sigma\})}, \quad (36)$$

where $\langle i, j \rangle$ denotes nearest-neighbor interaction and $\{\sigma\}$ denotes the sum over all possible spin configurations. β denotes the inverse temperature $1/T$. In the tensor network representation in Eq. (1), the local tensor T is explicitly given by

$$T_{xyx'y'} = 2(\cosh \beta)^2 (\sqrt{\tanh \beta})^{x+y+x'+y'} \delta_{(x+y+x'+y') \bmod 2, 0}, \quad (37)$$

where $x, y, x', y' \in \{0, 1\}$ are bond indices corresponding to the positive and negative directions of the x and y axes, respectively. The derivatives of the initial tensor are obtained by analytically differentiating Eq. (37) with respect to β . Alternatively, one may employ AD libraries to compute the derivatives in the general case. Once we obtain derivatives of initial tensor, we can iteratively apply the forward-mode AD rules.

We now apply the forward-mode AD to the HOTRG method. We perform the second-order forward-mode AD for the 2D Ising model using HOTRG with bond dimension $D = 80$. We vary the regularization parameter as

$$\eta \in \{10^{-20}, 10^{-16}, 10^{-12}, 10^{-8}, 10^{-4}, 10^0, \infty\}$$

and compare the results with those obtained by the impurity method. We compute the internal energy U and the specific heat C as functions of the temperature $T = 1/\beta$. Figures 1(a) and 1(b) show the relative errors of the internal energy $\delta U = \frac{|U - U_{\text{exact}}|}{|U_{\text{exact}}|}$ and the specific heat $\delta C = \frac{|C - C_{\text{exact}}|}{|C_{\text{exact}}|}$ at $V = 2^{40}$ with respect to the known exact solutions. In both figures, the forward-mode AD results are shown by colored circles for each value of η , while the impurity results are shown by black symbols.

In Fig. 1(a), the forward-mode AD results are more accurate than the impurity results. As η decreases, the relative error decreases and approaches the exact value and

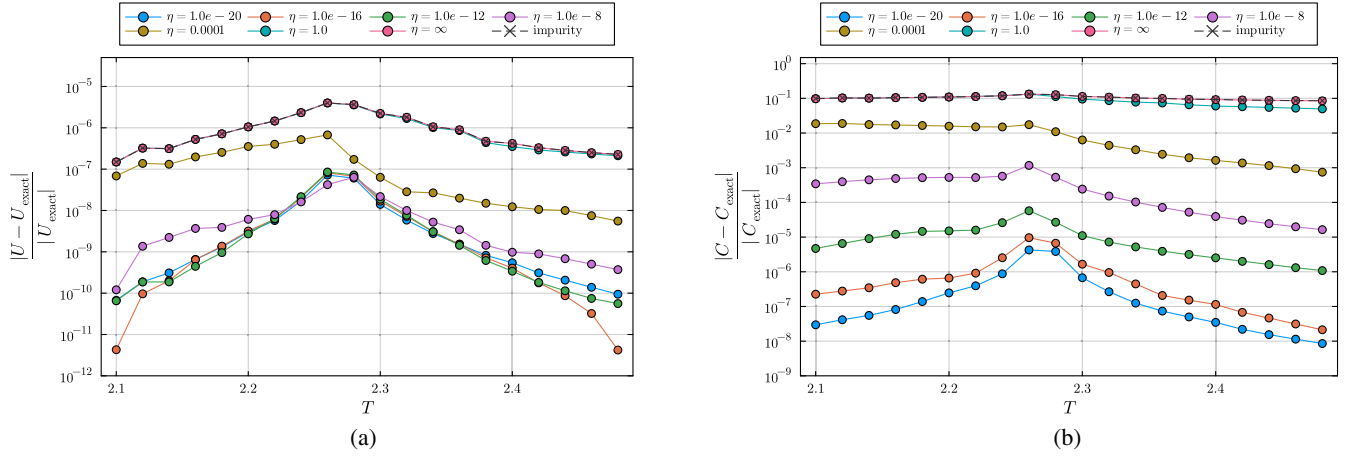


FIG. 1. Relative errors of (a) the internal energy and (b) the specific heat obtained from second-order forward-mode AD within HOTRG at $D = 80$. The horizontal axis shows the temperature. Results are shown for several regularization parameters η and are compared with the impurity method.

$\eta \leq 10^{-12}$ provide almost comparable results with each other. For $T = 2.1$, the forward-mode AD provides $\delta U \approx O(10^{-11})$ which is almost consistent with exact results. It is also shown that, as η increases, the forward-mode AD results approach those of the impurity method, and they are consistent in the limit $\eta = \infty$. This provides numerical evidence that, in the HOTRG, the impurity method corresponds to the limit of the forward-mode AD in which the derivatives of the SVD are effectively set to zero.

The difference becomes more pronounced for the second derivative, i.e., the specific heat, as shown in Fig. 1(b). As η decreases, the accuracy improves significantly, and the worst performance is again observed for the impurity method. In particular, the impurity method (equivalently, $\eta = \infty$) exhibits about $O(10^{-1})$ relative errors over a wide temperature range, whereas the forward-mode AD with $\eta = 10^{-20}$ achieves errors smaller than 10^{-5} . The improvement is especially clear in both the high- and low-temperature regimes, where the forward-mode AD calculation is approximately 10^7 times more accurate than the impurity method at $T = 2.248$ and $\eta = 10^{-20}$. These results demonstrate that the forward-mode AD enables substantially higher accuracy than the impurity method for calculating thermodynamic quantities within the HOTRG.

Next, we discuss the computational cost. We first measure the elapsed time of the bottleneck contraction part in Eq. (2) and its derivatives, or impurity update rule. Figure 2 shows the elapsed time of the bottleneck contraction in HOTRG for the forward-mode AD computations up to first and second order and for the corresponding first- and second-order impurity-tensor computations. These correspond to Eqs. (2), (18), and (19) for the forward-mode AD and to Eqs. (2), (29), and (30) for the impurity method. For the forward-mode AD, we employ contraction-tree-based differentiation described in Appendix C 1. All benchmarks are performed on a single CPU using

randomly generated dense tensors, and the bond dimension is varied from $D = 20$ to 70. The vertical axis shows the elapsed time normalized by that of the original HOTRG contraction in Eq. (2). The dotted line indicates theoretical scaling factor $\frac{(k+1)(k+2)}{2}$ for derivatives up to the order of k . We find that both the forward-mode AD method and the impurity-tensor method closely follow the expected scaling, indicating that their bottleneck costs are essentially identical, as expected theoretically.

Figure 3 shows the total elapsed time of HOTRG computations up to the first and second derivatives at volume $V = 2^{40}$ with $D = 20$ –70, measured on a single CPU. The vertical axis shows the elapsed time of each method normalized by the execution time of the original HOTRG without derivatives. The dotted line again represents theoretical bottleneck scaling factor $\frac{(k+1)(k+2)}{2}$. In the two-dimensional HOTRG with the squeezer formulation, each coarse-graining step involves three SVDs, and, hence, the subleading costs are non-negligible compared to the bottleneck $O(D^7)$ cost. As a result, the observed normalized runtime is smaller than the ideal bottleneck scaling. For both first- and second-derivative cases, the forward-mode AD method becomes slightly more expensive in practice, but it still provides comparable results. These deviations arise from the additional overhead of computing the squeezers and their derivatives.

Overall, the forward-mode AD exhibits comparable computational scaling and, for the bottleneck contraction, incurs essentially the same computational cost as the impurity method while achieving significantly higher accuracy.

Next, we investigate the BWTRG case. Figure 4 shows the relative error of the internal energy with respect to the exact value, computed using the forward-mode AD formulation of BWTRG with $\eta = 10^{-20}$. The bond dimension is varied from $D = 30$ to 128. For comparison, we also include the $D = 128$ multi-impurity BWTRG data reported

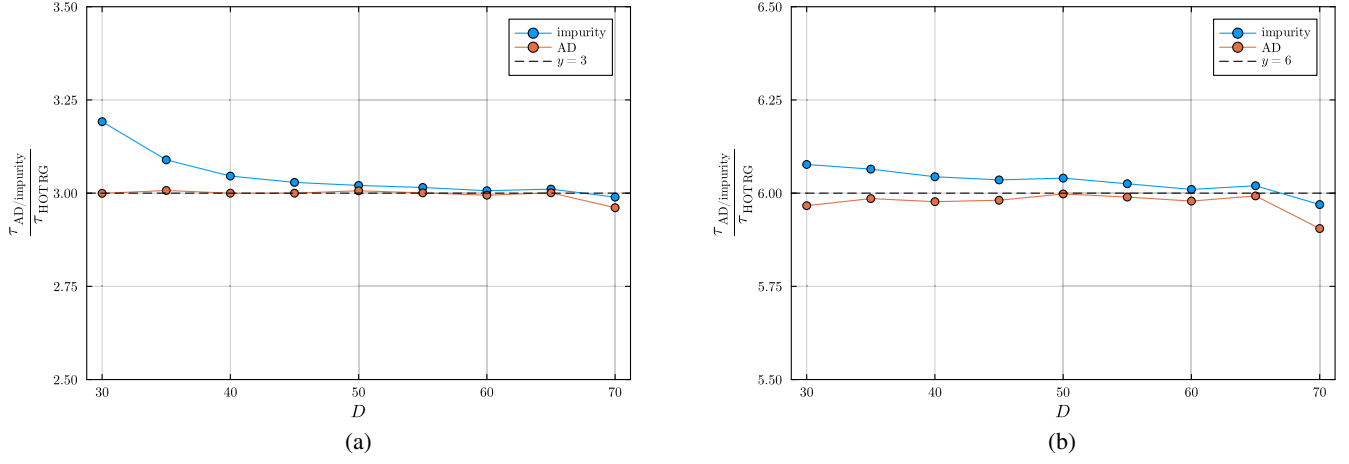


FIG. 2. Comparison of the elapsed time of the coarse-graining part of the forward-mode AD and the impurity-tensor method as a function of D . $\tau_{\text{AD/impurity}}$ denotes the computational time of the AD and impurity method, respectively. τ_{HOTRG} denotes the computational time of the HOTRG method alone. (a) Elapsed time of the bottleneck contraction part for obtaining up to the first-order derivative, normalized by the original HOTRG contraction time without derivatives. (b) Total elapsed time of HOTRG computations up to the second derivatives, normalized by the execution time of the original HOTRG. The dotted line indicates theoretical scaling factor $\frac{(k+1)(k+2)}{2}$.

in Ref. [60]. We further present the impurity limit of the forward-mode AD formulation, in which we use an alternative representation of the initial tensor and set the derivatives of the SVD to zero, as described in Sec. II E 2. Even at $D = 30$, the forward-mode AD results are more accurate than the multi-impurity result at $D = 128$, and they systematically improve as D increases. In the impurity limit, our results are consistent with the multi-impurity results obtained with the triad representation of BWTRG. We note that our BWTRG implementation is based on the original Levin-Nave-type TRG [1,51] and does not use a

partial SVD; accordingly, the computational cost scales as $O(D^6)$. In contrast, Ref. [60] employs a triad representation, leading to an $O(D^5)$ scaling. Nevertheless, we have confirmed that the two implementations yield consistent free energies at $D = 128$. We also confirmed that the two representations of the initial tensor yield nearly identical free energies and internal energies for finite η . Another remark is that the impurity limit based on the initial tensor in Eq. (37) does not correspond to a systematic summation of impurity-tensor networks, and, therefore, we could not find meaningful results. These observations indicate that,

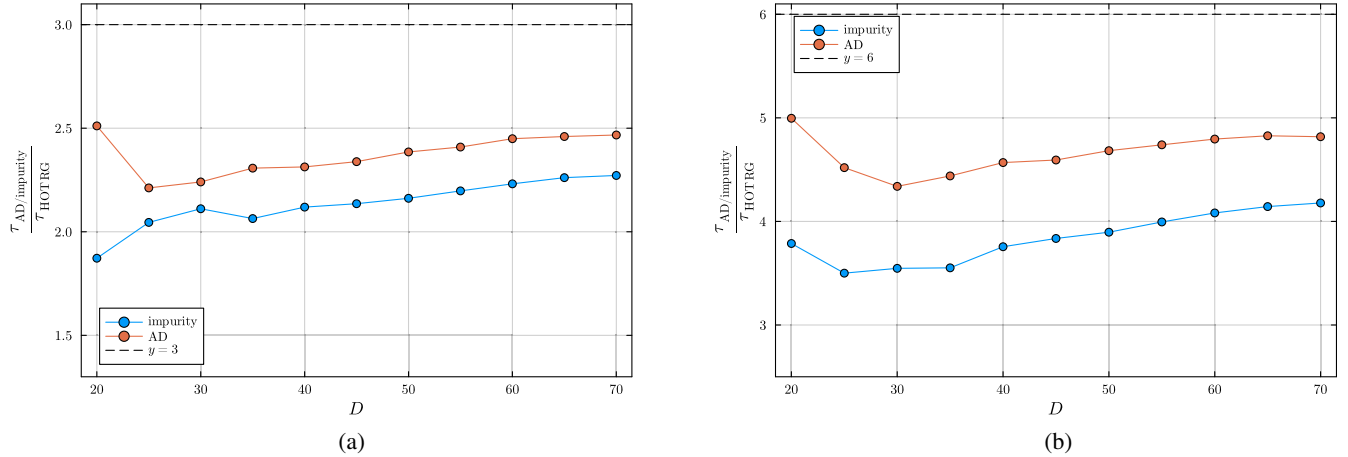


FIG. 3. Elapsed time comparison between the forward-mode AD and the impurity-tensor method of the HOTRG method as a function of D . $\tau_{\text{AD/impurity}}$ denotes the computational time of the AD and impurity method, respectively. τ_{HOTRG} denotes the computational time of the usual HOTRG method alone. (a) Elapsed time of the obtaining up to first derivatives on a $V = 2^{40}$ lattice, normalized by the original HOTRG contraction time without derivatives. (b) Total elapsed time of HOTRG computations up to the second derivatives on a $V = 2^{40}$ lattice, normalized by the execution time of the original HOTRG. The dotted line indicates theoretical scaling factor $\frac{(k+1)(k+2)}{2}$.

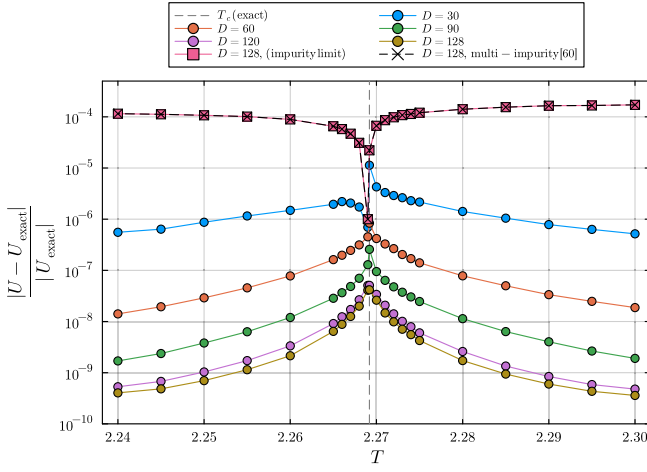


FIG. 4. Relative error of the internal energy with respect to the exact result as a function of temperature. Circles show the forward-mode AD data for various bond dimensions. The impurity limit corresponds to setting the SVD derivatives to zero and using the bond-based initial tensor described in Sec. II E 2; these data are shown by square symbols. The multi-impurity data from Refs. [60,77] are converted into relative errors and shown as black symbols. The dotted line indicates the exact critical temperature.

when there exists a correspondence between the impurity method and the limit in which the SVD derivatives are set to zero, the impurity method serves as a theoretical lower bound on the accuracy of the forward-mode AD formulation. Finally, we note that the original BWTRG study [51] already used automatic differentiation to compute the internal energy. Here, we include this comparison to demonstrate that our explicit forward-mode formulation reproduces accurate results.

B. Some applications

Since TRG based on the forward-mode AD directly calculates derivatives of renormalized tensor at each coarse-graining step, it is also possible to differentiate quantities obtained from renormalized tensors, not only the derivatives of the partition function. The Gu-Wen ratio or partition function ratio X [2] is an important quantity

computed from the renormalized tensor. The definition of X at the n th coarse-graining step is²

$$X^{(n)} = \frac{\left(\begin{array}{c} \text{Diagram: } T^{(n)} \text{ tensor with two loops} \end{array} \right)^2}{\begin{array}{c} \text{Diagram: } T^{(n)} \text{ tensor with two loops} \end{array}}. \quad (38)$$

It effectively measures the ground-state degeneracy and, therefore, serves as an order parameter for the Ising model. It was reported that the universal value of X at the critical point is determined by the conformal field theory spectrum [60,78]. It has also been reported that partition function ratio can be extended to systems with continuous symmetries [79]. In Ref. [60], it is shown that, since X is a dimensionless quantity, it obeys the finite-size scaling form

$$X = g(L^{1/\nu}\tau), \quad (39)$$

with $\tau = (T - T_c)/T_c$, where T_c is the critical temperature, g is a scaling function, and ν is the critical exponent. By differentiating Eq. (39) with respect to T , we obtain

$$\frac{\partial X}{\partial T} = \frac{1}{T_c} L^{1/\nu} g'(L^{1/\nu}\tau). \quad (40)$$

From Eq. (40), at $T = T_c$ ($\tau = 0$) we have

$$\log \left| \frac{\partial X}{\partial T} \right|_{T=T_c} = \frac{1}{\nu} \log L + A, \quad (41)$$

where A is constant. Equation (41) predicts a linear dependence of $\log |\frac{\partial X}{\partial T}|_{T=T_c}$ on $\log L$ with slope of $1/\nu$. Therefore, by evaluating Eq. (41) for multiple system sizes at critical temperature, we can extract $1/\nu$ from a linear fit. A similar analysis was used for a q -state Potts model by impurity method with the Binder parameter [58]. The diagrammatic representation of $\frac{\partial X}{\partial T}$ at the n th coarse-graining step is given by

$$\frac{\partial X^{(n)}}{\partial T} = -2\beta^2 \left[\frac{\begin{array}{c} \text{Diagram: } \dot{T}^{(n)} \text{ and } T^{(n)} \text{ tensors} \end{array}}{\begin{array}{c} \text{Diagram: } T^{(n)} \text{ tensor} \end{array}} - \frac{\begin{array}{c} \text{Diagram: } \dot{T}^{(n)} \text{ and } T^{(n)} \text{ tensors with loops} \end{array}}{\begin{array}{c} \text{Diagram: } T^{(n)} \text{ tensor with loops} \end{array}} \right]. \quad (42)$$

²Here, we denote the renormalized tensor $T^{(n)}$ including bond weights.

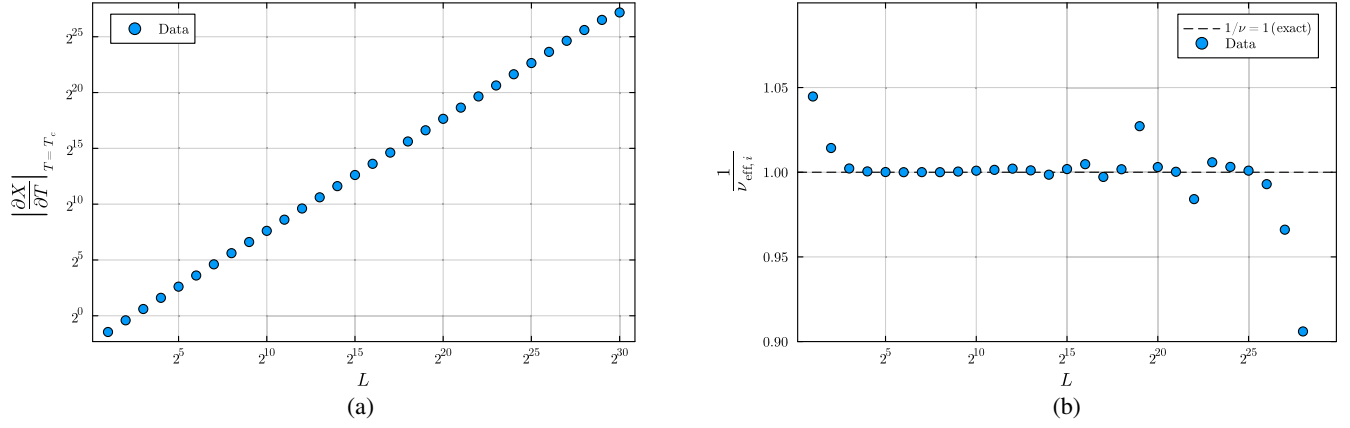


FIG. 5. Two-dimensional Ising model computed using the forward-mode AD BWTRG at $D = 130$. (a) L dependence of $|\frac{\partial X}{\partial T}|_{T=T_c}$. (b) L dependence of $1/\nu_{\text{eff},i}$.

Since our forward-mode AD formulation can calculate derivatives of T at every TRG step, a calculation at the critical temperature directly gives the volume dependence of $\frac{\partial X^{(n)}}{\partial T}$. We compute the critical exponent of the Ising model by a linear fit of Eq. (41). The critical temperature is determined as the point at which X becomes discontinuous or where it takes an intermediate value that is approximately volume independent at $L = 2^{30}$.

Figure 5(a) shows typical behavior of the data obtained by BWTRG at $D = 130$ and the critical temperature $T_c(D = 130) = 2.2691853$ estimated by X . Figure 5(b) shows an effective critical exponent measured by

$$\frac{1}{\nu_{\text{eff},i}} = \frac{\log_2 |\frac{\partial X}{\partial T}|_{i+1} - \log_2 |\frac{\partial X}{\partial T}|_i}{\log_2 L_{i+1} - \log_2 L_i} \quad (43)$$

for all data points. The data exhibit a plateau near $1/\nu = 1$ for small system sizes. As the system size increases, the data points fluctuate; this is likely due to finite bond dimension effects [80] and the limited resolution in the estimated critical temperature. Using data from $L = 2^5$ to 2^9 , we obtain $\frac{1}{\nu}(D = 130) = 1.000053(8)$, which is close to the exact value $1/\nu = 1$. Figure 6 shows the temperature dependence of the estimated critical exponent around $T_c(D = 130)$ for $D = 130$. The horizontal axis represents $T - T_c$. As the resolution of the estimated critical temperature is improved, the results become more stable with respect to temperature. Even at $T - T_c(D = 130) = 10^{-5}$, we obtain $1.00079(16)$, which differs from the exact value only by $O(10^{-4})$.

We also show the $1/D$ dependence of $1/\nu$ together with the result in Ref. [60], obtained by a scaling-collapse analysis of X around the critical temperature (see Fig. 7). The critical temperature is calculated with a resolution of 5×10^{-8} for all bond dimensions in order to examine the D dependence precisely. The estimated $1/\nu$ approaches the exact value asymptotically, and the results

are more stable and accurate than those reported in Ref. [60]. These results indicate that performing a linear fit to $\partial X/\partial T$ provides a useful and reliable means of extracting the critical scaling behavior.

We also extend the forward-mode AD formulation to the three-dimensional HOTRG. In the following, we define $X^{(n)}$ as

$$X^{(n)} = \frac{(Z(L_x, L_y, L_z))^2}{Z(2L_x, L_y, L_z)}, \quad (44)$$

where $Z(L_x, L_y, L_z)$ is the partition function obtained after the n th coarse-graining step with $L_x = L_y = L_z = 2^n$ and the denominator represents the spatially extended partition function in the $\hat{x}$ direction. We always perform coarse-graining by HOTRG in the order $\hat{x}, \hat{y}, \hat{z}$.

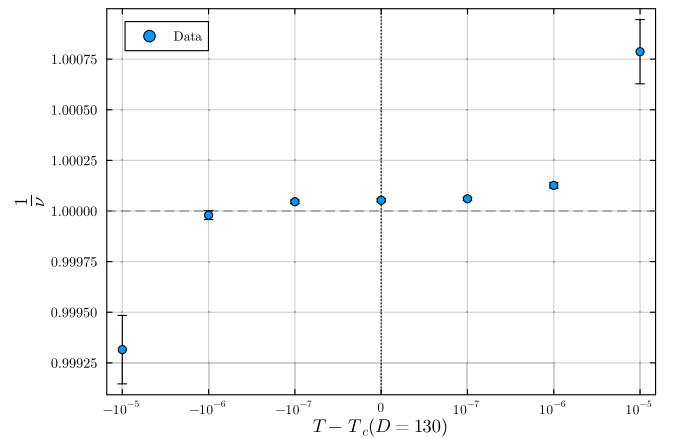


FIG. 6. The critical exponent $1/\nu$ at $D = 130$ obtained using the forward-mode AD BWTRG, plotted as a function of $T - T_c$, where $T_c(D = 130) = 2.2691853$.

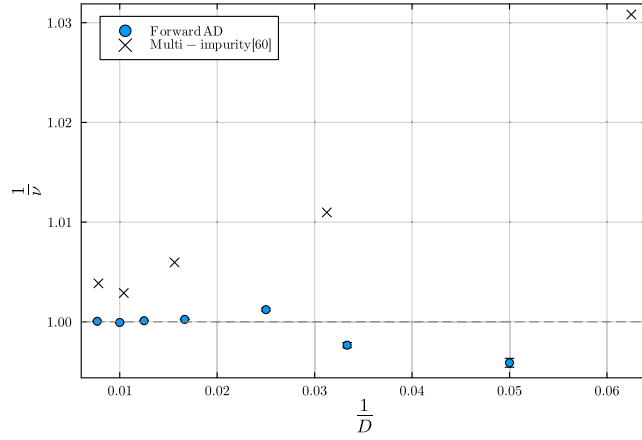


FIG. 7. $1/D$ dependence of the critical exponent $1/\nu$ obtained from the forward-mode AD BWTRG (circles). For comparison, the results from the scaling-collapse analysis taken from Refs. [60,77] are shown by black symbols. The dotted line indicates the exact value.

We perform calculations for the three-dimensional Ising model at $D = 32$ and at the estimated critical temperature $T_c(D = 32) = 4.5079$, determined from X . The volume dependence of $|\partial X/\partial T|_{T=T_c}$ and $1/\nu_{\text{eff},i}$ is shown in Figs. 8(a) and 8(b). Even in the three-dimensional case, $\log |\partial X/\partial T|_{T=T_c}$ exhibits linear scaling with respect to $\log L$. However, $1/\nu_{\text{eff},i}$ shows a strong dependence on L and no clear plateau is observed, unlike in the two-dimensional case. From a linear fit for $L = 2^2$ to 2^7 , we obtain $\nu(D = 32) = 0.571(4)$.

Figure 9 shows the $1/D$ dependence of the estimated $1/\nu$ together with the known value $\nu = 0.629971(4)$ obtained from the conformal bootstrap [81]. The critical temperatures are estimated with a resolution of 10^{-4} . As D increases, $1/\nu$ becomes closer to the known value. However, at small bond dimensions, we observed the

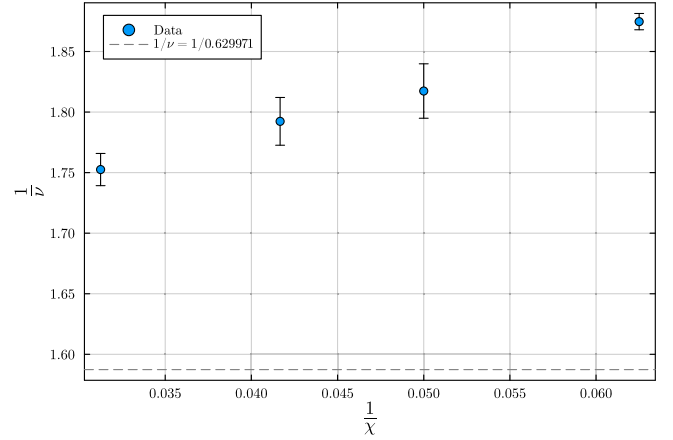
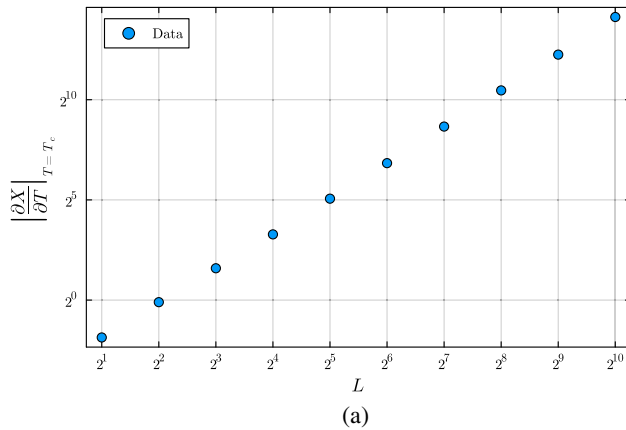


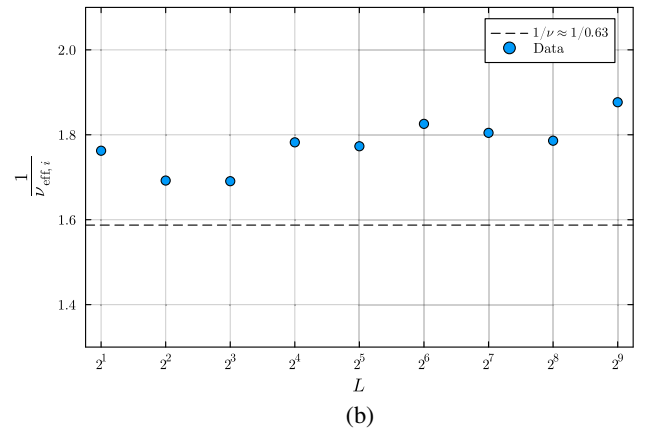
FIG. 9. $1/D$ dependence of the critical exponent $1/\nu$ obtained from the 3D HOTRG. The dotted line indicates the known value from Ref. [81].

estimated value strongly depends on the choice of the volume range and on the definition of X , namely, on which direction is extended. We also comment on the resolution of T_c . Since a more precise determination of T_c is computationally expensive, we performed the calculation with a resolution of only 10^{-4} , which is of the same order as the convergence of $T_c(D)$ with respect to D . Therefore, the estimated value also includes a systematic error coming from how close the estimated critical temperature is to the actual T_c at fixed D .

From the above discussion, we conclude that, in the three-dimensional case, the estimate is affected by several additional sources of systematic error. We attribute the deviation from the known value, as well as the instabilities discussed above, to the limited bond dimension in the three-dimensional HOTRG calculation. We also find that the specific heat is not smooth at accessible D in three



(a)



(b)

FIG. 8. Three-dimensional Ising model computed using the forward-mode AD HOTRG at $D = 32$. (a) L dependence of $|\frac{\partial X}{\partial T}|_{T=T_c}$. (b) L dependence of $1/\nu_{\text{eff},i}$.

dimensions (data not shown), which is likely due to discontinuities in the SVD differentiation and numerical instability associated with the large degeneracy of singular values as well as the limited bond dimension in the three-dimensional Ising model. We leave further investigation at larger D for future work.

IV. DISCUSSIONS AND SUMMARY

In this paper, we have proposed a simple framework for incorporating automatic differentiation into tensor renormalization group methods based on forward-mode AD. In our formulation, the memory cost for derivatives up to the order of k scales as $k + 1$ times that of the original TRG, independent of the graph depth while the contraction cost increases by a factor $(k + 1)(k + 2)/2$. The proposed forward-mode AD framework has been shown to be applicable to various TRG schemes, including HOTRG and BWTRG. In the limit where derivatives of the SVD are neglected, a theoretical correspondence to impurity methods can be established. In numerical calculations, derivatives of the partition function, including the internal energy and specific heat, have been accurately evaluated using the forward-mode AD formulation. In the impurity limit, consistency with established impurity approaches for both HOTRG and BWTRG is confirmed. Moreover, forward-mode AD achieves significantly higher accuracy than the impurity method, which neglects the parameter dependence of the isometric tensor [60]. We have also shown that, for the bottleneck tensor contraction in each coarse-graining step of the HOTRG, the computational cost for evaluating derivatives up to the order of k increases by a factor of $(k + 1)(k + 2)/2$, while the total cost of the HOTRG is slightly higher than that of the impurity method, due to the additional evaluations associated with differentiating the SVD or the squeezers. In Appendix C 1, we have shown that the same scaling behavior can be achieved for general tensor networks.

The forward-mode AD based TRG enables finite-size scaling analyses of derivatives of the Gu-Wen ratio, yielding a highly accurate estimate of the critical exponent $1/\nu$. This is a key advantage of the forward-mode AD, because the volume dependence of physical observables is naturally accessible within a single forward pass of the TRG flow.

We have also extended our algorithms to three-dimensional cases, where the increasing cost with D limits the accessible bond dimension, which leads to a less accurate estimate of ν .

Finally, we discuss degenerate singular values and AD. In the presence of degenerate singular values at the truncation threshold D , the TRG update map can become discontinuous with respect to β , and its derivatives may, therefore, also become discontinuous. In tensor network

calculations, truncating a degenerate multiplet should be avoided, because the singular value spectrum typically encodes symmetry information of the system [82]. Cutting a degenerate multiplet midway explicitly breaks the symmetry and may drive the renormalization flow toward an incorrect fixed-point tensor. Moreover, degeneracy implies a rotational freedom within the degenerate subspace: The singular vectors are not uniquely defined but are determined only up to a unitary transformation. This internal freedom can be viewed as a gauge freedom. If one truncates inside the degenerate subspace, the resulting truncation projector becomes gauge dependent, leading to numerical instabilities and inconsistencies among different implementations or algorithms. Although one may attempt to choose D so as to avoid truncating degenerate singular values, it is, in general, difficult to guarantee this condition at every TRG step. A possible remedy is to enforce gauge fixing by aligning the singular vectors continuously as β is varied, which can improve numerical stability and apparent smoothness in β [24,83]. However, such a procedure may still break the symmetry encoded in the tensor if the truncation itself splits the degenerate multiplet. Another strategy is to keep or discard the entire degenerate sector at the cutoff, effectively allowing the bond dimension to vary. While this preserves the symmetry, it introduces a β -dependent effective bond dimension and can, therefore, lead to discontinuities in the truncation scheme. In our 3D HOTRG calculations, this issue is more severe, because the accessible bond dimension D is not large enough to suppress such truncation-induced discontinuities, resulting in discontinuous specific heat. A similar issue arises when evaluating order parameters associated with Z_2 symmetry in the Ising model, such as the squared magnetization. Introducing an external source h in the Hamiltonian allows one, in principle, to apply AD and then take the limit $h \rightarrow 0$. In practice, however, we found that the squared magnetization at exactly $h = 0$ is unstable with respect to the RG steps due to the instability of the SVD derivatives associated with degenerate singular values. This instability can be mitigated by introducing an extremely small but finite h , at the cost of explicitly breaking the spin-flip symmetry. In the impurity limit, the instability is avoided, because derivatives associated with the SVD are entirely neglected; however, this does not necessarily imply improved accuracy.

A natural next step is to push to larger bond dimensions in three or four dimensions by incorporating symmetry-blocking techniques [84] or by adopting linear renormalization schemes [85]. We also plan to extend the method to magnetization or more generally, correlation functions in a stable way [83]. It is also interesting to extend the forward-mode AD to optimization-based TRG algorithms, for example, TRG with randomized SVD [48,86], which is particularly useful in higher dimensions, and Tensor

Network Renormalization-type algorithms [52–56]. Since optimization-based TRG methods require many QR decomposition and SVD operations at each coarse-graining step, reverse-mode differentiation becomes substantially more demanding.

ACKNOWLEDGMENTS

We thank Shinichiro Akiyama, Jutho Haegeman, Atsushi Ueda, and Adwait Naravane for fruitful discussions. Y. S. is supported by Graduate Program on Physics for the Universe (GP-PU), Tohoku University, and JSPS KAKENHI Grant No. 25KJ0537. Numerical calculations were performed on the supercomputer Pegasus under the Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available upon publication because it is not technically feasible and/or the cost of preparing, depositing, and hosting the data would be prohibitive within the terms of this research project. The data are available from the authors upon reasonable request.

APPENDIX A: ABOUT NORMALIZATION

In practical TRG implementations, one must carefully avoid numerical overflow. A common strategy is to normalize the tensor at each coarse-graining step as

$$\tilde{T}^{(i)} = \frac{T^{(i)}}{\text{tr}[T^{(i)}]}, \quad (\text{A1})$$

where $\tilde{T}^{(i)}$ denotes the normalized tensor at the i th step. By such normalization, the free energy at the n th iteration can be written as

$$\frac{\ln Z^{(n)}}{V^{(n)}} = \sum_{i=0}^n \frac{\ln k^{(i)}}{2^i}, \quad (\text{A2})$$

where $k^{(i)} = \text{tr}[T^{(i)}]$ and $V^{(n)} = 2^n$. In this way, the first derivative of $\ln Z/V$ with respect to β at the n th iteration is given by

$$\frac{1}{V^{(n)}} \frac{\partial \ln Z^{(n)}}{\partial \beta} = \sum_{i=0}^n \frac{1}{2^i} \frac{\dot{k}^{(i)}}{k^{(i)}}, \quad (\text{A3})$$

where

$$\dot{k}^{(i)} = \frac{\partial k^{(i)}}{\partial \beta} = \text{tr}[\dot{T}^{(i)}]. \quad (\text{A4})$$

In the next coarse-graining step, we have to use the derivative of normalized tensor $\tilde{T}^{(i)}$ and its derivative,

$$\dot{\tilde{T}}^{(n)} = \frac{\dot{T}^{(n)} k^{(n)} - T^{(n)} \dot{k}^{(n)}}{(k^{(n)})^2}. \quad (\text{A5})$$

For the second derivative case, we obtain

$$\ddot{\tilde{T}}^{(n)} = \frac{\ddot{T}^{(n)} (k^{(n)})^2 - 2\dot{T}^{(n)} \dot{k}^{(n)} k^{(n)} - T^{(n)} \ddot{k}^{(n)} k^{(n)} + 2T^{(n)} (\dot{k}^{(n)})^2}{(k^{(n)})^3} \quad (\text{A6})$$

and

$$\frac{1}{V^{(n)}} \frac{\partial^2 \ln Z^{(n)}}{\partial \beta^2} = \sum_{i=0}^n \frac{1}{2^i} \left[\frac{\ddot{k}^{(i)}}{k^{(i)}} - \left(\frac{\dot{k}^{(i)}}{k^{(i)}} \right)^2 \right]. \quad (\text{A7})$$

APPENDIX B: DIFFERENTIATION OF SVD

In this section, we write down the forward-mode AD rule for SVD used in HOTRG and BWTRG. This section is based on Ref. [87]. In these algorithms, the decomposition is applied to square matrices; hence, we restrict to $A(\beta) \in \mathbb{R}^{n \times n}$. Let $A(\beta)$ be a smooth function of a scalar parameter β , and assume we have a full SVD $A = U \Sigma V^T$

with $U, V \in \mathbb{R}^{n \times n}$. For simplicity, we assume that even numerically small singular values are treated as finite, so that the corresponding singular vectors are nontrivial. We denote

$$\dot{A} = \frac{dA}{d\beta}, \quad \ddot{A} = \frac{d^2 A}{d\beta^2}. \quad (\text{B1})$$

We use the full SVD in the form

$$A = U \Sigma V^T, \quad (\text{B2})$$

where $U, V \in \mathbb{R}^{n \times n}$ are orthogonal,

$$U^T U = I_n, \quad V^T V = I_n, \quad (\text{B3})$$

and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$ with $\sigma_i > 0$.

Let us introduce the matrix

$$P = U^T \dot{A} V. \quad (\text{B4})$$

Then the first derivative of the singular values is given by its diagonal part,

$$\dot{\sigma}_i = P_{ii}, \quad \dot{\Sigma} = \text{diag}(\dot{\sigma}_1, \dots, \dot{\sigma}_n). \quad (\text{B5})$$

For the singular vectors, we parametrize the derivatives as

$$\dot{U} = U \Omega_U, \quad \dot{V} = V \Omega_V, \quad (\text{B6})$$

where $\Omega_U, \Omega_V \in \mathbb{R}^{n \times n}$ satisfy the following conditions:

$$\Omega_U^T = -\Omega_U, \quad \Omega_V^T = -\Omega_V. \quad (\text{B7})$$

For $i \neq j$, assuming $\sigma_i \neq \sigma_j$, we then have

$$(\Omega_U)_{ij} = \frac{\sigma_j P_{ij} + \sigma_i P_{ji}}{\sigma_j^2 - \sigma_i^2}, \quad (\text{B8})$$

$$(\Omega_V)_{ij} = \frac{\sigma_i P_{ij} + \sigma_j P_{ji}}{\sigma_j^2 - \sigma_i^2}. \quad (\text{B9})$$

In tensor network calculations, one often encounters degenerate singular values, for which the denominators $\sigma_j^2 - \sigma_i^2$ become zero and the SVD derivatives are ill defined. Following Ref. [59], we adopt a Lorentzian regularization by defining

$$\frac{1}{\sigma_j^2 - \sigma_i^2} \rightarrow \frac{\sigma_j^2 - \sigma_i^2}{(\sigma_j^2 - \sigma_i^2)^2 + \eta}, \quad \eta > 0. \quad (\text{B10})$$

The forward-mode derivatives are then obtained as

$$\dot{U} = U \Omega_U, \quad \dot{V} = V \Omega_V, \quad \dot{\Sigma} = \text{diag}(P). \quad (\text{B11})$$

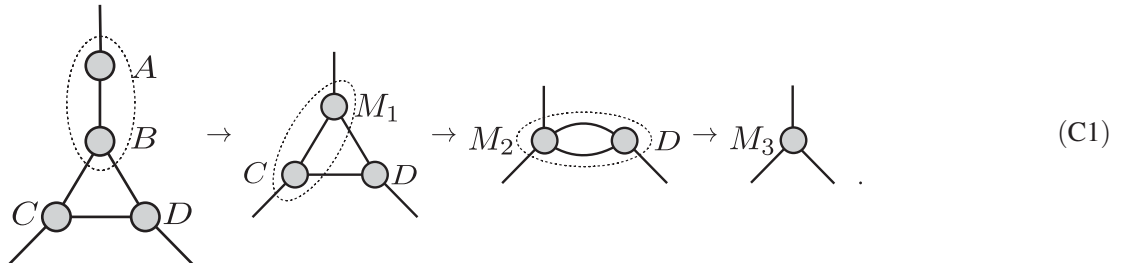
Even when a truncated SVD is used in TRG, the AD must be formulated at the level of the full SVD. This is because the derivatives of the retained vectors mathematically depend on the singular values and vectors of the discarded sector. While Ref. [59] neglected this contribution, Ref. [83] pointed out that the truncated subspace must be correctly included in the derivation. The second-order derivatives are obtained by differentiating the first-order relations once more with respect to β .

APPENDIX C: COMPUTATIONAL COST

1. General case

In this section, we provide details of a contraction-tree-based differentiation method. A contraction tree [73] is a binary tree that specifies the contraction order of a tensor networks. While contraction-tree-based reverse-mode automatic differentiation algorithms are described in Ref. [88], our method provides a general framework to compute arbitrary k th derivatives of tensor networks by forward-mode automatic differentiation.

Let us begin with an example. Consider the following contraction, where the dotted circles indicate contractions between pairs of tensors:



In this contraction, we evaluate the intermediate tensors and their derivatives in a single forward pass along the contraction tree. Let us denote the contraction of two tensors A and B by $\text{contract}(A, B)$. Then, the contraction-tree-based differentiation can be written as

Input: $(A, \dot{A}, \ddot{A}), (B, \dot{B}, \ddot{B}), (C, \dot{C}, \ddot{C}), (D, \dot{D}, \ddot{D})$

$$M_1 = \text{contract}(A, B), \quad (\text{C2})$$

$$\dot{M}_1 = \text{contract}(\dot{A}, B) + \text{contract}(A, \dot{B}), \quad (\text{C3})$$

$$\ddot{M}_1 = \text{contract}(\ddot{A}, B) + 2\text{contract}(\dot{A}, \dot{B}) + \text{contract}(A, \ddot{B}), \quad (\text{C4})$$

$$M_2 = \text{contract}(M_1, C), \quad (\text{C5})$$

$$\dot{M}_2 = \text{contract}(\dot{M}_1, C) + \text{contract}(M_1, \dot{C}), \quad (\text{C6})$$

$$\ddot{M}_2 = \text{contract}(\ddot{M}_1, C) + 2\text{contract}(\dot{M}_1, \dot{C}) + \text{contract}(M_1, \ddot{C}), \quad (\text{C7})$$

$$M_3 = \text{contract}(M_2, D), \quad (\text{C8})$$

$$\dot{M}_3 = \text{contract}(\dot{M}_2, D) + \text{contract}(M_2, \dot{D}), \quad (\text{C9})$$

$$\ddot{M}_3 = \text{contract}(\ddot{M}_2, D) + 2\text{contract}(\dot{M}_2, \dot{D}) + \text{contract}(M_2, \ddot{D}), \quad (\text{C10})$$

Output: $(M_3, \dot{M}_3, \ddot{M}_3)$.

In the above equations, the intermediate tensors are differentiated at each contraction step. A schematic picture of the contraction tree for Eq. (C1) is shown in Fig. 10(a). To evaluate derivatives of this network up to second order, the contraction tree is modified as illustrated in Fig. 10(b). The total cost of evaluating derivatives up to second order is exactly 6 times the usual cost.

It is straightforward to derive general formula. If the tensor network G is initially represented by a set of n tensors,

$$\mathcal{G}^{(0)} = \{A_1^{(0)}, A_2^{(0)}, \dots, A_n^{(0)}\}, \quad (\text{C11})$$

and assume that each tensor has a parameter dependence on β . The full contraction of this tensor network is denoted as

$$G = \text{contract}(A_1^{(0)}, A_2^{(0)}, \dots, A_n^{(0)}) \quad (\text{C12})$$

and can be performed recursively in $n - 1$ steps. At the i th step ($i = 1, \dots, n - 1$), the system transitions from the network $\mathcal{G}^{(i-1)}$ to a new set of $n - i$ tensors:

$$\mathcal{G}^{(i)} = \{A_1^{(i)}, A_2^{(i)}, \dots, A_{n-i}^{(i)}\}. \quad (\text{C13})$$

This transition is defined by selecting two tensors, $A_p^{(i-1)}$ and $A_q^{(i-1)}$, from the previous network $\mathcal{G}^{(i-1)}$ and contracting them to form the new tensor $A_{n-i}^{(i)}$:

$$A_{n-i}^{(i)} = \text{contract}(A_p^{(i-1)}, A_q^{(i-1)}). \quad (\text{C14})$$

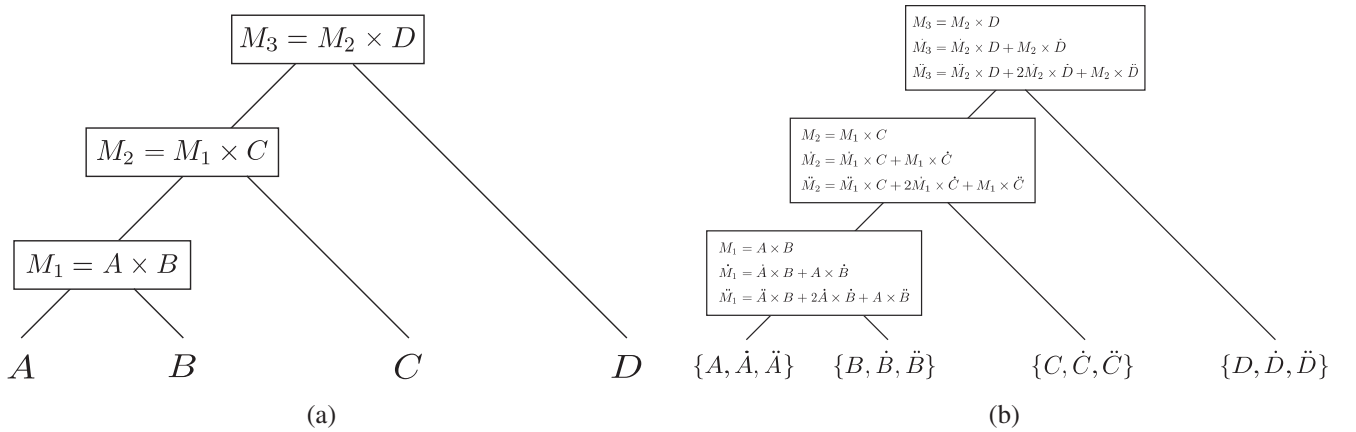


FIG. 10. (a) The contraction tree for Eq. (C1). (b) Modified contraction tree for evaluating derivatives up to second order.

The remaining $n - i - 1$ tensors in $\mathcal{G}^{(i)}$ correspond to the unselected tensors carried over from $\mathcal{G}^{(i-1)}$. Now let us introduce the notation $A[k]$ to represent the k th derivatives of a tensor A , formally defined as

$$A[k] \equiv \overbrace{A}^k. \quad (\text{C15})$$

Assume we have access to the derivatives of the initial tensor set $\mathcal{G}^{(0)}$ up to the k th order. Using the above notation, we define the set of m th derivatives, denoted as $\mathcal{G}^{(0)}[m]$ as follows:

$$\begin{aligned} \mathcal{G}^{(0)}[1] &= \{A_1^{(0)}[1], A_2^{(0)}[1], \dots, A_n^{(0)}[1]\} \\ \mathcal{G}^{(0)}[2] &= \{A_1^{(0)}[2], A_2^{(0)}[2], \dots, A_n^{(0)}[2]\} \\ &\vdots \\ \mathcal{G}^{(0)}[k] &= \{A_1^{(0)}[k], A_2^{(0)}[k], \dots, A_n^{(0)}[k]\}, \end{aligned} \quad (\text{C16})$$

where $A_j^{(0)}[m]$ corresponds to the m th derivative of the j th tensor in the initial set. At each contraction step, calculating the derivatives of the new tensor $A_{n-i}^{(i)}$ follows the general Leibniz rule:

$$\begin{aligned} A_{n-i}^{(i)}[1] &= \text{contract}(A_p^{(i-1)}[1], A_q^{(i-1)}[0]) \\ &\quad + \text{contract}(A_p^{(i-1)}[0], A_q^{(i-1)}[1]), \end{aligned} \quad (\text{C17})$$

$$\begin{aligned} A_{n-i}^{(i)}[2] &= \text{contract}(A_p^{(i-1)}[2], A_q^{(i-1)}[0]) \\ &\quad + 2\text{contract}(A_p^{(i-1)}[1], A_q^{(i-1)}[1]) \\ &\quad + \text{contract}(A_p^{(i-1)}[0], A_q^{(i-1)}[2]), \end{aligned} \quad (\text{C18})$$

$\vdots$

$$A_{n-i}^{(i)}[k] = \sum_{m=0}^k \binom{k}{m} \text{contract}(A_p^{(i-1)}[m], A_q^{(i-1)}[k-m]). \quad (\text{C19})$$

Consequently, after completing all $n - 1$ contraction steps, the network reduces to a single final tensor $A_1^{(n-1)}$. Thus, we obtain the derivatives of the network G for all orders up to k :

$$G[m] = A_1^{(n-1)}[m], \quad m = 0, 1, \dots, k. \quad (\text{C20})$$

The total cost of evaluating up to k th derivatives is exactly $\frac{(k+1)(k+2)}{2}$ times the original contraction.

2. Linear scaling algorithm

The contraction tree based algorithm is itself very efficient in terms of the computational cost; however, a simple modification can be considered, which results in the total computational cost to be $2k + 1$ times original contraction. Consider the second derivative case. We consider contraction in Eqs. (C2)–(C4). Let C be the total cost of evaluating Eq. (C2). The total cost of evaluating contraction in Eqs. (C2)–(C4) is $6C$. The key idea is to take combination of them, as

$$M_1 = \text{contract}(A, B), \quad (\text{C21})$$

$$\begin{aligned} S_+ &= \text{contract}(A + \dot{A}, B + \dot{B}), \\ S_- &= \text{contract}(A - \dot{A}, B - \dot{B}), \end{aligned} \quad (\text{C22})$$

$$\begin{aligned} T_+ &= \text{contract}(A + \ddot{A}, B + \ddot{B}), \\ T_- &= \text{contract}(A - \ddot{A}, B - \ddot{B}). \end{aligned} \quad (\text{C23})$$

We can construct $\dot{M}_1$ and $\ddot{M}_1$ from these tensors:

$$\dot{M}_1 = \frac{S_+ - S_-}{2}, \quad (\text{C24})$$

$$\ddot{M}_1 = \frac{T_+ - T_-}{2} + 2\left(\frac{S_+ + S_-}{2} - M_1\right). \quad (\text{C25})$$

In this scheme, the total cost of the matrix multiplication is $5C$, slightly smaller than $6C$.

This can be generalized in terms of the Taylor-mode AD [89]. Suppose we want to evaluate up to k th derivatives of M_1 . We define Taylor mode for A and B :

$$A(t) = \sum_{m=0}^k \frac{A[m]}{m!} t^m, \quad (\text{C26})$$

$$B(t) = \sum_{m=0}^k \frac{B[m]}{m!} t^m. \quad (\text{C27})$$

Then the product

$$W(t) = \text{contract}(A(t), B(t)) \quad (\text{C28})$$

is written as

$$W(t) = \sum_{m=0}^k \frac{M_1[m]}{m!} t^m + \sum_{m=k+1}^{2k} \frac{R_1[m]}{m!} t^m. \quad (\text{C29})$$

Thus, $W(t)$ serves as a generating function for $\{M_1[m]\}_{m=0}^k$, with additional coefficients $\{R_1[m]\}_{m=k+1}^{2k}$ that are not used in the computation of derivatives up to the order of k . Since $W(t)$ is a polynomial of degree at most $2k$,

it can be uniquely reconstructed by polynomial interpolation by choosing $2k + 1$ distinct points. In this scheme, the total cost of the matrix multiplications is $(2k + 1)C$, since we need to compute distinct points of W at $2k + 1$ points. Note that this interpolation scheme introduces an additional

overhead from tensor additions and scalar multiplications, although the number of dominant contractions is reduced, which could be even slower than contraction-tree-based differentiation at small k .

-
- [1] M. Levin and C. P. Nave, *Phys. Rev. Lett.* **99**, 120601 (2007).
 - [2] Z.-C. Gu and X.-G. Wen, *Phys. Rev. B* **80**, 155131 (2009).
 - [3] Z. Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang, *Phys. Rev. B* **86**, 045139 (2012).
 - [4] Y. Shimizu and Y. Kuramashi, *Phys. Rev. D* **90**, 074503 (2014).
 - [5] J. F. Unmuth-Yockey, Y. Meurice, J. Osborn, and H. Zou, *Proc. Sci. LATTICE2014* (**2014**) 325 [arXiv:1411.4213].
 - [6] J. F. Yu, Z. Y. Xie, Y. Meurice, Y. Liu, A. Denbleyker, H. Zou, M. P. Qin, and J. Chen, *Phys. Rev. E* **89**, 013308 (2014).
 - [7] S. Wang, Z. Y. Xie, J. Chen, B. Normand, and T. Xiang, *Chin. Phys. Lett.* **31**, 070503 (2014).
 - [8] Y. Shimizu and Y. Kuramashi, *Phys. Rev. D* **90**, 014508 (2014).
 - [9] S. Takeda and Y. Yoshimura, *Prog. Theor. Exp. Phys.* **2015**, 043B01 (2015).
 - [10] S. Takeda and Y. Yoshimura, *Prog. Theor. Exp. Phys.* **2015**, 043B01 (2015).
 - [11] H. Kawauchi and S. Takeda, *Phys. Rev. D* **93**, 114503 (2016).
 - [12] R. Sakai, S. Takeda, and Y. Yoshimura, *Prog. Theor. Exp. Phys.* **2017**, 063B07 (2017).
 - [13] R. Sakai, D. Kadoh, Y. Kuramashi, Y. Nakamura, S. Takeda, and Y. Yoshimura, *EPJ Web Conf.* **175**, 11019 (2018).
 - [14] Y. Yoshimura, Y. Kuramashi, Y. Nakamura, S. Takeda, and R. Sakai, *Phys. Rev. D* **97**, 054511 (2018).
 - [15] A. Bazavov, S. Catterall, R. G. Jha, and J. Unmuth-Yockey, *Phys. Rev. D* **99**, 114507 (2019).
 - [16] S. Iino, S. Morita, and N. Kawashima, *Phys. Rev. B* **100**, 035449 (2019).
 - [17] S. Hong and D.-H. Kim, *Phys. Rev. E* **101**, 012124 (2020).
 - [18] S. Akiyama, Y. Kuramashi, T. Yamashita, and Y. Yoshimura, *Phys. Rev. D* **100**, 054510 (2019).
 - [19] S. Akiyama, D. Kadoh, Y. Kuramashi, T. Yamashita, and Y. Yoshimura, *J. High Energy Phys.* **09** (2020) 177.
 - [20] C. Delcamp and A. Tilloy, *Phys. Rev. Res.* **2**, 033278 (2020).
 - [21] S. Akiyama and Y. Kuramashi, *Phys. Rev. D* **104**, 014504 (2021).
 - [22] S. Akiyama, Y. Kuramashi, T. Yamashita, and Y. Yoshimura, *J. High Energy Phys.* **01** (2021) 121.
 - [23] S. Akiyama, Y. Kuramashi, and Y. Yoshimura, *Phys. Rev. D* **104**, 034507 (2021).
 - [24] J. Bloch, R. G. Jha, R. Lohmayer, and M. Meister, *Phys. Rev. D* **104**, 094517 (2021).
 - [25] M. Fukuma, D. Kadoh, and N. Matsumoto, *Prog. Theor. Exp. Phys.* **2021**, 123B03 (2021).
 - [26] S. Akiyama and D. Kadoh, *J. High Energy Phys.* **10** (2021) 188.
 - [27] M. Hirasawa, A. Matsumoto, J. Nishimura, and A. Yosprakob, *J. High Energy Phys.* **12** (2021) 011.
 - [28] S. Akiyama, Y. Kuramashi, and T. Yamashita, *Prog. Theor. Exp. Phys.* **2022**, 023I01 (2022).
 - [29] G. Li, K. H. Pai, and Z.-C. Gu, *Phys. Rev. Res.* **4**, 023159 (2022).
 - [30] K. Nakayama, L. Funcke, K. Jansen, Y.-J. Kao, and S. Kühn, *Phys. Rev. D* **105**, 054507 (2022).
 - [31] S. Akiyama and Y. Kuramashi, *J. High Energy Phys.* **05** (2022) 102.
 - [32] J. Bloch, R. Lohmayer, and M. Meister, *Proc. Sci. LATTICE2021* (**2022**) 285.
 - [33] T. Kuwahara and A. Tsuchiya, *Prog. Theor. Exp. Phys.* **2022**, 093B02 (2022).
 - [34] X. Luo and Y. Kuramashi, *Phys. Rev. D* **107**, 094509 (2023).
 - [35] J. Bloch and R. Lohmayer, *Nucl. Phys. B* **986**, 116032 (2023).
 - [36] M. Asaduzzaman, S. Catterall, Y. Meurice, R. Sakai, and G. C. Toga, *J. High Energy Phys.* **01** (2023) 024.
 - [37] K. Yoshiyama and K. Hukushima, *Phys. Rev. E* **108**, 054124 (2023).
 - [38] J. Genzor, *Phys. Rev. E* **107**, 034131 (2023).
 - [39] S. Akiyama and Y. Kuramashi, *J. High Energy Phys.* **10** (2023) 77.
 - [40] S. Akiyama, R. G. Jha, and J. Unmuth-Yockey, *Phys. Rev. D* **110**, 034519 (2024).
 - [41] S. Akiyama and Y. Kuramashi, *J. High Energy Phys.* **09** (2024) 086.
 - [42] K. Nakayama and M. Schneider, *Phys. Rev. D* **110**, 094501 (2024).
 - [43] A. Samlodia, V. Longia, R. G. Jha, and A. Joseph, *Phys. Rev. D* **110**, 034504 (2024).
 - [44] K. Homma, S. Morita, and N. Kawashima, *Phys. Rev. B* **111**, 134427 (2025).
 - [45] Y. Sugimoto, S. Akiyama, and Y. Kuramashi, *Phys. Rev. D* **113**, 034503 (2026).
 - [46] H. Aizawa, S. Takeda, and Y. Yoshimura, *Prog. Theor. Exp. Phys.* **2026**, 013B07 (2026).
 - [47] H. Kanno, S. Akiyama, K. Murakami, and S. Takeda, *J. High Energy Phys.* **11** (2025) 036.
 - [48] S. Morita, R. Igarashi, H.-H. Zhao, and N. Kawashima, *Phys. Rev. E* **97**, 033310 (2018).
 - [49] D. Adachi, T. Okubo, and S. Todo, *Phys. Rev. B* **102**, 054432 (2020).
 - [50] D. Kadoh and K. Nakayama, *arXiv:1912.02414*.
 - [51] D. Adachi, T. Okubo, and S. Todo, *Phys. Rev. B* **105**, L060402 (2022).

- [52] G. Evenbly and G. Vidal, *Phys. Rev. Lett.* **115**, 180405 (2015).
- [53] S. Yang, Z.-C. Gu, and X.-G. Wen, *Phys. Rev. Lett.* **118**, 110504 (2017).
- [54] M. Hauru, C. Delcamp, and S. Mizera, *Phys. Rev. B* **97**, 045111 (2018).
- [55] S. Morita and N. Kawashima, *Phys. Rev. B* **103**, 045131 (2021).
- [56] K. Homma, T. Okubo, and N. Kawashima, *Phys. Rev. Res.* **6**, 043102 (2024).
- [57] F.-F. Song and N. Kawashima, *Phys. Rev. B* **112**, 235167 (2025).
- [58] S. Morita and N. Kawashima, *Comput. Phys. Commun.* **236**, 65 (2019).
- [59] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, *Phys. Rev. X* **9**, 031041 (2019).
- [60] S. Morita and N. Kawashima, *Phys. Rev. B* **111**, 054433 (2025).
- [61] M. Bartholomew-Biggs, S. Brown, B. Christianson, and L. Dixon, *J. Comput. Appl. Math.* **124**, 171 (2000).
- [62] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *J. Mach. Learn. Res.* **18**, 1 (2018), <https://www.jmlr.org/papers/v18/17-468.html>.
- [63] B.-B. Chen, Y. Gao, Y.-B. Guo, Y. Liu, H.-H. Zhao, H.-J. Liao, L. Wang, T. Xiang, W. Li, and Z. Y. Xie, *Phys. Rev. B* **101**, 220409 (2020).
- [64] C. Geng, H.-Y. Hu, and Y. Zou, *Mach. Learn. Sci. Technol.* **3**, 015020 (2022).
- [65] Y.-H. Tseng, S.-W. Li, and F.-J. Jiang, *Prog. Theor. Exp. Phys.* **2025**, 073A01 (2025).
- [66] <https://github.com/wangleiphy/tensorgrad> (2021).
- [67] <https://github.com/under-Peter/TensorNetworkAD.jl> (2023).
- [68] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimeshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, USA, 2019), Vol. 32, <https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>.
- [69] M. Innes, [arXiv:1810.07951](https://arxiv.org/abs/1810.07951).
- [70] F. I. Az-zahra, S. Takeda, and T. Yamazaki, *Phys. Rev. D* **110**, 034514 (2024).
- [71] L. Wang and F. Verstraete, [arXiv:1110.4362](https://arxiv.org/abs/1110.4362).
- [72] S. Iino, S. Morita, and N. Kawashima, *Phys. Rev. B* **101**, 155418 (2020).
- [73] G. Evenbly and R. N. C. Pfeifer, *Phys. Rev. B* **89**, 245118 (2014).
- [74] K. Nakayama and M. Schneider, *Phys. Rev. D* **110**, 094501 (2024).
- [75] G. Evenbly, *Phys. Rev. B* **95**, 045117 (2017).
- [76] Y. Nakamura, H. Oba, and S. Takeda, *Phys. Rev. B* **99**, 155101 (2019).
- [77] <https://datarepo.mdcl.issp.u-tokyo.ac.jp/repo/49> (2026).
- [78] S. Morita and N. Kawashima, *J. Phys. Soc. Jpn.* **95**, 044001 (2026).
- [79] S. Akiyama, R. G. Jha, J. Maeda, Y. Tanizaki, and J. Unmuth-Yockey, *Phys. Rev. D* **113**, 074502 (2026).
- [80] A. Ueda and M. Oshikawa, *Phys. Rev. B* **108**, 024413 (2023).
- [81] F. Kos, D. Poland, D. Simmons-Duffin, and A. Vichi, *J. High Energy Phys.* **08** (2016) 036.
- [82] S. R. White, *Phys. Rev. B* **48**, 10345 (1993).
- [83] A. Francuz, N. Schuch, and B. Vanhecke, *Phys. Rev. Res.* **7**, 013237 (2025).
- [84] L. Devos and J. Haegeman, [arXiv:2508.10076](https://arxiv.org/abs/2508.10076).
- [85] A. Ueda, S. D. Meyer, A. Naravane, V. Vanthilt, and F. Verstraete, [arXiv:2508.05406](https://arxiv.org/abs/2508.05406).
- [86] K. Nakayama, [arXiv:2307.14191](https://arxiv.org/abs/2307.14191).
- [87] J. Townsend, Differentiating the singular value decomposition, technical report, 2016, <https://j-towns.github.io/papers/svd-derivative.pdf>, 2016.
- [88] A. A. Gorodetsky, C. Safta, and J. D. Jakeman, *J. Mach. Learn. Res.* **23**, 1 (2022), <https://www.jmlr.org/papers/v23/21-0225.html>.
- [89] J. Bettencourt, M. J. Johnson, and D. Duvenaud, in *Program Transformations for ML Workshop at NeurIPS 2019* (2019), <https://openreview.net/forum?id=SkxEF3FNPH>.