Quantized Tensor Networks for the Vlasov-Maxwell Equations: Supplementary Information

I. EXAMPLES OF QUANTIZED TENSOR TRAINS

As stated in the main text, the quantized tensor train format is

$$h_i = h(x_i) \cong h(i_1, i_2, ..., i_L) = \sum_{\alpha_1=1}^{r_1} ... \sum_{\alpha_{L-1}=1}^{r_{L-1}} M^{(1)}_{\alpha_1}(i_1) M^{(2)}_{\alpha_1,\alpha_2}(i_2) \dots M^{(L)}_{\alpha_{L-1}}(i_L)\,, \tag{1}$$

and the quantized tensor train operator (QTT-Operator) format is

$$O(x'_i, x_i) \cong O((o_1, o_2, ..., o_L), (i_1, i_2, ..., i_L)) = \sum_{\alpha_1=1}^{r_1} ... \sum_{\alpha_{L-1}=1}^{r_{L-1}} O^{(1)}_{\alpha_1}(o_1, i_1) O^{(2)}_{\alpha_1,\alpha_2}(o_2.i_2) \dots O^{(L)}_{\alpha_{L-1}}(o_L, i_L)\,. \tag{2}$$

A. QTT Operators for the Vlasov Equation

In this section, we explicitly write the QTT representations of the operators used in the Vlasov equation.

1. *QTT-Operator for first derivative*

The QTT-Operator for the first derivative along one axis can be written explicitly, assuming uniform grid points and a finite difference scheme. With periodic boundary conditions and a centered second-order stencil, it is

$$\frac{\partial}{\partial x} \approx \frac{1}{\Delta x} \begin{bmatrix} \mathbb{I} & S^+ + S^- & S^+ + S^- \end{bmatrix} \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \dots \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \begin{bmatrix} 0.5(S^- - S^+) \\ 0.5S^+ \\ -0.5S^- \end{bmatrix}, \tag{3}$$

where

$$\mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad S^+ = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \qquad S^- = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

In this notation, the matrices from left to right are the tensors $O^{(1)}$ to $O^{(L)}$ in Eq. (2). For the p^{th} tensor, the dimensions of the blocks are labeled by o_p and i_p, while the dimensions of the explicitly written matrices are the virtual bonds α_{p-1} and α_p.

2. *Application of QTT-Operators to QTTs*

Example 1. As a brief aside, consider a system with 8 grid points and suppose our vector of interest is all ones. The corresponding QTT is rank 1, and given by

$$h = [1,1,1,1,1,1,1,1] \cong \mathbf{1} \otimes \mathbf{1} \otimes \mathbf{1} = \begin{bmatrix} \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{1} \end{bmatrix} \qquad \text{where } \mathbf{1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Applying the finite difference operator in Eq. (3) to this QTT, we obtain the following QTT:

$$\frac{\partial}{\partial x} h \approx \frac{1}{\Delta x} \begin{bmatrix} \mathbb{I}\mathbf{1} & (S^+ + S^-)\mathbf{1} & (S^+ + S^-)\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbb{I}\mathbf{1} & S^-\mathbf{1} & S^+\mathbf{1} \\ 0 & S^+\mathbf{1} & 0 \\ 0 & 0 & S^-\mathbf{1} \end{bmatrix} \begin{bmatrix} 0.5(S^- - S^+)\mathbf{1} \\ 0.5S^+\mathbf{1} \\ -0.5S^-\mathbf{1} \end{bmatrix}. \tag{4}$$

By combining blocks that are the same within each matrix, the QTT can be simplified to

$$\frac{\partial}{\partial x}h \approx 0.5\begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{1} & \begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix}\\ 0 & \begin{bmatrix}0\\1\end{bmatrix} & 0\\ 0 & 0 & \begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix} & \begin{bmatrix}-1\\0\end{bmatrix}\end{bmatrix} = 0.5\begin{bmatrix}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\\ \begin{bmatrix}0\\1\end{bmatrix}\\ \begin{bmatrix}-1\\0\end{bmatrix}\end{bmatrix} = 0.5\begin{bmatrix}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{1}\end{bmatrix}\begin{bmatrix}0\end{bmatrix} = 0\,, \tag{5}$$

which is the expected result.

Example 2. Now consider the vector $h = [-4, -3, -2, -1, 0, 1, 2, 3]$. There exists an analytical expression for QTTs representing the linear function $av + b$ on a uniform grid. Let the grid be defined along the interval $[-d/2, d/2)$, such that the i^{th} grid point is $v_i = -d/2 + di/N$. For a grid of $N = 2^L$ grid points ($\vec{v}$), the QTT of length L is

$$a\vec{v} + b = \begin{bmatrix}\mathbf{1} & 2^{-1}ad\vec{n} + (-ad/2 + b)\,\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{1} & 2^{-2}ad\vec{n}\\ 0 & \mathbf{1}\end{bmatrix}\cdots\begin{bmatrix}\mathbf{1} & 2^{-(L-1)}ad\vec{n}\\ 0 & \mathbf{1}\end{bmatrix}\begin{bmatrix}2^{-L}ad\vec{n}\\ \mathbf{1}\end{bmatrix} \qquad \text{where} \qquad \vec{n} = \begin{bmatrix}0\\1\end{bmatrix}. \tag{6}$$

Thus, taking $L = 3$ and plugging in $a = 1$, $b = 0$, and $d = 8$ into Eq. (6), we arrive at

$$h = [-4, -3, -2, -1, 0, 1, 2, 3] \cong \begin{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} & \begin{bmatrix}-4\\0\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} & \begin{bmatrix}0\\2\end{bmatrix}\\ 0 & \begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\\ \begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix}.$$

After applying the finite difference operator in Eq. (3), we obtain the QTT (rank 6, with 3 tensor cores)

$$\begin{aligned}\frac{\partial}{\partial x}h \approx{}& \frac{1}{\Delta x}\begin{bmatrix}\begin{bmatrix}\mathbb{I}\begin{bmatrix}1\\1\end{bmatrix} & (S^+ + S^-)\begin{bmatrix}1\\1\end{bmatrix} & (S^+ + S^-)\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix} & \begin{bmatrix}\mathbb{I}\begin{bmatrix}-4\\0\end{bmatrix} & (S^+ + S^-)\begin{bmatrix}-4\\0\end{bmatrix} & (S^+ + S^-)\begin{bmatrix}-4\\0\end{bmatrix}\end{bmatrix}\end{bmatrix}\times\\
&\begin{bmatrix}\begin{bmatrix}\mathbb{I}\begin{bmatrix}1\\1\end{bmatrix} & S^-\begin{bmatrix}1\\1\end{bmatrix} & S^+\begin{bmatrix}1\\1\end{bmatrix}\\ 0 & S^+\begin{bmatrix}1\\1\end{bmatrix} & 0\\ 0 & 0 & S^-\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix} & \begin{bmatrix}\mathbb{I}\begin{bmatrix}0\\2\end{bmatrix} & S^-\begin{bmatrix}0\\2\end{bmatrix} & S^+\begin{bmatrix}0\\2\end{bmatrix}\\ 0 & S^+\begin{bmatrix}0\\2\end{bmatrix} & 0\\ 0 & 0 & S^-\begin{bmatrix}0\\2\end{bmatrix}\end{bmatrix}\\ 0 & \begin{bmatrix}\mathbb{I}\begin{bmatrix}1\\1\end{bmatrix} & S^-\begin{bmatrix}1\\1\end{bmatrix} & S^+\begin{bmatrix}1\\1\end{bmatrix}\\ 0 & S^+\begin{bmatrix}1\\1\end{bmatrix} & 0\\ 0 & 0 & S^-\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix}\end{bmatrix}\times\begin{bmatrix}\begin{bmatrix}0.5(S^- - S^+)\begin{bmatrix}0\\1\end{bmatrix}\\ 0.5S^+\begin{bmatrix}0\\1\end{bmatrix}\\ -0.5S^-\begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix}\\ \begin{bmatrix}0.5(S^- - S^+)\begin{bmatrix}1\\1\end{bmatrix}\\ 0.5S^+\begin{bmatrix}1\\1\end{bmatrix}\\ -0.5S^-\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix}\end{bmatrix}\\
={}& 0.5\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix} & \begin{bmatrix}\begin{bmatrix}-4\\0\end{bmatrix} & \begin{bmatrix}0\\-4\end{bmatrix}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\end{bmatrix} & \begin{bmatrix}\begin{bmatrix}0\\2\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix}\end{bmatrix}\\ 0 & \begin{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} & \begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix}\\ 0 & \begin{bmatrix}0\\1\end{bmatrix} & \begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}0\end{bmatrix}\\ \begin{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\\ \begin{bmatrix}0\\1\end{bmatrix}\\ \begin{bmatrix}-1\\0\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{aligned}$$

where we use the same simplification techniques as before. Now, removing the zero terms and using the fact that

$\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$, we can further simplify the QTT and obtain the expected result:

$$\begin{aligned}
\ldots &= 0.5 \begin{bmatrix}\begin{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix} & \begin{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix}\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}\begin{bmatrix}0\\2\end{bmatrix} & \begin{bmatrix}0\\2\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix}\\ \begin{bmatrix}0\\2\end{bmatrix} & \begin{bmatrix}0\\2\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix}\end{bmatrix}\\
\begin{bmatrix}\begin{bmatrix}-4\\-4\end{bmatrix} & \begin{bmatrix}-4\\-4\end{bmatrix} & \begin{bmatrix}-4\\0\end{bmatrix} & \begin{bmatrix}0\\-4\end{bmatrix}\\ 0 & 0 & \begin{bmatrix}0\\-4\end{bmatrix} & \begin{bmatrix}-4\\0\end{bmatrix}\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\\ \begin{bmatrix}0\\-1\end{bmatrix}\end{bmatrix}\\
\begin{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\\ \begin{bmatrix}-1\\0\end{bmatrix}\end{bmatrix}
\end{bmatrix}\\
&= 0.5 \begin{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}0\\1\end{bmatrix}\end{bmatrix}
\begin{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} & \begin{bmatrix}6\\-2\end{bmatrix}\\ \begin{bmatrix}2\\-6\end{bmatrix} & \begin{bmatrix}-2\\-2\end{bmatrix}\end{bmatrix}
\begin{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\\ \begin{bmatrix}-1\\0\end{bmatrix}\end{bmatrix} = [-3, 1, 1, 1, 1, 1, 1, -3] \qquad (7)
\end{aligned}$$

3. Element-wise Multiplication

The Vlasov equation also requires performing elemental multiplication of generic functions $h_1(x)h_2(x)$. This computation is performed by writing $h_1(x)$ as a diagonal matrix and $h_2(x)$ as a vector, and then performing matrix-vector multiplication.

For the spatial advection term, we require diagonalizing $h_1(v) = v$. Analogous to Eq. (6), a matrix with $a\vec{v} + b$ along its diagonal (assuming a uniform grid along the interval $[-d/2, d/2)$) can be written as an QTT-Operator of bond dimension 2. For a a grid with $N = 2^L$ grid points, we obtain the QTT-Operator of length L,

$$\mathrm{diag}(a\vec{v} + b) = \begin{bmatrix}\mathbb{I} & 2^{-1}ad\hat{n} + (-ad/2 + b)\,\mathbb{I}\end{bmatrix} \begin{bmatrix}\mathbb{I} & 2^{-2}ad\hat{n}\\ 0 & \mathbb{I}\end{bmatrix} \ldots \begin{bmatrix}\mathbb{I} & 2^{-(L-1)}ad\hat{n}\\ 0 & \mathbb{I}\end{bmatrix} \begin{bmatrix}2^{-L}ad\hat{n}\\ \mathbb{I}\end{bmatrix} \quad \text{where} \quad \hat{n} = \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}. \quad (8)$$

For the velocity advection term, we require performing elemental multiplication of the distribution function f with components of the Lorentz force F. The force must be computed at each time step so we do not have an analytical form. But, since F is already represented in the QTT format, the tensors of the QTT-Operator representing $\mathrm{diag}(F)$ are simply

$$O^{(p)}_{\alpha_{p-1},\alpha_p}(o_p, i_p) = \sum_{x_p \in \{0,1\}} M^{(p)}_{\alpha_{p-1},\alpha_p}(x_p)\,\delta_{x_p,i_p,o_p}\,, \qquad (9)$$

where $M^{(p)}$ is the p^{th} tensor in the original QTT, and δ_{ijk} as the multi-index Kronecker delta function.

4. Multi-dimensional operators

While the operators in the above discussion act only on one dimension, they can still be represented in a multi-dimensional space. To do this, one simply pads the original QTT with a tensor train of identity matrices for the remaining dimensions. For example, for a 2-D system with dimensions along x and v, if a sequential ordering of the tensors is used, the QTT representation of operator $O : f(x,v) \mapsto \frac{\partial}{\partial x} f(x,v)$ would look like

$$\frac{\partial}{\partial x} \otimes \mathbb{I}_v = \underbrace{\begin{bmatrix}\mathbb{I} & S^+ + S^- & S^+ + S^-\end{bmatrix} \begin{bmatrix}\mathbb{I} & S^- & S^+\\ 0 & S^+ & 0\\ 0 & 0 & S^-\end{bmatrix} \begin{bmatrix}\mathbb{I} & S^- & S^+\\ 0 & S^+ & 0\\ 0 & 0 & S^-\end{bmatrix} \ldots \begin{bmatrix}\mathbb{I} & S^- & S^+\\ 0 & S^+ & 0\\ 0 & 0 & S^-\end{bmatrix} \begin{bmatrix}0.5(S^- - S^+)\\ 0.5S^+\\ -0.5S^-\end{bmatrix}}_{x\text{-axis tensors}} \underbrace{[\mathbb{I}] \ldots [\mathbb{I}]}_{y\text{-axis tensors}}, \qquad (10)$$

where $\mathbb{I}_v$ is the identity operator acting on dimension v. (The multidimensional QTC is constructed similarly, except one must now also define the spine tensors to be of size $1 \times 1 \times 1$ and value 1.)

Similarly, the representation of operator $O : f(x,v) \mapsto vf(x,v)$ is

$$\mathbb{I}_x \otimes \mathrm{diag}(v) = \underbrace{[\mathbb{I}] \ldots [\mathbb{I}]}_{x\text{-axis tensors}} \underbrace{\begin{bmatrix}\mathbb{I} & v_{\max}(\hat{n} - \mathbb{I})\end{bmatrix} \begin{bmatrix}\mathbb{I} & 2^{-1}v_{\max}\hat{n}\\ 0 & \mathbb{I}\end{bmatrix} \ldots \begin{bmatrix}2^{-L+1}v_{\max}\hat{n}\\ \mathbb{I}\end{bmatrix}}_{y\text{-axis tensors}}, \qquad (11)$$

where we take the grid points along the v-axis to be uniformly spaced along the interval $[-v_{\max}, v_{\max})$.

These 1-D operators can be combined via the tensor train analog of matrix multiplication. For example, to obtain operator $O : f(x,v) \mapsto v\frac{\partial}{\partial x} f(x,v)$, one would apply the QTT-operator defined in Eq. (10) to that defined in Eq. (11) (or vice versa; the operators commute in this case), yielding

$$\frac{\partial}{\partial x} \otimes \mathrm{diag}(v) = \underbrace{\begin{bmatrix} \mathbb{I} & S^+ + S^- & S^+ + S^- \end{bmatrix} \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \dots \begin{bmatrix} \mathbb{I} & S^- & S^+ \\ 0 & S^+ & 0 \\ 0 & 0 & S^- \end{bmatrix} \begin{bmatrix} 0.5(S^- - S^+) \\ 0.5S^+ \\ -0.5S^- \end{bmatrix}}_{x\text{-axis tensors}}$$

$$\otimes \underbrace{\begin{bmatrix} \mathbb{I} & v_{\max}(\hat{n} - \mathbb{I}) \end{bmatrix} \begin{bmatrix} \mathbb{I} & 2^{-1} v_{\max}\hat{n} \\ 0 & \mathbb{I} \end{bmatrix} \dots \begin{bmatrix} 2^{-L+1} v_{\max}\hat{n} \\ \mathbb{I} \end{bmatrix}}_{y\text{-axis tensors}} . \tag{12}$$

II. RELEVANT BASIC CONCEPTS IN TENSOR NETWORKS

In this section, we will introduce key tensor network concepts relevant for this work. This is not meant to be a thorough introduction; we instead refer the reader to other works [1–3]. Throughout the supplementary information, we will use the same notation introduced in the main paper to describe discretized functions represented as quantized tensor trains (QTTs)

$$h_i = h(x_i) \cong h(i_1, i_2, ..., i_L) = \sum_{\alpha_1=1}^{r_1} ... \sum_{\alpha_{L-1}=1}^{r_{L-1}} H^{(1)}_{\alpha_1}(i_1) H^{(2)}_{\alpha_1,\alpha_2}(i_2) \dots H^{(L)}_{\alpha_{L-1}}(i_L) \,, \tag{13}$$

or as quantized tree tensor networks with a comb geometry (QTCs) (see Fig. 1(c) of the main text)

$$\begin{aligned} f_{i^{(1)},i^{(2)},\dots,i^{(K)}} &= f(x_{1,i^{(1)}}, x_{2,i^{(2)}}, \dots, x_{K,i^{(K)}}) \\ &\cong f(i_1^{(1)}, \dots, i_L^{(1)}, i_1^{(2)}, \dots, i_L^{(2)}, \dots, i_1^{(K)}, \dots, i_L^{(K)}) = \\ &= \sum_{\gamma_1=1}^{r_1} ... \sum_{\gamma_{K-1}=1}^{r_{K-1}} B^{(1)}_{\gamma_1}(i_1^{(1)}, ..., i_L^{(1)})\, B^{(2)}_{\gamma_1,\gamma_2}(i_1^{(2)}, ..., i_L^{(2)}) \dots B^{(K)}_{\gamma_{L-1}}(i_1^{(K)}, ..., i^{(K)}) \,, \end{aligned} \tag{14}$$

where $B^{(k)}$ is a QTT for the k^{th} dimension

$$B^{(k)}_{\gamma_k,\gamma_{k+1}}(i_1^{(k)}, ..., i_L^{(k)}) = \sum_{\beta_k} S^{(k)}_{\gamma_k,\gamma_{k+1},\beta_k} \left(\sum_{\alpha_{(k,1)}} ... \sum_{\alpha_{(k,L-1)}} \tilde{M}^{(k;1)}_{\beta_k,\alpha_{(k,1)}}(i_1^{(k)})\, M^{(k;2)}_{\alpha_{(k,1)},\alpha_{(k,2)}}(i_2^{(k)}) \dots M^{(k;L)}_{\alpha_{(k,L-1)}}(i_L^{(k)}) \right) . \tag{15}$$

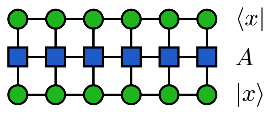
The respective quantized tensor networks (QTNs) for operators (QTT-Os and QTC-Os) have the same form, but with two physical indices (e.g., o, i) at each tensor instead of just one (denoted by i in the above equations).

A. Tensor Network Differentiation

The working principle behind most local tensor network algorithms involve updating a single tensor or a pair of neighboring tensors in a sequential fashion. For optimization problems (e.g. finding extreme eigenvalues and solving linear equations), this involves minimizing the cost function (a scalar value) assuming all but one tensor (or pair of tensors) remain fixed. This requires taking the derivative of the cost function with respect to the complex conjugate of that tensor.

Consider the expectation value $\langle x|A|x\rangle$,

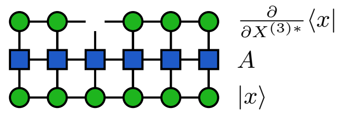
$\langle x|A|x\rangle$ =



.

Let $X^{(i)}$ denote the ith tensor in the QTT for x. Because the tensor network depends linearly on $X^{(i)*}$, (treating $X^{(i)}$ and its complex conjugate as independent), taking the derivative with respect to $X^{(i)*}$ amounts to removing that tensor from the tensor network. For example, for $i = 3$,

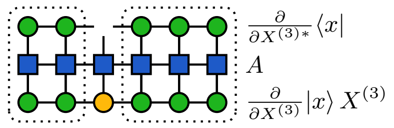
$$\frac{\partial}{\partial X^{(3)*}} \langle x|A|x\rangle \quad = $$



.

An equivalent representation is

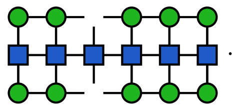
$$\left(\frac{\partial}{\partial X^{(3)*}} \frac{\partial}{\partial X^{(3)}} \langle x|A|x\rangle\right) X^{(3)} \quad = \quad A_{\text{eff}} X^{(3)} \quad = $$


$\frac{\partial}{\partial X^{(3)*}}\langle x|$
A
$\frac{\partial}{\partial X^{(3)}}|x\rangle\, X^{(3)}$

.

In the figure, $X^{(3)}$ is highlighted in orange. The expression in the parentheses can be considered an effective operator for the reduced problem;

$$A_{\text{eff}} \quad = $$



.

Also note that though the figures depict the tensor networks for the tensor train geometry, the same concept holds for the comb geometry. If the portions of the tensor network in the dashed boxes are precomputed (see the following section), then the cost of computing A_{eff} scales like $\mathcal{O}(D^2 D_W^2 d^2 + D^4 D_W d^2)$, where D and D_W are the bond dimensions of x and A, respectively. The cost of computing $A_{\text{eff}} X$ is actually cheaper, scaling like $\mathcal{O}(2D^3 D_W d + D^2 D_W^2 d^2)$.

B. Environment tensors

Computing expectation values and overlaps require performing a series of tensor contractons. While the most efficient contract pattern may vary depending on the size of each bond, building environment tensors is often most practical since they are used in multiple steps of most algorithms.

1. *Tensor train geometry*

Let us consider tensor trains of length L. In this case, the left or right environments for $\langle b|A|x\rangle$ are (see Fig 1(a) and (b))

$$(E_L^{(p)})_{\alpha_p^b,\alpha_p^A,\alpha_p^x} = \sum_{\alpha_{p-1}^b,\alpha_{p-1}^A,\alpha_{p-1}^x} \sum_{o_p,i_p} B^{(p)*}_{\alpha_{p-1}^b,\alpha_p^b}(o_p) A^{(p)}_{\alpha_{p-1}^A,\alpha_p^A}(o_p,i_p) X^{(p)}_{\alpha_{p-1}^x,\alpha_p^x}(i_p) (E_L^{(p-1)})_{\alpha_{p-1}^b,\alpha_{p-1}^A,\alpha_{p-1}^x}\,, \tag{16}$$

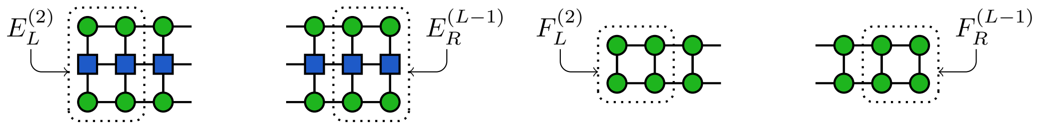
$$(E_R^{(p)})_{\alpha_{p-1}^b,\alpha_{p-1}^A,\alpha_{p-1}^x} = \sum_{\alpha_p^b,\alpha_p^A,\alpha_p^x} \sum_{o_p,i_p} B^{(p)*}_{\alpha_{p-1}^b,\alpha_p^b}(o_p) A^{(p)}_{\alpha_{p-1}^A,\alpha_p^A}(o_p,i_p) X^{(p)}_{\alpha_{p-1}^x,\alpha_p^x}(i_p) (E_R^{(p+1)})_{\alpha_p^b,\alpha_p^A,\alpha_p^x}\,, \tag{17}$$

where p denotes the site of the tensor in the tensor train and B, A and X denote tensors in the QTTs for b, A, and x, respectively. In essence, the left/right environments at site p are the left/right half of the tensor network for $\langle b|A|x\rangle$, including the tensors at site p. If the bond dimensions of x, b, and A are D_x, D_b, and D_A, respectively, and they all have physical bonds of size d, then the cost of building the environment scales like $\mathcal{O}((D_x^2 D_b + D_b^2 D_x) D_A d + D_x D_b D_A^2 d^2)$. Similarly, the left and right environments for $\langle b|x\rangle$ are (see Fig 1(c) and (d))

$$(F_L^{(p)})_{\alpha_p^b,\alpha_p^x} = \sum_{\alpha_{p-1}^b,\alpha_{p-1}^x} \sum_{i_p} B^{(p)*}_{\alpha_{p-1}^b,\alpha_p^b}(i_p) X^{(p)}_{\alpha_{p-1}^x,\alpha_p^x}(i_p) (F_L^{(p-1)})_{\alpha_{p-1}^b,\alpha_{p-1}^x} \tag{18}$$

$$(F_R^{(p)})_{\alpha_{p-1}^b,\alpha_{p-1}^x} = \sum_{\alpha_p^b,\alpha_p^x} \sum_{i_p} B^{(p)*}_{\alpha_{p-1}^b,\alpha_p^b}(i_p) X^{(p)}_{\alpha_{p-1}^x,\alpha_p^x}(i_p) (F_R^{(p+1)})_{\alpha_p^b,\alpha_p^x} \tag{19}$$

The cost of the tensor contraction scales like $\mathcal{O}(D_x^2 D_b d + D_b^2 D_x d)$.



(a) $E_L^{(3)}$ (b) $E_R^{(L-2)}$ (c) $F_L^{(3)}$ (d) $F_R^{(L-2)}$

FIG. 1. Tensor network diagrams for left environments ($E_L^{(3)}$, $F_L^{(3)}$) and right environments ($E_R^{(L-2)}$, $F_R^{(L-2)}$), where L is the length of the QTT. Building environments is typically done recursively, with portions of the TN in the dotted boxes computed in the previous iteration. Tensors in QTT-vectors (x, b) are in green, while tensors in the QTT-operator (A) are in blue.

Note that the environments are defined recursively. This is because in addition to computing the next the environment tensor, they can also be used to compute the effective operators introduced in the previous section. At the ends of the tensor train, for $p = 0$ or $p = L+1$, the environment tensor is equal to one (and the indices of the environment tensor are dummy indices of size one).

2. *Comb geometry*

For a comb tensor network with K branches, one first computes the environments starting from the ends of the branch as described above, obtaining the right environments $\{E_{R,\text{branch}}^{(k)}\}$. The primary difference lies in contracting the tensor along the spine.

Again, for concreteness, let us consider the expectation value $\langle b|A|x\rangle$. In this case the kth branch environment tensor has three indices, $\beta_k^b, \beta_k^A, \beta_k^x$, where β denotes the bond connecting the spine tensor ($\tilde{M}^{(k)}$ in Eq. (15)) to the first tensor of the branch tensor train ($M^{(k;1)}$).

For the first branch, the environment tensor including the spine is (see Fig. 2(a))

$$\left(E_{L,\text{spine}}^{(1)}\right)_{\gamma_1^b,\gamma_1^A,\gamma_1^x} = \sum_{\beta_1^b,\beta_1^A,\beta_1^x} \left(E_{R,\text{branch}}^{(1)}\right)_{\beta_1^b,\beta_1^A,\beta_1^x} X^{(1)}_{\beta_1^x,\gamma_1^x} A^{(1)}_{\beta_1^A,\gamma_1^A} B^{(1)*}_{\beta_1^b,\gamma_1^b} . \tag{20}$$

Again, B, A, and X correspond to tensors in the QTCs for b, A, and x, respectively. For subsequent branches ($k > 1$), the environment tensor is (see Fig. 2(b))

$$\left(E_{L,\text{spine}}^{(k)}\right)_{\gamma_l^b,\gamma_k^A,\gamma_k^x} = \sum_{\substack{\beta_1^b,\,\beta_1^A,\,\beta_1^x\\ \gamma_k^b,\,\gamma_k^A,\,\gamma_k^b}} \left(E_{L,\text{spine}}^{(k-1)}\right)_{\gamma_{k-1}^b,\gamma_{k-1}^A,\gamma_{k-1}^x} \left(E_{R,\text{branch}}^{(k)}\right)_{\beta_k^b,\beta_k^A,\beta_k^x} X^{(k)}_{\beta_k^x,\gamma_k^x} A^{(k)}_{\beta_k^A,\gamma_k^A} B^{(k)*}_{\beta_k^b,\gamma_k^b} . \tag{21}$$

The optimal order for tensor contraction will greatly depend on the size of the bonds. For most PDEs (in which operators are sums of differentials), the sizes of bonds γ^A and β^A are of order 1. In this case, using a tensor contraction order of $E_{L,\text{spine}}$, A, X, $E_{R,\text{branch}}$, and finally B, the computational cost scales like $\mathcal{O}(S_b^3 S_x + S_x^3 S_b)$, where S_x and S_b are the bond dimension of the spine (both γ and β) for QTC-vectors x and b. The right spine environment ($E_{R,\text{spine}}^{(k)}$) would be computed analogously starting from the last tensor in the spine.

At times, one would like to compute the left environment of the spine (starting the from the side connected to the branch. In this case, one has to utilize the spine environment tensors (see Fig. 2(c)):

$$\left(E_{L,\text{branch}}^{(k;0)}\right)_{\beta_k^b,\beta_k^A,\beta_k^x} = \sum_{\substack{\gamma_{k-1}^b,\,\gamma_{k-1}^A,\,\gamma_{k-1}^x\\ \gamma_k^b,\,\gamma_k^A,\,\gamma_k^x}} \left(E_{L,\text{spine}}^{(k-1)}\right)_{\gamma_{k-1}^b,\gamma_{k-1}^A,\gamma_{k-1}^x} \left(E_{R,\text{spine}}^{(k+1)}\right)_{\gamma_k^b,\gamma_k^A,\gamma_k^x} B^{(k)*}_{\gamma_{k-1}^b,\gamma_k^b,\beta_k^b} A^{(k)}_{\gamma_{k-1}^A,\gamma_k^A,\beta_k^A} X^{(k)}_{\gamma_{k-1}^x,\gamma_k^x,\beta_k^x} . \tag{22}$$

III. DENSITY-MATRIX ALGORITHM FOR MATRIX-VECTOR MULTIPLICATION

The naive algorithm for performing matrix-vector multiplication involves contracting the pth tensor in the QTN and QTN-O together for all sites p in the tensor network, and then compressing the newly obtained QTN [1, 6]. The

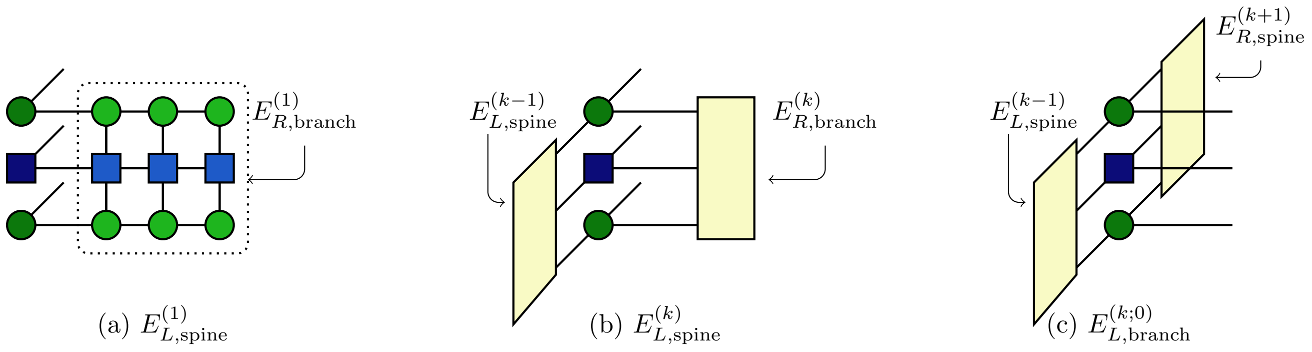

$E^{(1)}_{R,\text{branch}}$
(a) $E^{(1)}_{L,\text{spine}}$
$E^{(k-1)}_{L,\text{spine}}$
$E^{(k)}_{R,\text{branch}}$
(b) $E^{(k)}_{L,\text{spine}}$
$E^{(k+1)}_{R,\text{spine}}$
$E^{(k-1)}_{L,\text{spine}}$
(c) $E^{(k;0)}_{L,\text{branch}}$

FIG. 2. Tensor network diagrams depicting Eqs. (20) –(22). Tensors in QTC-vectors (x, b) are in green, while tensors in the QTC-operator (A) are in blue. Darker shades denote tensors in the spine.

Procedure	**QTT**	**QTC-spine**
Operator-vector multiplication		
Naive [4]	$\mathcal{O}(D^3 D_W^3 d^2)$	$\mathcal{O}(S^4 S_W^4)$
Zip-up [5]	$\mathcal{O}(D^3 D_W d^2)$ to $\mathcal{O}(D^3 D_W^3 d^2)$	$\mathcal{O}(S^4 S_W)$ to $\mathcal{O}(S^4 S_W^4)$
Density matrix [2]	$\mathcal{O}(D^3 D_W^2 d + D^2 D_W^3 d^2)$ + $(Dd \times Dd)$ eigenvalue problem	$\mathcal{O}(S^5 S_W^2)$ + $(S^2 \times S^2)$ eigenvalue problem
Local update schemes		
Build environment tensor (with n operators)	$\mathcal{O}(D^3 (D_w)^n d + n D^2 D_W^2 d^2)$	$\mathcal{O}(S^4 S_W^n + n S^2 S_W^5)$
Compute A_{eff}	$\mathcal{O}(D_W^2 D^2 d^2 + D^4 D_W d^2)$	$\mathcal{O}(S^2 S_W^3 + S^4 S_W^2 + S^6 S_W)$
Compute $A_{\text{eff}} x$	$\mathcal{O}(2 D_W D^3 d + D_W^2 D^2 d^2)$	$\mathcal{O}(S^2 S_W^3 + 2 S^4 S_W^2 + S^4 S_W)$

TABLE I. Theoretical costs of common procedures for the quantized tensor train (QTT) and the spine of the comb-like tree tensor network (QTC-spine). In the QTT geometry, vectors have bond dimension D while operators have bond dimension D_W. In the spine of the comb geometry, vectors and operators have bond dimension S, S_W along the spine, respectively. The costs are obtained assuming S_W and D_W are smaller than S and D. If this is not the case, other tensor contraction orderings may be optimal.

compression (in particular, the canonicalization, which is like a preconditioning step) is expensive; for the tensor train geometry, the computational cost scales like $\mathcal{O}(D^3 D_W^3 d)$, where D is the bond dimension of the QTT and D_W is the bond dimension of the QTT-O.

Instead, one could consider algorithms that canonicalize the resulting QTT as the tensors are being contracted. One such algorithm is the zip-up algorithm, whose cost scales like $\mathcal{O}(D D_W \tilde{D}^2 d^2)$, where $\tilde{D}$ is an intermediate bond dimension that is problem dependent, ranging in value from D to $D D_W$ [5]. In this work, we mainly used the density-matrix algorithm [2, 7], detailed below. Note that after these algorithms are used, the QTNs are in canonical form. We observe better performance after compressing again using SVD, sweeping in the opposite direction. This calculation is no longer as costly, since the bond dimension has already been reduced.

A. Basic Algorithm for QTTs

In the density matrix algorithm, instead of canonicalizing the QTT and compressing using SVD, it involves computing the partial density matrix of $\langle x|A^\dagger A|x\rangle$ at each site, and computing its eigenvalues (which one truncates in the low-rank approximation). The basic algorithm for QTTs is written in Alg. 1. However, we recommend referring to Institute [2] for diagrammatic representations of the tensor contractions.

Algorithm 1 Density Matrix Compression

```
Input: QTT $x$, QTT-O $A$ of length $L$
Output: QTT $y$, the low-rank approximation of $Ax$, and of bond dimension $D$
Notes: sums are performed over all repeated indices

Build left environments
If $E_L^{(0)}$ is not given, set to one
Build left environments $\{E_L^{(p)}\}$ for $\langle x|A^\dagger A|x\rangle$ starting from $E_L^{(0)}$

Compute compressed tensors
procedure COMPRESS REDUCED DENSITY MATRIX (END)
    Compute density matrix $\rho^{(L)}_{o'_L,o_L} \leftarrow \sum E^{(L-1)}_{\alpha'_{L-1},\beta'_{L-1},\beta_{L-1},\alpha_{L-1}} X^{(L)*}_{\alpha'_{L-1}}(i'_L)\, A^{(L)*}_{\beta'_{L-1}}(i'_L,o'_L)\, A^{(L)}_{\beta_{L-1}}(o_L,i_L)\, X^{(L)}_{\alpha_{L-1}}(i_L)$
    Solve eigenvalue problem $U^{(L)}_{o'_L,\eta_L},\ \lambda^{(L)}_{\eta_L} \leftarrow \textsc{Eig}(\rho^{(L)}_{o'_L,o_L})$, keeping only the $D$ largest eigenvalues
    Define $M^{(L)}_{\alpha_{L-1}}(i_L) \leftarrow U_{o'_L,\eta_L}$, $\textsc{Reindex}(\eta_L \to \alpha_{L-1}, o'_L \to i_L)$
    Compute right (compressed) environment $C^{(L)}_{\alpha_{L-1},\beta_{L-1},\eta_L} \leftarrow \sum X^{(L)}_{\alpha_{L-1}}(i_L)\, O^{(L)}_{\beta_{L-1}}(o_L,i_L)\, U^{*(L)}_{o_L,\eta_L}$

for site $p = L-1, ..., 2$ do
    procedure COMPRESS REDUCED DENSITY MATRIX
        Compute density matrix $\rho^{(p)}_{o'_p,\eta'_{p+1},o_p,\eta_{p+1}} \leftarrow \sum E^{(p-1)}_{\alpha'_{p-1},\beta'_{p-1},\beta_{p-1},\alpha_{p-1}} X^{(p)*}_{\alpha'_{p-1},\alpha'_p}(i'_p)\, A^{(p)*}_{\beta'_{p-1},\beta'_p}(i'_p,o'_p)$
                $A^{(p)}_{\beta_{p-1},\beta'_p}(o_p,i_p)\, X^{(p)}_{\alpha_{p-1},\alpha_p}(i_p)\, C^{(p+1)}_{\alpha_o,\beta_p,\eta'_{p+1}}\, C^{*(p+1)}_{\alpha'_p,\beta'_p,\eta'_{p+1}}$
        Solve eigenvalue problem $U^{(i)}_{(o'_p,\eta'_{p+1}),\eta_p},\ \lambda^{(p)}_{\eta_p} \leftarrow \textsc{Eig}\ (\rho^{(p)}_{(o'_p,\eta'_{p+1}),(o_p,\eta_{p+1})})$, keeping only the $D$ largest eigenvalues
        Define $M^{(p)}_{\alpha_{p-1},\alpha_p}(i_p) \leftarrow U_{o'_p,\eta'_{p+1},\eta_p}$, $\textsc{Reindex}(\eta'_{p+1} \to \alpha_p,\ \eta_p \to \alpha_{p-1}, o'_p \to i_p)$
        Compute right (compressed) environment $C^{(i)}_{\alpha_{p-1},\beta_{p-1},\eta_p} \leftarrow \sum X^{(i)}_{\alpha_{p-1},\alpha_p}(i_p) O^{(i)}_{\beta_{p-1},\beta_p}(o_p,i_p) U^{(i)*}_{o_p,\eta_{p+1},\eta_p} C^{(i+1)}_{\alpha_p,\beta_p,\eta_{p+1}}$

$M^{(1)}_{\alpha_1}(i_1) \leftarrow \sum x^{(1)}_{\alpha_1}(i_1) A^{(1)}_{\beta_1}(o_1,i_1) C^{(2)}_{\alpha_1,\beta_1,\eta_2}$, $\textsc{Reindex}(\eta_2 \to \alpha_1, o_1 \to i_1)$

$y \leftarrow \mathrm{QTT}(M^{(1)}, M^{(2)}, ..., M^{(L)})$ which is in right canonical form.
```

B. Computational Cost

The primary costs arise from computing the left environments and performing the eigendecomposition of the reduced density matrices ρ. With physical bond dimension d, QTT bond dimension D, and QTT-O bond dimension D_W, the cost of tensor contraction scales like $\mathcal{O}(D^3 D_W^2 d + D^2 D_W^3 d^2)$. The cost of the eigendecomposition depends on the algorithm used. For exact eigendecomposition (which is not necessary since we are only interested in the D largest eigenvalues), we must first explicitly compute the density matrix, whose cost scales like $\mathcal{O}(D^3 D_W^2 d^2 + D^2 D_W^3 d^2))$. The eigendecomposition itself scales like $\mathcal{O}((Dd)^3)$. An iterative eigensolver would be attractive, as it may allow one to avoid explicitly computing ρ.

C. Extension to Comb Geometry

The algorithm introduced above is designed for the 1-D tensor train geometry. However, it can be easily extended to the comb geometry. Application of the operator along the branches uses the same procedure as above. One must then compress the branches into the spine, and then compress the spine itself. The algorithm is described in Alg. 2. Following the discussion earlier in Section II B 2, the cost of computing the environments in the comb geometry scale like $\mathcal{O}(S^4 S_W^2 + S^2 S_W^5)$ in addition to the costs of computing the environments of the branches. Here, S is the size of the bonds in the QTC spine, S_W is the size of the bonds in the QTC-O spine, and we assume that $S_W < S$. The cost of computing ρ scales like $\mathcal{O}(S^5 S_W^2)$, and the resulting eigenvalue problem is of size S^2. Not only is this becoming relatively expensive, we encounter numerical issues with standard partial eigenvalue solvers (near-zero eigenvalues are not easily found).

We instead update spine tensors using the zip-up algorithm. The procedure for updating the tensors is given in Alg. 3; the algorithm otherwise remains essentially the same. The cost of updating the spine now scales ranges from $\mathcal{O}(S^4)$ to $\mathcal{O}(S^4 S_W^4)$, depending on the problem.

Algorithm 2 Density Matrix Compression for Comb Geometry

```
Input: QTC $x$, QTC-O $A$ with $K$ branches
Output: QTC $y$, the low-rank approximation of $Ax$

Build left environments
▷ Build branch environments ◁
for spine index $k = 1, ..., K-1$ do
    Compute environments of each branch $\left\{\left(E^{(k;p)}_{R,\text{branch}}\right)\right\}$ for $p = 1, \ldots, L$.   ▷ The branches are QTTs of length $L$
▷ Build left spine environments ◁
Left spine environments from $k \in 1 \ldots K$: $\left\{\left(E^{k}_{L,\text{branch}}\right)\right\}$

Compute compressed tensors
for branch $k = K, ..., 1$ do
    ▷ Compress branch using Alg. 1. The branch is a QTT of length $L$ ◁
    Build left branch environment $E^{(k;0)}_{L,\text{branch}}$ from spine environments (Eq. (22))
    Compute compressed branch $B^{(k)} = \text{QTT}(M^{(k;1)}, ..., M^{(k;L)})$ using Alg. 1, with left environments built from $\tilde{E}^{(k;0)}_{R,\text{branch}}$
    ▷ Compress branch into spine ◁
    Compute unitary $U^{(k;1)}_{o'^k_1,\eta_2,\eta_1}$, $C^{(k;1)}_{\beta^A_k,\beta^x_k,\eta_1}$ using COMPRESS REDUCED DENSITY MATRIX step
    Update $M^{(k;1)}_{\beta_k,\alpha_{(k,1)}}(i^{(k)}_1) \leftarrow U^{(i;1)}_{o'^k_1,\eta_2,\eta_1}$, REINDEX $(\eta_2 \to \alpha_{(k,1)},\ \eta_1 \to \beta_k,\ o'^k_1 \to i^{(k)}_1)$

    procedure UPDATE SPINE TENSORS
        Initialize right (compressed) spine environment $\tilde{C}^{(K+1)} \leftarrow 1$
        if $k \neq 1$ then
            ▷ Compress spine tensor ◁
            ▷ for $k = K$, indices with $K+1$ are dummy indices of size 1 ◁
            Compute density matrix for spine
                $\tilde{\rho}^{(k)}_{\eta'_k,\xi'_{k+1},\eta_k,\xi_{k+1}} \leftarrow \sum \left(\tilde{E}^{(k-1)}_{L,\text{spine}}\right)_{\gamma'^x_{k-1},\gamma'^A_{k-1},\gamma^A_{k-1},\gamma^x_{k-1}} \tilde{X}^{(k)*}_{\gamma'^x_{k-1},\gamma'^x_k,\beta'^x_k} \tilde{A}^{(k)*}_{\gamma'^A_{k-1},\gamma'^A_k,\beta'^A_k} \tilde{A}^{(k)}_{\gamma^A_{k-1},\gamma^A_k,\beta^A_k} \tilde{X}^{(k)}_{\gamma^x_{k-1},\gamma^x_k,\beta^x_k}$
                    $C^{(k;1)}_{\beta^A_k,\beta^x_k,\eta_k} C^{(k;1)*}_{\beta'^A_k,\beta'^x_k,\eta'_k} \tilde{C}^{(k+1)}_{\gamma^A_k,\gamma^x_k,\xi_{k+1}} \tilde{C}^{(k+1)*}_{\gamma'^A_k,\gamma'^x_k,\xi'_{k+1}}$
            Solve eigenvalue problem $\tilde{U}^{(k)}_{(\eta'_k,\xi'_{k+1}),\xi_k}$, $\tilde{\lambda}^{(k)}_{\rho_k} \leftarrow \text{EIG}(\tilde{\rho}^{(k)}_{(\eta'_k,\xi'_{k+1}),(\eta_k,\xi_{k+1})})$
            Define $\tilde{M}^{(b)}_{\gamma_{k-1},\gamma_k,\beta_k} \leftarrow U_{\eta'_k,\xi'_{k+1},\xi_k}$, REINDEX$(\eta'_k \to \beta_k,\ \xi'_{k+1} \to \gamma_k,\ \xi_k \to \gamma_{k-1}))$   ▷ Updated spine tensor
            Define right (compressed) spine environment
                $\tilde{C}^{k}_{\gamma^A_{k-1},\gamma^x_{k-1},\xi_k} \leftarrow \sum \tilde{X}^{(k)}_{\gamma^x_{k-1},\gamma^x_k,\beta^x_k} \tilde{A}^{(k)}_{\gamma^A_{k-1},\gamma^A_k,\beta^A_k} \tilde{U}^{(k)*}_{\eta_k,\xi_{k+1},\rho_k} C^{(k;1)}_{\beta^A_k,\beta^x_k,\eta_k} \tilde{C}^{(k+1)}_{\gamma^A_k,\gamma^x_k,\xi_{k+1}}$
        else
            ▷ Define spine tensor ◁
            $\tilde{M}^{(1)}_{\gamma_1,\beta_1} \leftarrow \sum \tilde{x}^{(1)}_{\gamma^x_1,\beta^x_1} \tilde{A}^{(1)}_{\gamma^A_1,\beta^A_1} C^{(1;1)}_{\beta^A_1,\beta^x_1,\eta_1} \tilde{C}^{(2)}_{\gamma^A_1,\gamma^x_1,\xi_2}$
        Define right spine environment $\tilde{E}^{(k)}_{R,\text{spine}} \leftarrow \tilde{C}^{(k)}$.

Define new comb
Spine $\leftarrow \text{QTT}(\tilde{M}^{(1)}, ..., \tilde{M}^{(K)})$
QTC $y \leftarrow \text{Comb}(\text{Spine}; (B^{(1)}, ..., B^{(K)}))$
```

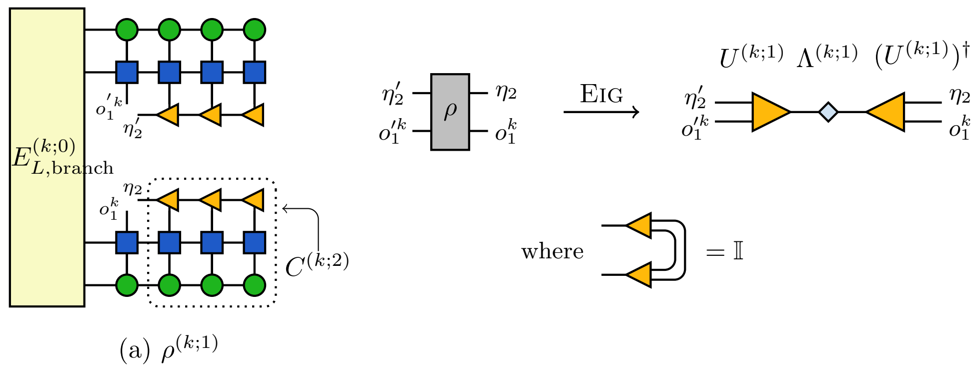


FIG. 3. TN diagrams showing the compression and canonicalization of first tensor in the branch ($M^{(k,1)}$) into the spine. The diagram looks the same as for a normal tensor train. The orange triangles denote that the tensor is in left/right canonical form, depending on the direction the triangle is pointing in.

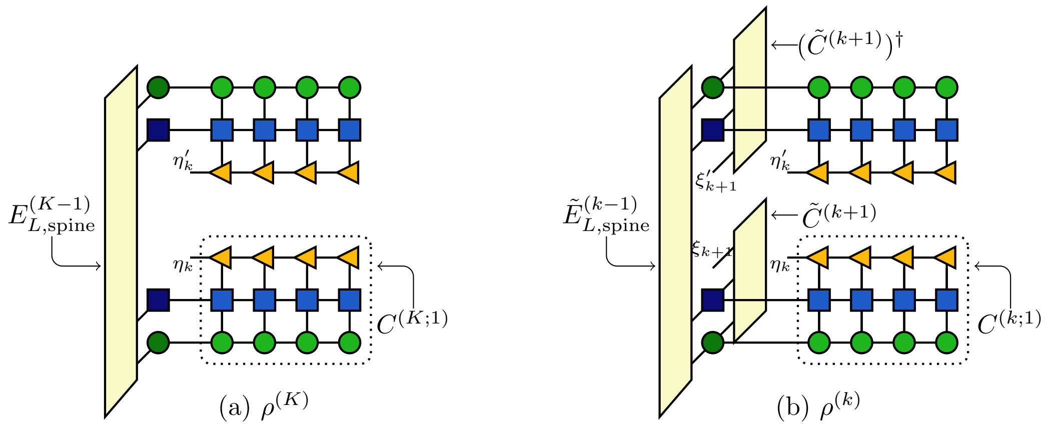


FIG. 4. TN diagrams showing the computation of density matrix ρ for spine tensors using the density matrix algorithm (a) at site K and (b) in the middle of the chain ($k = 2, ..., K-1$). This computation is expensive so we instead use the zip-up algorithm.

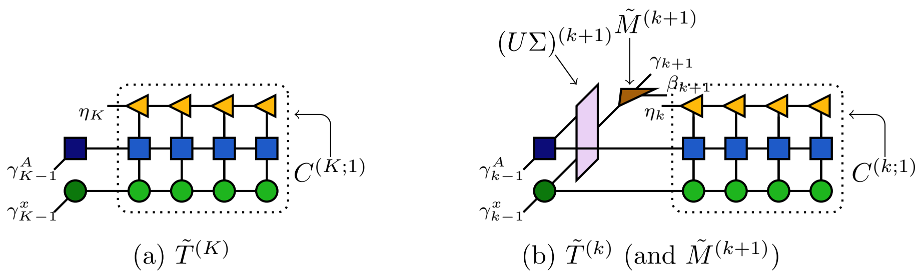


FIG. 5. TN diagrams showing the computation of tensor $T^{(k)}$ for spine tensors using the zip-up algorithm (a) at site $k = K$ and (b) in the middle of the chain ($k = 2, ..., K-1$)

.

Algorithm 3 Update spine procedure based on zip-up algorithm

```
Input: QTC $x$, QTC-O $A$ with $K$ branches
Output: QTC $y$, the low-rank approximation of $Ax$
procedure UPDATE SPINE TENSORS(spine index $k \in (1,\dots,K)$)
    ▷ Zip-up algorithm for spine ◁
    if $k = K$ then
        $\tilde{T}^{(K)}_{\gamma^A_{K-1},\gamma^x_{K-1},\eta_K} \leftarrow \sum \tilde{X}^{(K)}_{\gamma^x_{K-1},\gamma^x_K,\beta^x_K}\, \tilde{A}^{(K)}_{\gamma^A_{K-1},\gamma^A_K,\beta^A_K}\, C^{(K;1)}_{\beta^A_K,\beta^x_K,\eta_K}$
        $U^{(K)}_{(\gamma^A_{K-1},\gamma^x_{K-1}),\gamma_{K-1}}, \Sigma^{(K)}_{\gamma_{K-1}}, V^{(K)}_{\eta_K,\gamma_{K-1}} \leftarrow \mathrm{SVD}\left(\tilde{T}^{(K)}_{(\gamma^A_{K-1},\gamma^x_{K-1}),\eta_K}\right)$
            where only singular values above a cutoff threshold ($\sim 10^{-10}$) are retained.
        Define $\tilde{M}^{(K)}_{\gamma_{K-1},\beta_K} \leftarrow V^{(K)}_{\eta_K,\gamma_{K-1}}$, REINDEX($\eta_K \to \beta_K$)    ▷ Updated spine tensor
    else if $2 \le k < K$ then
        $\tilde{T}^{(k)}_{\gamma^A_{k-1},\gamma^x_{k-1},\gamma_k,\eta_k} \leftarrow \sum \tilde{X}^{(k)}_{\gamma^x_{k-1},\gamma^x_k,\beta^x_k}\, \tilde{A}^{(k)}_{\gamma^A_{k-1},\gamma^A_k,\beta^A_k}\, C^{(k;1)}_{\beta^A_k,\beta^x_k,\eta_k}\, \tilde{U}^{(k+1)}_{(\gamma^A_k,\gamma^x_k),\gamma_k}\, \tilde{\Sigma}^{(k+1)}_{\gamma_k}$
        $U^{(k)}_{(\gamma^A_{k-1},\gamma^x_{k-1}),\gamma_{k-1}}, \Sigma^{(k)}_{\gamma_{k-1}}, V^{(k)}_{\gamma_k,\eta_k,\gamma_{k-1}} \leftarrow \mathrm{SVD}\left(\tilde{T}^{(k)}_{(\gamma^A_{k-1},\gamma^x_{k-1}),\gamma_k,\eta_k}\right)$
            where only singular values above a cutoff threshold ($\sim 10^{-10}$) are retained.
        Define $\tilde{M}^{(k)}_{\gamma_{k-1},\gamma_k,\beta_k} \leftarrow V^{(k)}_{\gamma_k,\eta_k,\gamma_{k-1}}$, REINDEX($\eta_k \to \beta_k$)    ▷ Updated spine tensor
    else if $k = 1$ then
        $\tilde{T}^{(1)}_{\gamma_1,\eta_1} \leftarrow \sum \tilde{X}^{(1)}_{\gamma^x_1,\beta^x_1}\, \tilde{A}^{(1)}_{\gamma^A_1,\beta^A_1}\, C^{(1;1)}_{\beta^A_1,\beta^x_1,\eta_1}\, \tilde{U}^{(2)}_{(\gamma^A_1,\gamma^x_1),\gamma_1}\, \tilde{\Sigma}^{(2)}_{\gamma_1}$
        Define $\tilde{M}^{(1)}_{\gamma_1,\beta_1} \leftarrow T^{(1)}_{\gamma_1,\eta_1}$, REINDEX($\eta_1 \to \beta_1$)    ▷ Updated spine tensor
    If sweeping from right to left, build right spine environment $\tilde{E}^{(k)}_{R,\mathrm{spine}}$. Else, build left spine environment.
```

IV. TIME-DEPENDENT VARIATIONAL PRINCIPLE (TDVP)

Consider a unitary system, with

$$\frac{\partial}{\partial t}\psi(t) = A\psi(t) \tag{23}$$

where A is an anti-Hermitian operator. According to the Dirac-Frenkel principle, the time derivative at time t exists on the submanifold tangent to $u(t)$,

$$\langle v|\dot{u}(t) - Au(t)\rangle = 0, \quad \forall v \in \mathcal{T}_{u(t)}\mathcal{M}\,. \tag{24}$$

In the time-dependent variational principle (TDVP) algorithm, we evolve the tensors forward in time one by one, using equation of motions obtained from satisfying Eq. (24). Let us begin by representing the QTT for discretized $u(x)$ in right-canonical form,

$$|u(i_1, i_2, ...i_L; t)\rangle = \sum_{\alpha_1,...\alpha_{L-1}} M^{(1)}_{\alpha_1}(i_1; t)\, B^{(2)}_{\alpha_1,\alpha_2}(i_2; t) \ldots B^{(L)}_{\alpha_{L-1}}(i_L; t) \tag{25}$$

where the tensors denoted by B are right canonical; i.e., $\sum_{i_p,\alpha_p} B^{*(p)}_{\alpha_{p-1},\alpha_p}(i_p) B^{(p)}_{\alpha_{p-1},\alpha_p}(i_p) = \mathbb{I}$. We want to obtain an equation of motion for $M^{(1)}$. Keeping the other tensors fixed, the time derivative of $u(t; M^{(1)})$ is

$$\frac{\partial}{\partial t}|u(t; M^{(1)})\rangle = \dot{M}^{(1)} \frac{\partial}{\partial M^{(1)}}|u(t; M^{(1)})\rangle\,. \tag{26}$$

According to Eq. (24), the approximate dynamics is obtained by minimizing

$$||\dot{M}^{(1)} \frac{\partial}{\partial M^{(1)}}|u(t; M^{(1)})\rangle - A|u(t; M^{(1)})\rangle||. \tag{27}$$

Utilizing the fact that u is in right canonical form, we obtain

$$\begin{aligned}\dot{M}^{(1)}(t) &= \frac{\partial}{\partial M^{(1)*}}\langle u(t; M^{(1)})|A|u(t; M^{(1)}\rangle \\ &= \left(\frac{\partial}{\partial M^{(1)*}}\frac{\partial}{\partial M^{(1)}}\langle u(t; M^{(1)})|A|u(t; M^{(1)})\rangle\right) M^{(1)} \\ &= A^{(1)}_{\text{eff}}\, M^{(1)}.\end{aligned} \tag{28}$$

Note that A_{eff} is the original anti-Hermitian matrix projected onto the submanifold tangent to $|u(t; M^{(1)})\rangle$.

Once $M^{(1)}$ is updated to the next time step using some time integration scheme (e.g., exact integration or RK4), we prepare to shift the orthogonality center of the QTT to the next site by performing a QR decomposition on $M^{(1)}(t + dt) = A^{(1)}(t + dt)R^{(1)}(t + dt)$.

Because of the gauge degree of freedom, the tensors are not actually independent of each other, so one must propagate $R^{(1)}(t + dt)$ backwards in time before updating site 2. This is done in a similar fashion, solving

$$\dot{R}^{(1)} = -A^{(1|2)}_{\text{eff}} R^{(1)}\,, \tag{29}$$

$$A^{(1|2)}_{\text{eff}} = \left(\frac{\partial}{\partial R^{(1)*}}\frac{\partial}{\partial R^{(1)}}\langle u(t; M^{(1)})|A|u(t; R^{(1)})\rangle\right)\,. \tag{30}$$

Note the negative sign, since we are propagating backwards in time. Once $R^{(1)}(t)$ is obtained, it is contracted with the tensor at the second site, $B^{(2)}$. The orthogonality center has now been shifted to site 2, and we repeat the above steps to evolve site 2 forward in time.

In summary, the algorithm is as follows. For each site in the QTT from 1 to $L - 1$: evolve site p forward in time, evolve the bond between p and $p + 1$ backwards in time, and canonicalize the QTT to site $p + 1$. Finally, the last site at $p = L$ is propagated forwards in time. Note that the bond dimension of the QTT does not change as one propagates forward in time.

Updating the tensors in this sweeping fashion has a Trotter error associated with it. The scheme presented is a first-order scheme, and has error $\mathcal{O}(\Delta t)$. By sweeping from left to right and then right to left, doing time evolution with $\Delta t/2$ in each sweep, one obtains a second-order time evolution scheme with error $\mathcal{O}(\Delta t^2)$. Projecting the true dynamics onto the tensor network manifold also introduces some error. However, in the case that a single site is updated at a time, as introduced here, the expectation value $\langle u|A|u\rangle$, and the norm of $|u\rangle$ and conserved [8, 9]. In the two-site variant of TDVP, one updates two neighboring tensors at one time. The advantage is that one can adaptively increase the rank as needed, thereby reducing some of the projection error. However, this update typically requires compression of the two tensors (so that the bond dimension remains manageable), breaking the conservation properties of TDVP.

We refer readers to Haegeman *et al.* [8] and Paeckel *et al.* [9] for more technical details.

A. Computational Cost (TT geometry)

The cost of tangent-space methods is dominated by the calculation of $A_{\text{eff}}^{(i)}$ and the time evolution of a the tensor M_i, which is of size $D \times D \times d$. Recall that D is the bond dimension of the QTT representation of x and d is the size of the physical indices. If A is represented as an QTT-O with bond dimension D_W, then the cost of computing $A_{\text{eff}}^{(i)}$ scales like $\mathcal{O}(D^2 D_W^2 d^2 + D^4 d^2 D_W^2)$ and exact time evolution via exact diagonalization scales like $\mathcal{O}((D^2 d)^3)$. If one solved Eq. (28) using an approximate scheme such as RK4, one can avoid explicitly computing $A_{\text{eff}}^{(i)}$ and instead compute $A_{\text{eff}}^{(i)} M$ (where M is a generic tensor of the appropriate size). For RK4, the cost of time evolution then scales like $\mathcal{O}(2D^3 D_W d + D^2 D_W^2 d^2)$, arising solely from tensor contraction.

B. TDVP in Comb Geometry

The TDVP algorithm for QTCs mainly differ in the order in which tensors are propagated forward in time. We use a 1-site TDVP algorithm with the ordering proposed in Bauernfeind and Aichhorn [10] and summarized in Alg. 4:

Algorithm 4 TDVP sweeping algorithm for QTC

```
for branch $k = 1, \ldots K - 1$ do
    Perform TDVP on branch starting from the free end (away from the spine), including the spine tensor
    Perform back propagation on the bond between spine tensor $\tilde{M}^{(k)}$ and $\tilde{M}^{(k+1)}$.
For branch $k = K$, perform TDVP on the branch, starting from spine tensor $\tilde{M}^{(K)}$ and ending at the free end.
```

For a second order algorithm, the above algorithm and its exact reverse is performed, each with time step $\Delta t/2$. Also note that while not stated explicitly, the QTC must be properly canonicalized when performing TDVP. Because the sweeping algorithm is not continuous, TDVP does not automatically put the QTC in the proper canonical form (as is the case for QTTs).

C. Computational Cost (Comb geometry)

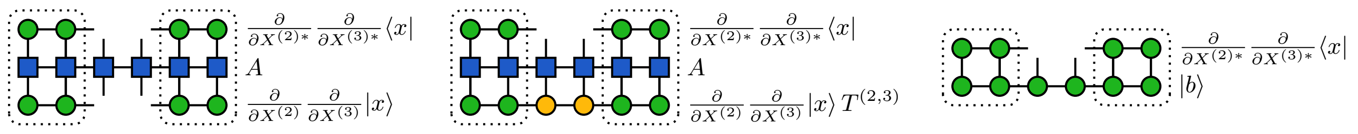
The cost of TDVP for the branches is the same as for the TT geometry. If using exact diagonalization, the cost of calculating the effective operator scales like $\mathcal{O}(S^6)$ and the cost of exact diagonalization scales like $\mathcal{O}(S^9)$. Again, S is the size of all bonds in the spine for the QTC. While small for most initial states, S will generally will grow over time. We assume the bonds in the spine for the QTC-O to be small and constant, so they are ignored in the cost estimates. Unless we require S to be very small, we likely would never use exact diagonalization for spine tensors because of the high cost. If one instead uses RK4, the cost scales like $\mathcal{O}(S^4)$.

V. DENSITY MATRIX RENORMALIZATION GROUP

A. Basic DMRG Algorithm

The density-matrix renormalization group (DMRG) algorithm is an optimization algorithm for quantized tensor networks in which each tensor (or each neighboring pair of tensors) is optimized in a sequential fashion. Typical applications include finding extreme eigenvalues [1, 11] and solving linear equations [12, 13]. In this work, we are interested in the latter application.

Let $|x\rangle$ and $|b\rangle$ be QTTs of bond dimensions D_x and D_b, respectively, and let A be a QTT-O of bond dimension D_A. For our application, $D_b \sim \mathcal{O}(D_x)$ and D_A is a small integer value. All have physical bond dimension d (2 in our


$\frac{\partial}{\partial X^{(2)*}}\frac{\partial}{\partial X^{(3)*}}\langle x|$
A
$\frac{\partial}{\partial X^{(2)}}\frac{\partial}{\partial X^{(3)}}|x\rangle$
$\frac{\partial}{\partial X^{(2)*}}\frac{\partial}{\partial X^{(3)*}}\langle x|$
A
$\frac{\partial}{\partial X^{(2)}}\frac{\partial}{\partial X^{(3)}}|x\rangle\, T^{(2,3)}$
$\frac{\partial}{\partial X^{(2)*}}\frac{\partial}{\partial X^{(3)*}}\langle x|$
$|b\rangle$

(a) $A_{\text{eff}}^{(2,3)}$ (b) $A_{\text{eff}}^{(2)}T^{(2,3)}$ (c) $b_{\text{eff}}^{(2,3)}$

FIG. 6. Tensor network diagrams for (a) $A_{\text{eff}}^{(2,3)}$, (b) $A_{\text{eff}}^{(2,3)}T^{(2,3)}$, and (c) $b_{\text{eff}}^{(2,3)}$. Portions of the TN in the dotted boxes are typically precomputed as environment tensors.

case) and are of length L. To solve the linear equation $A|x\rangle = |b\rangle$, each neighboring pair of tensors are to be updated in an iterative fashion with the solution to the reduced problem

$$A_{\text{eff}}^{(p,p+1)}T^{(p,p+1)} = b_{\text{eff}}^{(p,p+1)}, \tag{31}$$

where

$$T^{(p,p+1)} = \sum_{\alpha_p} X^{(p)}_{\alpha_{p-1},\alpha_p} X^{(p+1)}_{\alpha_p,\alpha_{p+1}}\,, \tag{32}$$

$$b_{\text{eff}}^{(p,p+1)} = \frac{\partial}{\partial T^{(p,p+1)*}}\langle x|b\rangle\,, \tag{33}$$

$$A_{\text{eff}}^{(p,p+1)} = \frac{\partial}{\partial T^{(p,p+1)}}\frac{\partial}{\partial T^{(p,p+1)*}}\langle x|A|x\rangle\,, \tag{34}$$

and $x^{(p)}$ is the p^{th} tensor in $|x\rangle$. The tensor network diagram representations are shown in Fig. 6. The size of the reduced problems are $D_x^2 d^2$, and can be solved using traditional methods. We opt to solve the reduced problem using conjugate gradient descent (CGD) [14] or conjugate gradient squared (CGS) for non-Hermitian A [15], so as to avoid computing $A_{\text{eff}}^{(i,i+1)}$ explicitly (see Alg. 5) [16].

The entire DMRG algorithm is summarized in Alg. 6, with three primary steps:

1. First is to build the left or right environments for $\langle x|A|x\rangle$ and for $\langle x|b\rangle$. Again, the environments are generally built recursively because each are also used to compute $A_{\text{eff}}^{(p,p+1)}$ and $b_{\text{eff}}^{(p,p+1)}$. The cost of building the environment scales like $\mathcal{O}(D_x^3 D_A d + D_x^2 D_A^2 d^2)$.

2. The second step is to update the tensors in $|x\rangle$ by solving Eq. (31) as described above, starting with sites 1 and 2 of the QTT and then sweeping left to right through the chain of tensors. In this case, one only needs to first build the right environments. After updating sites p and $p+1$, one computes new left environments $E^{L,(p)}$ and $F^{L,(p)}$ as depicted in Fig. 1. It is also equally valid to sweep through the QTT tensors from right to left. In this case, one first only builds the left environments, and computes new right environments $E^{R,(p+1)}$ and $F^{R,(p+1)}$ after updating tensors p and $p+1$. The cost of each sweep is dominated by the calculation of $A_{\text{eff}}^{(p,p+1)}T$, which scales like $\mathcal{O}((D_x^2 D_b + D_x D_b^2)D_A d + D_x D_b D_A^2 d^2)$ per CGD iteration.

3. At the end of each sweep, one computes the error $||b - Ax||^2$. If the error is still too large, one continues to the next iteration, sweeping in the opposite direction. By computing the error as $\langle b|b\rangle - 2\langle b|A|x\rangle + \langle x|A^\dagger A|x\rangle$, the cost typically is dominated by the last term, which scales like $\mathcal{O}(2D_x^3 D_A^2 d + 2D_x^2 D_A^3 d^2)$. One could consider a less exact measure of error to reduce the cost of this step.

B. Block matrix DRMG

Suppose that A contains some structure and is a block matrix. For example, in the case of an implicit solver for Maxwell's equations, each component of the electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$ correspond to a subblock in the input $|x\rangle$ and target $|b\rangle$, and the matrix A denotes couplings between the components. In this case, the linear equation to be solved is of the form

$$\sum_{\xi} A_{\eta,\xi}|x_\xi\rangle = |b_\eta\rangle \tag{35}$$

Algorithm 5 Conjugate gradient descent

```
Input: tensor network $A$, tensor $b$, tensor $x$ (initial guess)
Output: tensor $x$
procedure CGD($A$, $b$, $x$)
    $\beta \leftarrow Ax$
    $r \leftarrow b - Ax$
    $d \leftarrow r$
    error $\varepsilon \leftarrow |r|$
    $i \leftarrow 1$
    while $i <$ MAXITER and $\varepsilon/|b| >$ CONVTOL do
        tensor $q \leftarrow$ CONTRACT($A, d$)                ▷ Contract all tensors together
        scalar $p \leftarrow$ CONTRACT($A, d, d^\dagger$)
        scalar $\alpha \leftarrow \varepsilon^2/p$
        tensor $x \leftarrow x + \alpha d$
        tensor $r \leftarrow r - \alpha q$
        scalar $\varepsilon_{\text{old}} \leftarrow \varepsilon$
        scalar $\varepsilon \leftarrow |r|$                   ▷ new error is the norm of the new residual
        scalar $\beta \leftarrow \varepsilon^2/\varepsilon_{\text{old}}^2$
        tensor $d \leftarrow r + \beta d$
        int $i \leftarrow i + 1$
    return $x$
```

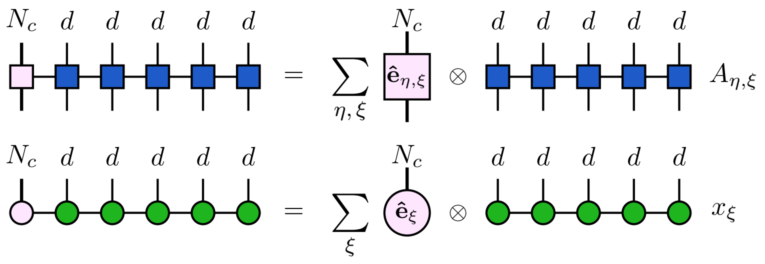


FIG. 7. Expanded QTT-O (top) and QTT (bottom), which are originally of length $L = 5$ and physical bond d. They are generated by summing together QTTs multiplied by an additional tensor that indexes the QTTs position in the block matrix or vector. Matrix $\hat{e}_{\eta,\xi}$ is an $N_c \times N_c$ matrix with all elements set to zero except for the $(\eta, \xi)^{\text{th}}$ element which is set to one. Vector η_ξ is a unit vector of length M of where the ξ^{th} element is one and otherwise is zero. If the QTTs are originally of bond dimension D, the bond dimension of the expanded QTT (without compression) is $N_c^2 D$ for the QTT-Operator and $N_c D$ for the QTT-State.

where η and ξ index the position of each submatrix.

Similar to what is done in classical solvers by appending vectors and matrices together and expanding the problem space, one can combine the QTTs for each field by adding an extra tensor which denotes which field component is being operated on (see Fig. 7). The size of the physical bond dimension would be the number of of field components N_c (six in the case of Maxwell's equations). One then solves the system of equations using DMRG as described in the previous section. By construction, the optimization of a single site in the tensor train updates all field components. While the implementation is straightforward, the calculation can be costly. If the QTTs of each field component is of bond dimension D, then it is likely that the bond dimension of the composite QTT will need to be larger than D in order to capture all of the original information.

Alternatively, one can perform a modified DMRG algorithm, in which one updates the tensors of each component $|x_\eta\rangle$ by solving the reduced problem

$$\sum_{\xi} (A_{\eta,\xi})_{\text{eff}}^{(p,p+1)} \, T_{\xi}^{(p,p+1)} = (b_\eta)_{\text{eff}}^{(p,p+1)} \,, \tag{36}$$

Algorithm 6 Two-site DMRG with conjugate gradient descent

```
Input: QTT $b$ and initial guess QTT $x$, QTT-Operator $A$. QTTs are of $L$ sites, integer bond dimension $D$
Output: the solution QTT-State $x$ of bond dimension $D$

Define environment $E_L^{(0)} \leftarrow 1$, $E_R^{(L+1)} \leftarrow 1$
Build left environments
for site $p = 1, ..., L-1$ do
    Compute left environment $E_L^{(p)} \leftarrow \textsc{Contract}(E_L^{(p-1)}, x^{(p)}, A^{(p)}, x^{(p)*})$

Define environment $F_L^{(0)} \leftarrow 1$, $F_R^{(L+1)} \leftarrow 1$
for site $p = 1, ..., L-1$ do
    Compute left environment $F_L^{(p)} \leftarrow \textsc{Contract}(F_L^{(p-1)}, b^{(p)}, x^{(p)*})$

while error $\varepsilon <$ CONVTOL and iteration $i <$ MAXITER do
    ▷ Begin DMRG sweeps (right to left) ◁
    for site $p = L-1, ..., 1$ do

        $A_{\text{eff}}^{(p,p+1)} \leftarrow (E_L^{(p-1)}, E_R^{(p+2)}, A^{(p)}, A^{(p+1)})$
        $b_{\text{eff}}^{(p,p+1)} \leftarrow \textsc{Contract}(F_L^{(p-1)}, F_R^{(p+2)}, b^{(p)}, b^{(p+1)})$
        $T^{(p,p+1)} \leftarrow \textsc{Contract}(x^{(p)}, x^{(p+1)})$

        $T^{(p,p+1)} \leftarrow \textsc{Conjugate Gradient Descent}(A_{\text{eff}}^{(p,p+1)}, b_{\text{eff}}^{(p,p+1)}, T^{(p,p+1)})$
        $U_{(\alpha_{p-1},i_p),\beta}, S_\beta, V_{\beta,(\alpha_{p+1},i_{p+1})} = \text{SVD}\left(T^{(p,p+1)}_{(\alpha,i_{p-1}),(\alpha_{p+1},i_{p+1})}\right)$ keeping $D$ largest singular values

        ▷ Put in right canonical form ◁
        $x^{(p+1)}_{\alpha_p,\alpha_{p+1}}(i_{p+1}) \leftarrow V_{\beta,\alpha_{p+1},i_{p+1}}$
        $x^{(p)}_{\alpha_{p-1},\alpha_p}(i_p) \leftarrow U_{\alpha_{p-1},i_p,\beta}, S_\beta$

        ▷ Update left environments ◁
        $E_R^{(p+1)} \leftarrow \textsc{Contract}(E_R^{(p+2)}, A^{(p+1)}, x^{(p+1)}), x^{(p+1)*})$
        $F_R^{(p+1)} \leftarrow \textsc{Contract}(F_R^{(p+2)}, b^{(p+1)}, x^{(p+1)*})$

    Measure error $\varepsilon \leftarrow \textsc{Error}(A,\, b,\, x)$
    $i \leftarrow i + 1$

    if $\varepsilon <$ CONVTOL then
        break

    ▷ Begin DMRG sweeps (left to right) ◁
    for site $p = 1, ..., L-1$ do

        $A_{\text{eff}}^{(p,p+1)} \leftarrow (E_L^{(p-1)}, E_R^{(p+2)}, A^{(p)}, A^{(p+1)})$
        $b_{\text{eff}}^{(p,p+1)} \leftarrow \textsc{Contract}(F_L^{(p-1)}, F_R^{(p+2)}, b^{(p)}, b^{(p+1)})$
        $T^{(p,p+1)} \leftarrow \textsc{Contract}(x^{(p)}, x^{(p+1)})$

        $T^{(i,i+1)} \leftarrow \textsc{Conjugate Gradient Descent}(A_{\text{eff}}^{(i,i+1)}, b_{\text{eff}}^{(i,i+1)}, T^{(i,i+1)})$
        $U_{(\alpha_{p-1},i_p),\beta}, S_\beta, V_{\beta,(\alpha_{p+1},i_{p+1})} = \text{SVD}\left(T^{(i,i+1)}_{(\alpha_{p-1},i_p),(\alpha_{p+1},i_{p+1})}\right)$ keeping $D$ largest singular values

        ▷ Put in left canonical form ◁
        $x^{(p+1)}_{\alpha_p,\alpha_{p+1}}(i_{p+1}) \leftarrow S_\beta V_{\beta,\alpha_{p+1},i_{p+1}}$
        $x^{(p)}_{\alpha_{p-1},\alpha_p}(i_p) \leftarrow U_{\alpha_{p-1},i_p,\beta}$

        ▷ Update right environments ◁
        $E_L^{(p)} \leftarrow \textsc{Contract}(E_L^{(p-1)}, A^{(p)}, x^{(p)}), x^{(p)*})$
        $F_L^{(p)} \leftarrow \textsc{Contract}(F_R^{(p-1)}, b^{(p)}, x^{(p)*})$

    Measure error $\varepsilon \leftarrow \textsc{Error}(A,\, b,\, x)$
    $i \leftarrow i + 1$
```

where

$$T_\xi^{(p,p+1)} = \sum_{\alpha_p} (X_\xi)^{(p)}_{\alpha_{p-1},\alpha_p} (X_\xi)^{(p+1)}_{\alpha_p,\alpha_{p+1}} \,, \tag{37}$$

$$(b_\eta)^{(p,p+1)}_{\text{eff}} = \frac{\partial}{\partial T_\eta^{(p,p+1)*}} \langle x_\eta | b_\eta \rangle \,, \tag{38}$$

$$(A_{\eta,\xi})^{(p,p+1)}_{\text{eff}} = \frac{\partial}{\partial T_\xi^{(p,p+1)}} \frac{\partial}{\partial T_\eta^{(p,p+1)*}} \langle x_\eta | A_{\eta,\xi} | x_\xi \rangle \,. \tag{39}$$

The modified DMRG algorithm for solving the linear equation in block form (summarized in Alg. 7) is similar to the original DMRG algorithm, though additional bookkeeping is needed. Notice that we decided to simultaneously update the p^{th} tensor in the QTTs for each component in a single optimization step. The primary procedural differences are in performing CGD of the reduced problem (see Alg. 8) and computing the error,

$$\varepsilon = \sum_\eta \left(\langle b_\eta | b_\eta \rangle - 2 \sum_\xi \langle b_\eta | A_{\eta,\xi} | x_\xi \rangle + \sum_{\xi,\beta} \langle x_\xi | A^\dagger_{\eta,\xi} A_{\eta,\beta} | x_\beta \rangle \right) \,.$$

Computationally, it is often advantageous to utilize the block form because the bond dimension needed to represent each subvector $|x_\xi\rangle$ is likely smaller than that required for the full vector.

Algorithm 7 DMRG utilizing block matrices

```
Input: QTT-Os $\{A_{\eta,\xi}\}$, QTTs $\{x_\xi\}$ (initial guess), QTTs $\{\beta_\xi\}$, number of components $M$
Output: QTTs $\{x_\xi\}$

for all $\xi$, $\eta \in \{1, \dots, M\}$ do
    Build Right Environments($A_{\eta,\xi}$, $x_\xi$)
    Build Right Environments($b_\eta$, $\xi_\eta$)

while error $\varepsilon <$ CONVTOL and iteration $i <$ MAXITER do

    ▷ Sweeping left to right ◁
    for site $p = 1, ..., L-1$ do
        procedure Solve reduced($p$)
            Obtain tensor network for $(A_{\eta,\xi})^{(p,p+1)}_{\text{eff}}$ for all $\eta$, $\xi$
            Compute $(b_\eta)^{(p,p+1)}_{\text{eff}}$ for all $\eta$
            Compute $T_\eta^{(p,p+1)}$ for all $\eta$
            Solve for $\{T_\eta\} \leftarrow$ BlockCGD($\{A_{\eta,\xi}\}$, $\{T_\eta\}$, $\{b_\eta\}$)    ▷ (See Alg. 8)
        for component $\eta = 1, \dots, M$ do
            Compute $U_{\alpha_{p-1},i_p,\beta} S_\beta V^\dagger_{\beta,\alpha_{p+1},i_{p+1}} \leftarrow \text{SVD}\left(\left(T_\eta^{(p,p+1)}\right)_{(\alpha_{p-1},i_p),(\alpha_{p+1},i_{p+1})}\right)$ keeping the $D$ largest singular values
            Update $x_\eta^{(p)} \leftarrow U_{\alpha_{p-1},i_p,\beta}$ and $x_\eta^{(p+1)} \leftarrow V_{\beta,\alpha_{p+1},i_{p+1}} S_\beta$
            Extend left environments with updated $x_\eta^{(p)}$
    Compute error $\varepsilon$. If converged, return $\{x_\xi\}$

    ▷ Sweeping right to left ◁
    for $p = L-1, ..., 1$ do
        $T^{(i,i+1)} \leftarrow$ Solve reduced($p$)
        for component $\eta = 1, \dots, M$ do
            Compute $U_{\alpha_{p-1},i_p,\beta} S_\beta V^\dagger_{\beta,\alpha_{p+1},i_{p+1}} \leftarrow \text{SVD}\left(\left(T_\eta^{(p,p+1)}\right)_{(\alpha_{p-1},i_p),(\alpha_{p+1},i_{p+1})}\right)$ keeping the $D$ largest singular values
            Update $x_\eta^{(p)} \leftarrow U_{\alpha_{p-1},i_p,\beta} S_\beta$ and $x_\eta^{(p+1)} \leftarrow V_{\beta,\alpha_{p+1},i_{p+1}}$
            Extend right environments with updated $x_\eta^{(p+1)}$
    Compute error. If converged, return $\{x_\xi\}$
```

Algorithm 8 Conjugate gradient descent with block matrices

```
Input: list of tensor networks $A_{\eta,\xi}$, kust if tensors $b_\eta$, list of tensors $x_\xi$ (initial guess)
Output: tensor $x$
procedure BLOCKCGD($A_{\eta,\xi}$, $b_\eta$, $x_\xi$)
    $\beta_\eta \leftarrow \sum_\xi A_{\eta,\xi} x_\xi$
    $r_\eta \leftarrow b_\eta - \beta_\eta$
    $d_\eta \leftarrow r_\eta$
    error $\varepsilon \leftarrow \sqrt{\sum_\eta |r_\eta|^2}$
    $i \leftarrow 1$
    while $i <$ MAXITER and $\varepsilon/|b| >$ CONVTOL do
        tensor $q_\eta \leftarrow \sum_\xi$ CONTRACT$(A_{\eta,\xi}, d_\xi)$
        scalar $p \leftarrow \sum_{\eta,\xi}$ CONTRACT$(A_{\eta,\xi}, d_\eta, d_\xi^\dagger)$
        scalar $\alpha \leftarrow \varepsilon^2/p$
        tensor $x_\eta \leftarrow x_\eta + \alpha d_\eta$
        tensor $r_\eta \leftarrow r_\eta - \alpha q_\eta$
        scalar $\varepsilon_{\text{old}} \leftarrow \varepsilon$
        scalar error $\varepsilon \leftarrow \sqrt{\sum_\eta |r_\eta|^2}$        ▷ new error is the norm of the new residual
        scalar $\beta \leftarrow \varepsilon^2/\varepsilon_{\text{old}}^2$
        tensor $d_\eta \leftarrow r_\eta + \beta d_\eta$
        int $i \leftarrow i + 1$
    return $x$
```

C. Extension to Comb Geometry

In this work, we do not extend this DMRG solver to the comb geometry, since the problems that must be solved are at most 2-D problems. In this case, the comb geometry is essentially a tensor train.

VI. ADDITIONAL RESULTS

A. Orszag-Tang vortex

1. Results for calculations with QTC geometry

The electric and magnetic (EM) fields for simulations of the Orszag-Tang vortex performed with bond dimension $D = 64$ and time step $\Delta t = 0.001\Omega_{c,p}^{-1}$ are shown in Fig. 8. As mentioned in the main text, the electric fields appear to be the primary source of noise.

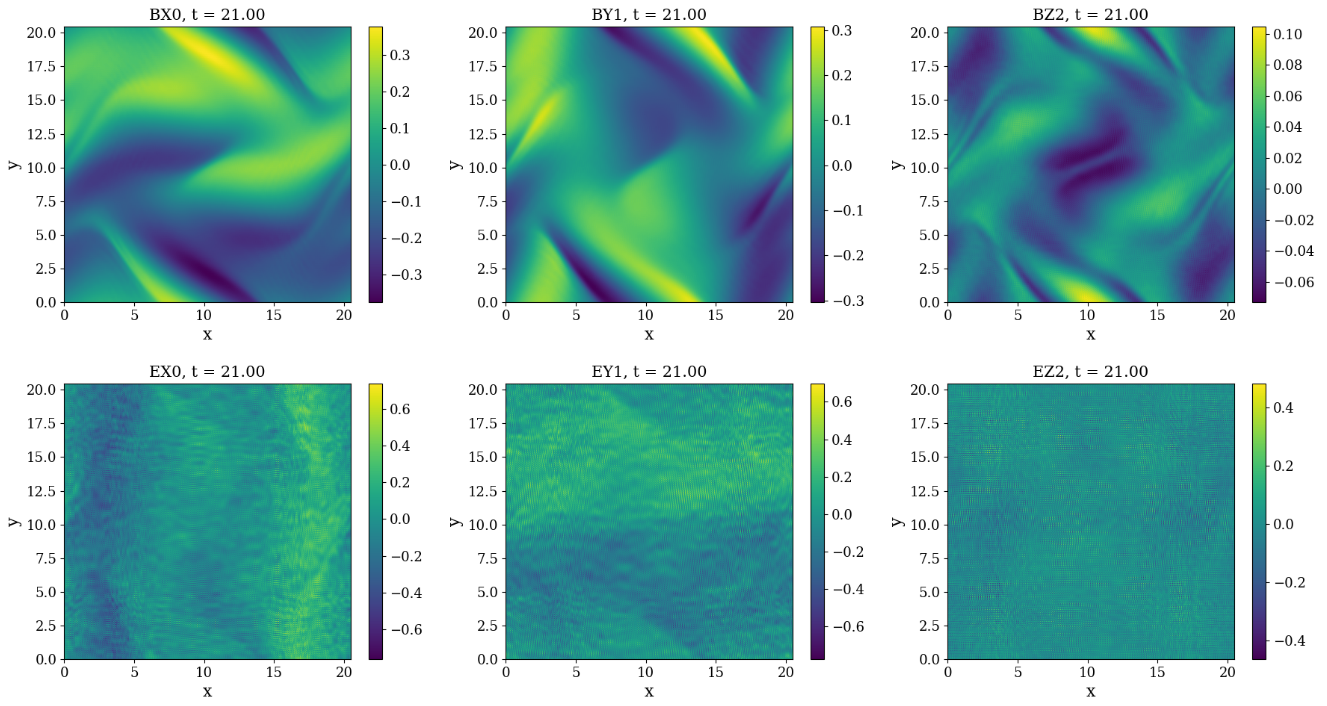


FIG. 8. Each component of the magnetic (top) and electric (bottom) fields at a time of $21\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.001\Omega_{c,p}^{-1}$ and bond dimension $D = 64$.

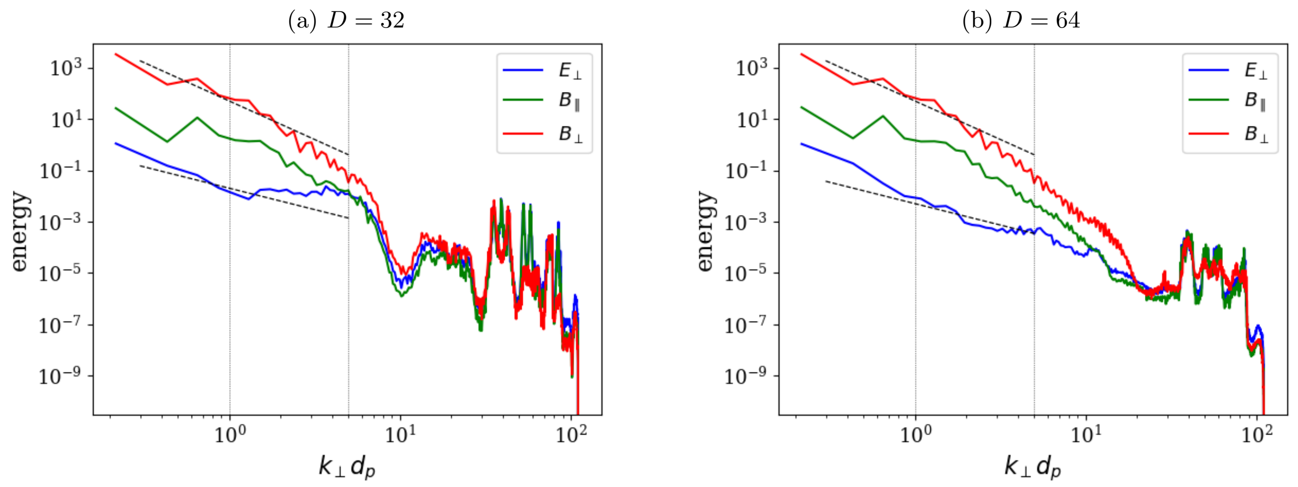


FIG. 9. Energy spectrum at a time of $21\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.001\Omega_{c,p}^{-1}$. Dashed black lines show power laws of -3 and -5/3.

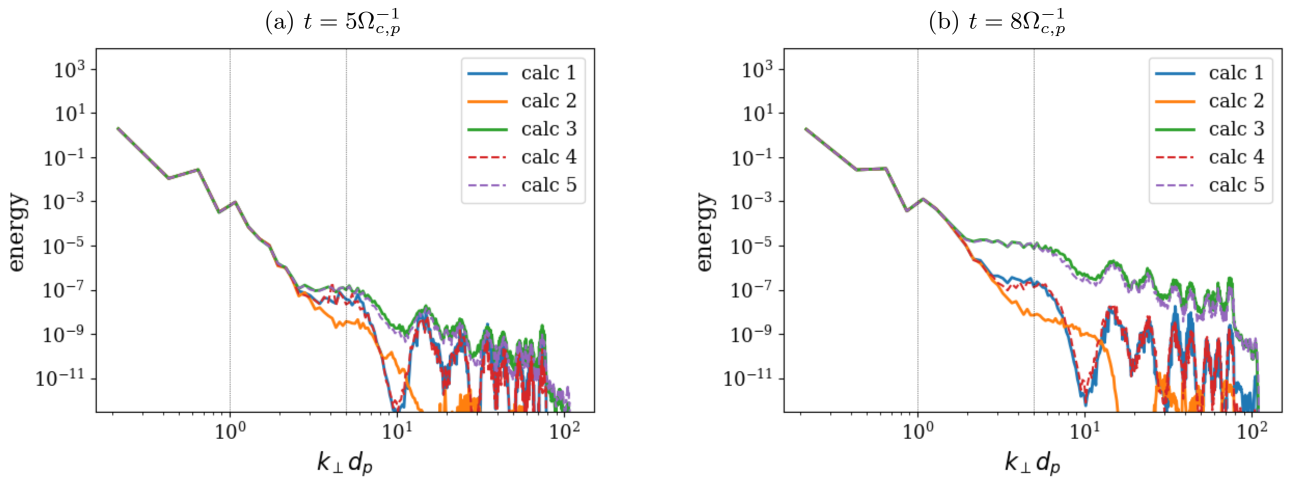


FIG. 10. Energy spectrum of the electric field at times of (a) $5\Omega_{c,p}^{-1}$ and (b) $7\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.001\Omega_{c,p}^{-1}$. Calculations are as follows; (1) calculations with $D = 32$, (2) calculations with $D = 64$, (3) calculations with $D_f = 32$ for the distribution functions and $D_{EM} = 128$ for the EM fields, (4) calculations with $D = 32$ but the DMRG solver is performed with bond dimension 128, (5) calculations with $D_f = 32$ and $D_{EM} = 128$ but the electric field is compressed to bond dimension 32 before measuring the energy spectrum.

Fig. 9 shows the energy spectra of the EM fields for calculations with $D = 32$ and $D = 64$ at time $t = 21\Omega_{c,p}^{-1}$. Both calculations show similar spectra for $k_\perp d_e < 1$. (Note that given the amount of noise in the electric field, its spectrum is not accurate.) However, the spectra are noticeably different for $k_\perp d_e > 1$. Contrary to expectations, the $D = 64$ result does not show a significant steepening in the magnetic energy spectra. The spectra also have noticeable dips at $k_\perp d_p = 10$ for the $D = 32$ calculation and $k_\perp d_p = 20$ for the $D = 64$ calculation, which exactly correspond to the effective grid resolutions for their respective bond dimensions. This dip seems to limit the resolution that the EM fields can achieve, again explaining the observation mentioned in the main text that the features in J_z were broader for the $D = 32$ calculation than for $D = 64$. This would suggest that the bond dimension and its associated effective grid resolution must be large enough to resolve the desired features in the simulation. This is worrying, since in the event that one would like to consider larger scale separation between the box size and the kinetic scales, one would need to use larger bond dimension such that the effective grid can fully resolve the same fine structures.

However, as argued in the main text, this observation is likely a result of the numerical noise. This noise may be introduced through low-rank approximation as well as the Crank-Nicolson solver, and does not necessarily reflect the efficiency of the QTT for representing the true solution. Fig. 10 shows the energy spectra of E_z obtained with different bond dimensions for the EM fields at times of (a) $t = 5\Omega_{c,p}^{-1}$ and (b) $t = 7\Omega_{c,p}^{-1}$. Solid lines are results for (1) $D = 32$ and (2) $D = 64$ computed using the same bond dimension for the distribution functions, and (3) $D = 128$ but computed using bond dimension 32 for the distribution functions. Complementary to results shown in the main text, the flat spectrum for the third calculation is indicative of numerical noise. Because the Crank-Nicolson solver and the Vlasov equation have no dissipation, any numerical noise that is introduced continues to build. Dashed lines are results for (4) $D = 32$ but the QTT is compressed after solving Crank-Nicolson using DMRG with bond dimension 128 and (5) the same as (3) but the QTT of the E_z field is compressed to $D = 32$ when constructing this plot. Results from (4) closely match results from (1), suggesting that the dip does not result from poor convergence of the DMRG solver but instead results from compression of the QTT to smaller rank. However, because results for (5) match those from (3), it is not solely the compression of the fields that generates the dip in the spectra. It is instead a cumulative effect from performing the low-rank approximation at each time step. While the observations of an “effective grid resolution” certainly warrant further investigation, it is not necessarily the case that finite bond dimension will limit the maximum resolution that can be achieved.

2. *Entanglement entropy for different QTN ansatzë*

In Fig. 11, we measure the entanglement entropy at each bond for three different geometries:

- the QTC geometry with flipped binary mapping for each axis (presented in the main paper).
- QTT-SFB: QTT geometry with sequential ordering, with binary mapping for position axes and flipped binary mapping for velocity axes
- QTT-PFM: QTT geometry with interleaved ordering, with forward binary mapping for position axes and mirror mapping for velocity axes

As expected, of the three geometries studied, QTT-SFB on average has the largest EE. Interestingly, while the distribution function for the QTT-PFM geometry has the lowest EE, the EE in the electric field grows much faster than in the other two cases.

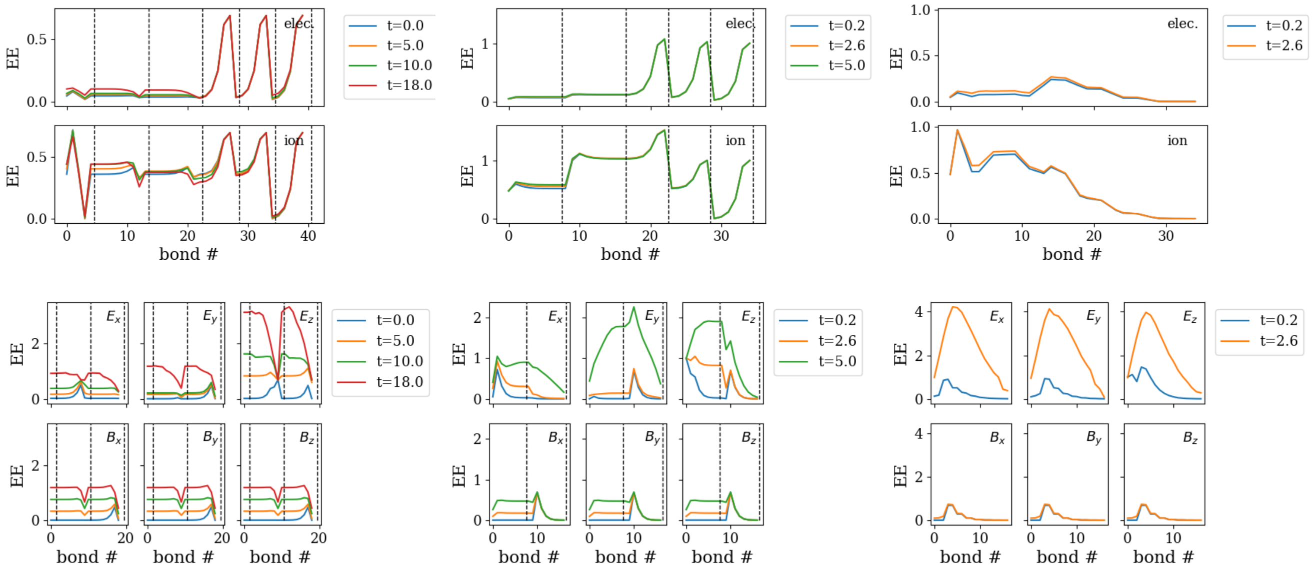


FIG. 11. Entanglement entropy measured at each bond in the QTN for the electron and ion distributions (top) and the electric and magnetic fields (bottom), obtained using (left) the QTC ansatz with $D = 64$, (middle) the QTT-SFB ansatz with $D = 128$, and (right) and the QTT-PFM ansatz with $D = 32$. Calculations were performed with a time step of $\Delta t = 0.002$. Time is in units of $\Omega_{c,p}^{-1}$.

3. *Calculations with uncentered Crank-Nicolson*

Single-step time integration schemes can generally be written as

$$\frac{\phi_{n+1} - \phi_n}{\Delta t} = (1-\alpha)F(\phi_n, t_n) + \alpha F(\phi_{n+1}, t_{n+1})\,, \tag{40}$$

where ϕ_n is the solution and $F(\phi_n, t_n)$ is the time derivative of ϕ at time t_n. In the case that $\alpha = 1/2$, one obtains the Crank-Nicolson scheme. The time integration scheme is stable for $\alpha \geq 0.5$ [17].

In Fig. 12, we compare the contour plots and spectra of the electric fields obtained using (1) the original Crank-Nicolson scheme and (2) Eq. (40) with $\alpha = 0.6$ to perform integration of Maxwell's equations. Unlike the Crank-Nicolson scheme, the un-centered scheme introduces numerical dissipation, which visibly helps dampen the noise in the electric field. However, additional work to determine how one can efficiently and accurately reduce noise even further is required.

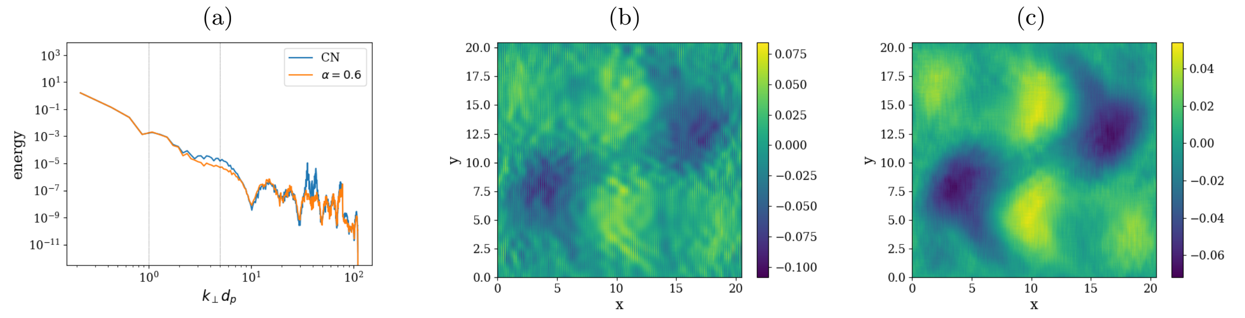


FIG. 12. Plots comparing results using different solvers for updating Maxwell's equations, obtained using bond dimension $D = 32$ and time step $\Delta t = 0.002\Omega_{c,p}^{-1}$. Plots are at a time of $t = 12\Omega_{c,p}^{-1}$. (a) Energy spectrum of the perpendicular electric field. (b) Contour plot of E_z computed using a Crank-Nicolson solver (Eq. (40) with $\alpha = 0.5$). (c) Contour plot of E_z computed using an un-centered solver (Eq. (40)) with $\alpha = 0.6$.

4. *Observed scalings of cost with respect to QTN bond dimension*

The wall-time of the real-space and velocity-space advection steps are measured with respect to QTN bond dimension D for the Orszag-Tang problem. (Since the velocity-space advection step is performed twice per actual time step, the plotted value is those two times averaged together.) The wall-times are measured at various time steps and appear to converge by 1000 time steps. The observed costs are roughly $\mathcal{O}(D^{2.5})$ for computing the advection term in real-space and $\mathcal{O}(D^4)$ for the advection term in velocity-space, which is consistent with theoretical expectations.

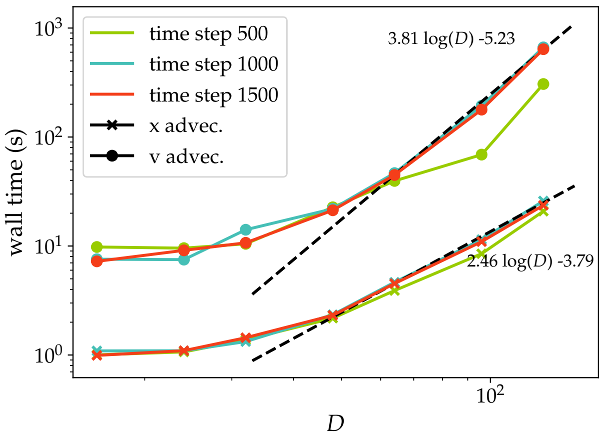


FIG. 13. Plot of wall times for the real-space advection step (denoted using 'x' markers) and velocity-space advection step (denoted using circle markers) computed with QTCs of various bond dimension D. Wall times are measured at the specified time steps of the Orszag-Tang problem, and the time step size is $\Delta t = 0.002\Omega_{c,p}^{-1}$. The values of D measured are 16, 24, 32, 48, 64, 96, and 128. Dashed black lines are fitted lines, demonstrating the computational scalings for each calculation.

B. GEM reconnection problem

Figs. 14 and 15 show the magnetic field, electric field, and distribution function cross sections at times of $25\Omega_{c,p}^{-1}$ and $40\Omega_{c,p}^{-1}$. The simulations were performed with with $D = 64$ and time step $0.005\Omega_{c,p}^{-1}$. At $t = 25\Omega_{c,p}^{-1}$, some numerical noise is already evident in the electric field components.

Fig. 16 shows the spectra of the electric and magnetic (EM) fields at those two times, for calculations performed with $D = 32$ and $D = 64$. At $t = 25\Omega_{c,p}^{-1}$, the spectra for the two bond dimensions look similar, and show the expected steepening at $k_\perp d_e > 1$. For $d_p^{-1} < k_\perp < d_e^{-1}$, $B_\perp$ roughly obeys a k^{-3} power law while $E_\perp$ and $B_\parallel$ appear to fit a $k^{-5/3}$ power law; for $k_\perp d_e > 1$, the decay in magnetic energy roughly follows k^{-8}. However, at $t = 40\Omega_{c,p}^{-1}$, the spectra for $D = 32$ appear much flatter, which is due to the fields being dominated by noise.

Following the discussion in the previous section, since the EM fields are represented as QTTs with rank $D = 128$, we might expect to observe a dip in the spectra at around $k_\perp d_p = 40$. Though not as pronounced as in the Orszag-Tang simulations, it does appear as if there might be a dip instead of noise at that length scale.

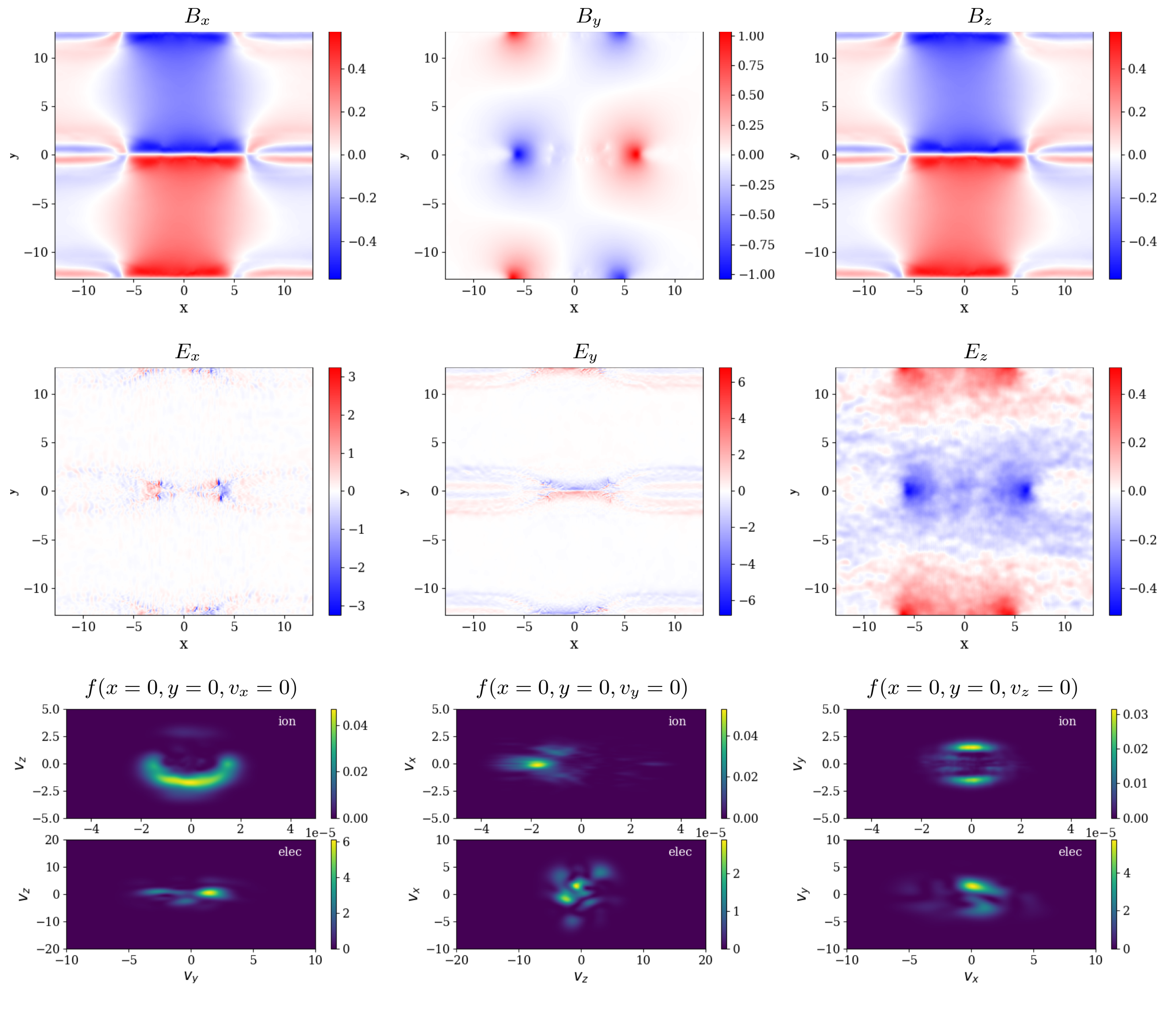


FIG. 14. Each component of the magnetic (top) and electric (center) fields at a time of $25\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.005$ and bond dimension $D = 64$. At the bottom are plots of the distribution function at the X-point, showing the cross section at zero for the axis not plotted (ordered as v_x, v_y, v_z from left to right).

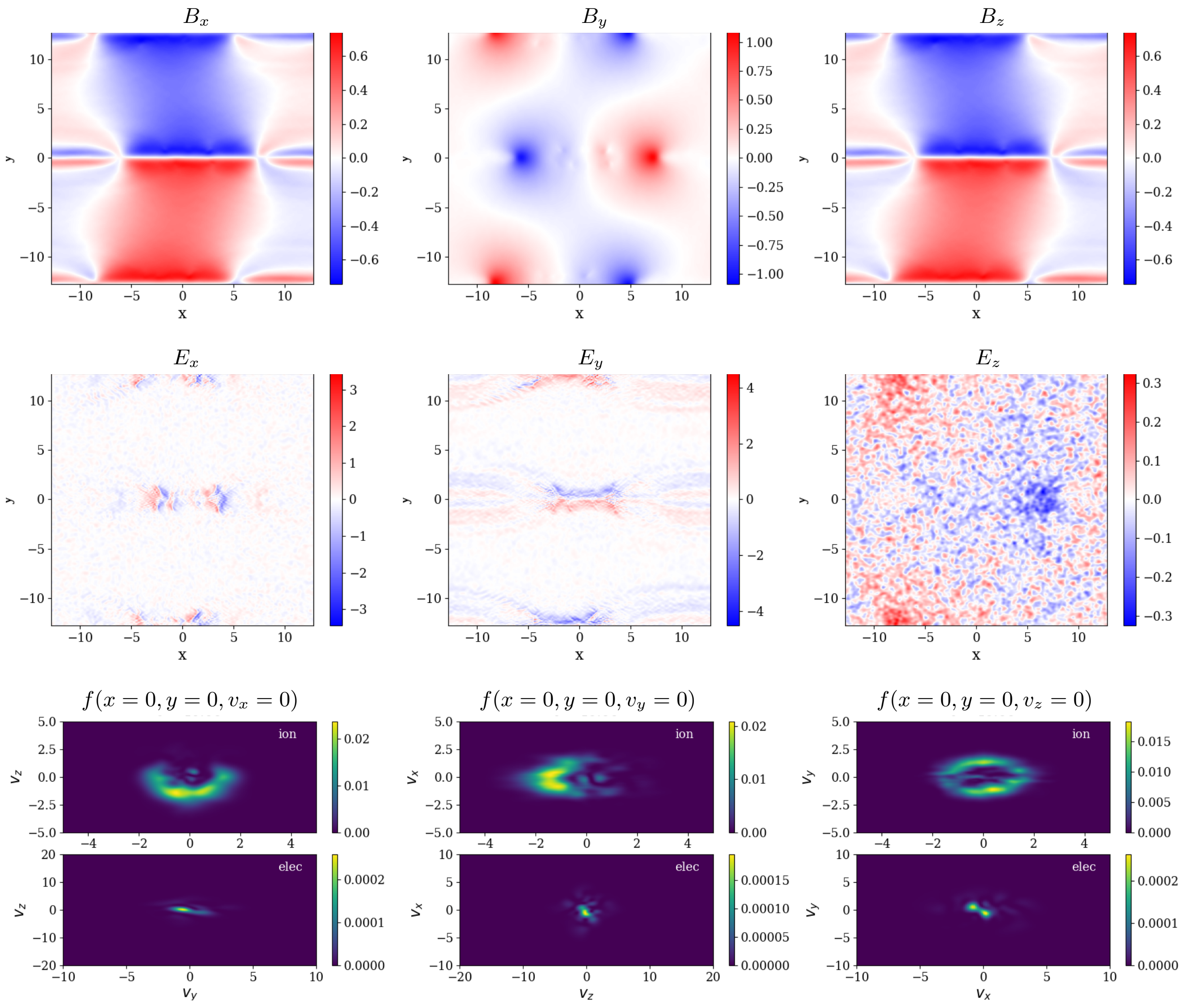
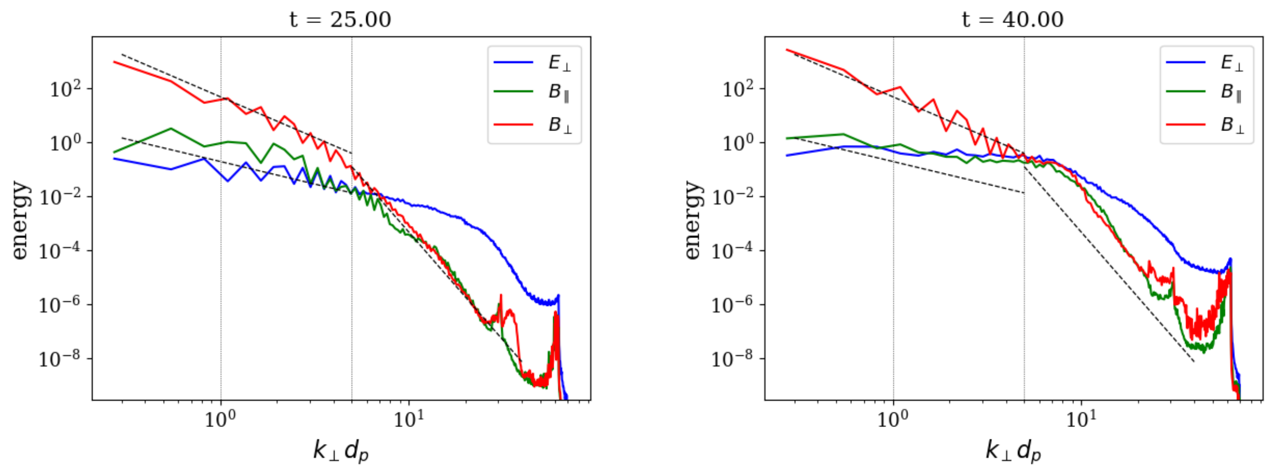
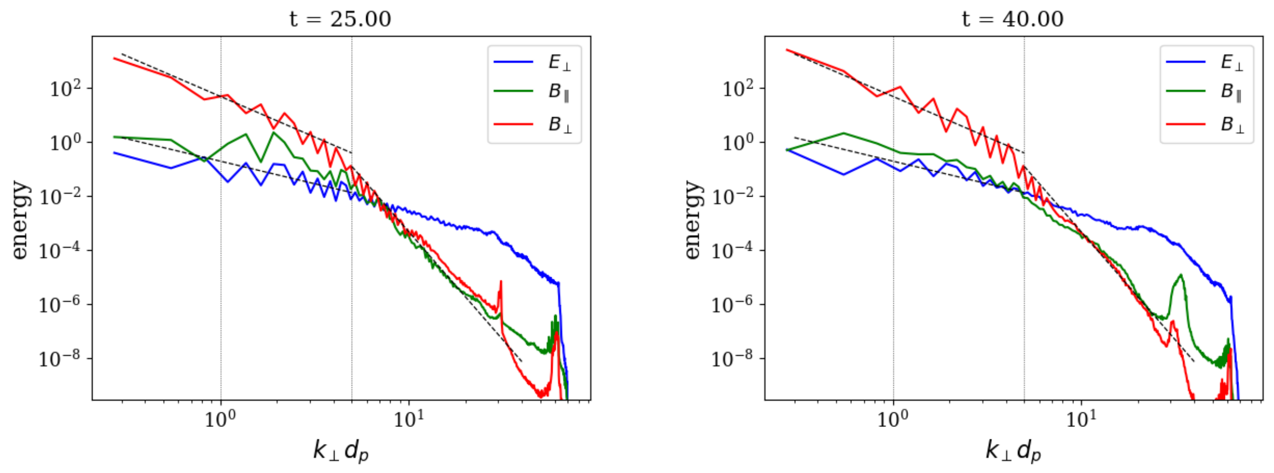


FIG. 15. Each component of the magnetic (top) and electric (middle) fields at a time of $40\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.005\Omega_{c,p}^{-1}$ and bond dimension $D = 64$. At the bottom are plots of the distribution function at the X-point, showing the cross section at zero for the axis not plotted (ordered as v_x, v_y, v_z from left to right). We observe that the electric fields become dominated by noise.



(a) $D = 32$



(b) $D = 64$

FIG. 16. Energy spectrum at a time of $25\Omega_{c,p}^{-1}$ and $40\Omega_{c,p}^{-1}$, computed with a time step of $\Delta t = 0.005\Omega_{c,p}^{-1}$. The thin vertical dotted lines depict $k = d_p^{-1}$ and $k = d_e^{-1}$. The sloped dashed black lines show power laws of -3, -5/3, and -8. The $D = 64$ calculations show match expectations reasonably well. In contrast, for the $D = 32$ results, while the spectra at earlier times appear within expectation, at later times the results show much flatter spectra. This is due to the noise in the electric field dominating the dynamics.

[1] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. **326**, 96 (2011).
[2] F. Institute, Resources for tensor network algorithms, theory, and software (2023).
[3] J. J. G. Ripoll, Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equation, Quantum **5**, 431 (2021).
[4] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. **33**, 2295 (2011).
[5] E. M. Stoudenmire and S. R. White, Minimally entangled typical thermal state algorithms, New J. Phys. **12**, 055026 (2010).
[6] E. Ye and N. F. G. Loureiro, Quantum-inspired method for solving the Vlasov–Poisson equations, Phys. Rev. E **106**, 035208 (2022).
[7] I. P. McCulloch, From density-matrix renormalization group to matrix product states, J. Stat. Mech. , P10014 (2007).
[8] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete, Unifying time evolution and optimization with matrix product states, Phys. Rev. B **94**, 165116 (2016).
[9] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, Time-evolution methods for matrix-product states, Ann. Phys. **411**, 167998 (2019).
[10] D. Bauernfeind and M. Aichhorn, Time dependent variational principle for tree tensor networks, **8**, 024 (2020).
[11] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, Computation of extreme eigenvalues in higher dimensions using block tensor train format, Comput. Phys. Commun. **185**, 1207 (2014).
[12] I. V. Oseledets and S. V. Dolgov, Solution of linear systems and matrix inversion in the TT-format, SIAM J. Sci. Comput. **34**, A2718 (2012).
[13] S. V. Dolgov and D. V. Savostyanov, Alternating minimal energy method for linear systems in higher dimensions, SIAM J. Sci. Comput **36**, A2248 (2014).
[14] J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, Tech. Rep. (Pittsburgh, PA, 1994).
[15] P. Sonneveld, CGS, A fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comp. **10**, 36 (1989).
[16] N. Gourianov, M. Lubasch, S. Dolgov, *et al.*, A quantum inspired approach to exploit turbulence structures, Nature Comput. Sci. **2**, 30 (2022).
[17] D. R. Durran, *Numerical Methods for Fluid Dynamics: With Applications to Geophysics* (Springer, 2010).