

## Quantum computing and tensor networks for laminate design

### A novel approach to stacking sequence retrieval

Wulff, Arne; Chen, Boyang; Steinberg, Matthew; Tang, Yinglu; Möller, Matthias; Feld, Sebastian

**DOI**

[10.1016/j.cma.2024.117380](https://doi.org/10.1016/j.cma.2024.117380)

**Publication date**

2024

**Document Version**

Final published version

**Published in**

Computer Methods in Applied Mechanics and Engineering

**Citation (APA)**

Wulff, A., Chen, B., Steinberg, M., Tang, Y., Möller, M., & Feld, S. (2024). Quantum computing and tensor networks for laminate design: A novel approach to stacking sequence retrieval. *Computer Methods in Applied Mechanics and Engineering*, 432, Article 117380. <https://doi.org/10.1016/j.cma.2024.117380>

**Important note**

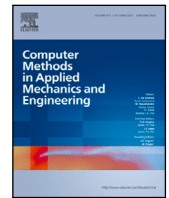
To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

**Copyright**

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**

Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.



# Quantum computing and tensor networks for laminate design: A novel approach to stacking sequence retrieval

Arne Wulff<sup>a</sup>, Boyang Chen<sup>a,\*</sup>, Matthew Steinberg<sup>b</sup>, Yinglu Tang<sup>a</sup>, Matthias Möller<sup>b</sup>, Sebastian Feld<sup>b</sup>

<sup>a</sup> Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands

<sup>b</sup> Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands

## ARTICLE INFO

### Keywords:

Quantum computing  
Tensor networks  
Composite laminates  
Stacking sequence retrieval

## ABSTRACT

As with many tasks in engineering, structural design frequently involves navigating complex and computationally expensive problems. A prime example is the weight optimization of laminated composite materials, which to this day remains a formidable task, due to an exponentially large configuration space and non-linear constraints. The rapidly developing field of quantum computation may offer novel approaches for addressing these intricate problems. However, before applying any quantum algorithm to a given problem, it must be translated into a form that is compatible with the underlying operations on a quantum computer. Our work specifically targets stacking sequence retrieval with lamination parameters, which is typically the second phase in a common bi-level optimization procedure for minimizing the weight of composite structures. To adapt stacking sequence retrieval for quantum computational methods, we map the possible stacking sequences onto a quantum state space. We further derive a linear operator, the Hamiltonian, within this state space that encapsulates the loss function inherent to the stacking sequence retrieval problem. Additionally, we demonstrate the incorporation of manufacturing constraints on stacking sequences as penalty terms in the Hamiltonian. This quantum representation is suitable for a variety of classical and quantum algorithms for finding the ground state of a quantum Hamiltonian. For a practical demonstration, we performed numerical state-vector simulations of two variational quantum algorithms and additionally chose a classical tensor network algorithm, the DMRG algorithm, to numerically validate our approach. For the DMRG algorithm, we derived a matrix product operator representation of the loss function Hamiltonian and the penalty terms. Although this work primarily concentrates on quantum computation, the application of tensor network algorithms presents a novel quantum-inspired approach for stacking sequence retrieval.

## 1. Introduction

Across various engineering disciplines, researchers and practitioners regularly encounter complex and computationally intensive problems [1–3]. The rapidly evolving field of quantum computing could offer crucial breakthroughs in addressing these issues and exceed the capabilities of conventional methods [4–9]. As this technology continues to mature, the potential benefits it offers to complex engineering problems are becoming more apparent [10–23]. However, the challenge lies in accurately identifying those scenarios where quantum computing holds an advantage and in crafting algorithms to harness this potential. To address

\* Corresponding author.

E-mail address: [b.chen-2@tudelft.nl](mailto:b.chen-2@tudelft.nl) (B. Chen).

<https://doi.org/10.1016/j.cma.2024.117380>

Received 13 February 2024; Received in revised form 16 August 2024; Accepted 8 September 2024

Available online 12 September 2024

0045-7825/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

this, initiating the development of quantum algorithms for engineering applications now – in parallel with quantum hardware advancements – is essential. Such an approach positions us to immediately harness the full capabilities of quantum computing in engineering as soon as it becomes available.

The design of laminated composite materials [24,25] is an example of a complex engineering challenge which could benefit from a quantum-based approach. These materials are extensively used in aerospace applications for their high strength-to-weight ratio. Furthermore, these materials consist of many configurable layers, each impacting the overall stiffness. The stiffness characteristics of the whole material can therefore be tailored to the specific requirements of a structure [26,27]. However, determining the optimal arrangement for specific applications from the exponentially large configuration space is a significant challenge, positioning laminated composites as a suitable candidate for quantum computing applications [28,29].

A widely used method in designing composite structures involves a bi-level optimization procedure [30–36]. The initial phase focuses on optimizing the material's thickness and stiffness characteristics, expressed through lamination parameters [37,38]. Subsequently, the second phase involves finding a feasible stacking sequence that aligns with the lamination parameters determined earlier. While the first phase typically employs continuous gradient descent methods, the second stage, stacking sequence retrieval, constitutes an intricate combinatorial optimization problem, particularly when incorporating manufacturing considerations [39–42].

Optimizing stacking sequences for constant stiffness structures has been addressed through various methods, including layer-wise optimization [43,44], branch-and-bound [36,45–47] and beam search [48] techniques. The complexity increases substantially for structures that necessitate variable stiffness, often achieved by dividing the structure into distinct panels [49,50], each requiring a specific stacking sequence and stiffness. Ensuring structural integrity by matching the stacking sequences across different panels, in accordance with blending rules [51–54], adds a further layer of complexity, and adapting the previously mentioned methods to this multi-panel scenario is challenging. Given this complexity, often intensified by the curse of dimensionality, current methods generally rely on meta-heuristics such as genetic algorithms [31–33,35,36,55–57] or particle swarm optimization [57–59], which are frequently used in conjunction with problem-specific methods like stacking sequence tables [60,61] to address these challenges. Despite this progress, the quest for more efficient solutions in stacking sequence retrieval remains a pivotal area of research, indicating the necessity for novel approaches in this complex domain. Quantum computing may offer a unique approach to manage these complex, high dimensional scenarios effectively. The formulation of these algorithms however and their implementation on contemporary quantum hardware remain a challenging tasks. Before tackling complex multi-panel cases, it is essential to gain a deeper understanding of simpler scenarios, where the limitations of existing technology are less constraining. Therefore, and despite the existence of effective stacking sequence retrieval algorithms for single constant-stiffness panels, the foundational work for quantum algorithms must also begin with this simpler case.

In this work, we explore the adaptation of the *stacking sequence retrieval problem* (SSR) with lamination parameters for constant stiffness panels to quantum algorithms. Initially, we formalize the problem as an integer-optimization problem, suitable for translation into a quantum mechanics framework. This involves representing variable configurations of the problem as states in a Hilbert space and deriving a Hamiltonian to embody the problem's loss function. The objective then becomes identifying the ground-state of the Hamiltonian, a task addressable by various quantum algorithms [6,11,62–72]. We also present how manufacturing constraints can be incorporated as penalty terms within the Hamiltonian. As a demonstration of our formalism, we selected two variational quantum algorithms, the quantum approximate optimization algorithm (QAOA) [65,73] and a hardware efficient approach, for which we perform state-vector simulations. Additionally, we chose the density matrix renormalization group algorithm (DMRG), a classical tensor network method effective in finding ground states of many-body systems [74–79]. For this purpose, the Hamiltonian must first be expressed in terms of a matrix product operator (MPO), which is a particular tensor network structure. This enables us to optimize systems as large as 200 plies, by controlling the bond dimension of the states' MPS. For all three algorithms, we conducted tests under various conditions dependent on the algorithm at hand. The optimization results successfully approximated the optimal solutions, validating our Hamiltonian construction and confirming the efficacy of our approach. Furthermore, the notable performance of the DMRG algorithm points to the potential of tensor network methods as a quantum-inspired alternative to conventional stacking sequence retrieval methods.

Following this introduction, Section 2 begins with an overview of laminated composite materials and concludes with the formulation of the specific integer-optimization problem that we employ in the remainder of this work. In Section 3, we map the problem to a quantum state space, and construct the Hamiltonian corresponding to the problem's loss function. We additionally show how to integrate manufacturing constraints as penalty terms in the Hamiltonian. In Section 4, we discuss the algorithms that we selected for numerical demonstrations. In particular, this section discusses how the Hamiltonian must be adapted for QAOA and the DMRG algorithm. The numerical demonstration can be found in Section 5. The paper concludes with Section 6, and summarizes our findings, as well as ramifications for the field of quantum computing in the context of engineering and materials science applications.

## 2. Stacking sequence retrieval as an optimization problem

### 2.1. Laminated composite materials

*Fiber-reinforced composite materials* are heterogeneous materials consisting of fibers incorporated in a supportive matrix, which gives rise to an orthotropic material with distinct stiffness characteristics along the longitudinal and the orthogonal fiber-directions [80]. *Laminated composites* are formed by stacking multiple layers of this material, for which the anisotropic stiffness characteristics can be tailored to the unique strength requirements of a structure by adjusting the orientation of each ply (Fig. 1 a).

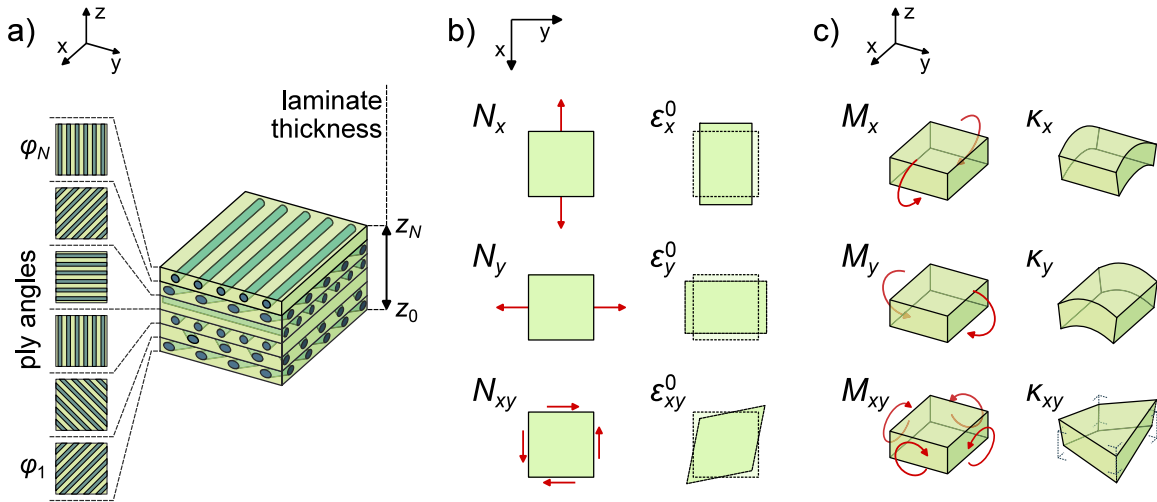


Fig. 1. (a) Diagram of a laminated composite material, which consists of multiple layers that are oriented in different directions. (b) A representation of the stress resultants  $\vec{N}$ , which for isotropic materials result in in-plane deformations  $\vec{\varepsilon}^0$ . (c) A pictographical representation of the moment resultants  $\vec{M}$ , which for isotropic materials result in out-of-plane bending  $\vec{\kappa}$ . Non-diagonal elements in the **ABD**-matrix allow for coupling between all possible stress and moment resultants on the one hand, and all in- and out-of-plane deformations on the other hand.

For laminated composites, the mechanical behavior, to a first-order approximation, is typically characterized using a variation of Hooke's law [81–83]:

$$\begin{bmatrix} \vec{N} \\ \vec{M} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \vec{\varepsilon}^0 \\ \vec{\kappa} \end{bmatrix}. \quad (1)$$

Here, the stress and moment resultants,  $\vec{N}$  and  $\vec{M}$ , represent the loadings within the structure. The deformations are characterized by  $\vec{\varepsilon}^0$  (in-plane bending) and  $\vec{\kappa}$  (out-of-plane bending). Figs. 1 b and c illustrate the stress and moment resultants, along with the deformations, visually demonstrating their effects on the laminate. The **ABD**-matrix analogously functions as the stiffness tensor. Specifically, **A** and **D** define the in-plane and out-of-plane stiffness, respectively. A non-zero **B** indicates a coupling between them, meaning in-plane stresses can result from out-of-plane bending and vice versa. Both vectors  $\vec{N}$  and  $\vec{M}$  along with  $\vec{\varepsilon}^0$  and  $\vec{\kappa}$  consist of three components. Furthermore, matrices **A**, **B** and **D** are of dimensions  $3 \times 3$ .

For a laminate defined by a specific stacking sequence, the **ABD**-matrix is completely determined by the thickness of the material, the angles of the individual plies, and the inherent properties of the selected composite material. Furthermore, these dependencies separate when introducing *lamination parameters*, which distill the complete ply-angle dependence of the **ABD**-matrix into exactly 12 real values  $v_l^A, v_l^B, v_l^D$  with  $l = 1, 2, 3, 4$  [37,38]. Denoting the laminate thickness with  $h$ , the **ABD**-matrix is determined by [42,84]:

$$\begin{aligned} \mathbf{A} &= h(\Gamma_0 + \Gamma_1 v_1^A + \Gamma_2 v_2^A + \Gamma_3 v_3^A + \Gamma_4 v_4^A), \\ \mathbf{B} &= \frac{h^2}{4}(\Gamma_0 + \Gamma_1 v_1^B + \Gamma_2 v_2^B + \Gamma_3 v_3^B + \Gamma_4 v_4^B), \\ \mathbf{D} &= \frac{h^3}{12}(\Gamma_0 + \Gamma_1 v_1^D + \Gamma_2 v_2^D + \Gamma_3 v_3^D + \Gamma_4 v_4^D), \end{aligned} \quad (2)$$

where the intrinsic material properties enter the equation in form of the matrices  $\Gamma_l$  with  $l = 0, 1, 2, 3, 4$ . For a stacking sequence comprising  $N$  plies of equal thickness and ply angles  $\phi_1, \dots, \phi_N \in (-\pi/2, \pi/2]$ , and the distance of ply  $n$  from the mid-plane captured in  $z_n = -N/2 + n$ , the lamination parameters read:

$$\begin{aligned} (v_1^A, v_2^A, v_3^A, v_4^A) &= \frac{1}{N} \sum_{n=1}^N (z_n - z_{n-1}) (\cos(2\phi_n), \sin(2\phi_n), \cos(4\phi_n), \sin(4\phi_n)), \\ (v_1^B, v_2^B, v_3^B, v_4^B) &= \frac{2}{N^2} \sum_{n=1}^N (z_n^2 - z_{n-1}^2) (\cos(2\phi_n), \sin(2\phi_n), \cos(4\phi_n), \sin(4\phi_n)), \\ (v_1^D, v_2^D, v_3^D, v_4^D) &= \frac{4}{N^3} \sum_{n=1}^N (z_n^3 - z_{n-1}^3) (\cos(2\phi_n), \sin(2\phi_n), \cos(4\phi_n), \sin(4\phi_n)). \end{aligned} \quad (3)$$

The matrices  $\Gamma_k$  are constructed with the Tsai–Pagano material invariants  $U_1, \dots, U_5 \in \mathbb{R}$  [37]:

$$\Gamma_0 = \begin{bmatrix} U_1 & U_4 & 0 \\ U_4 & U_1 & 0 \\ 0 & 0 & U_5 \end{bmatrix}, \quad \Gamma_1 = \begin{bmatrix} U_2 & 0 & 0 \\ 0 & -U_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \Gamma_2 = \begin{bmatrix} 0 & 0 & U_2/2 \\ 0 & 0 & U_2/2 \\ U_2/2 & U_2/2 & 0 \end{bmatrix} \quad (4)$$

$$\Gamma_3 = \begin{bmatrix} U_3 & -U_3 & 0 \\ -U_3 & U_3 & 0 \\ 0 & 0 & -U_3 \end{bmatrix}, \quad \Gamma_4 = \begin{bmatrix} 0 & 0 & U_3 \\ 0 & 0 & -U_3 \\ U_3 & -U_3 & 0 \end{bmatrix}.$$

These material invariants depend only on the longitudinal, transverse, and shear moduli, as well as the Poisson's ratio of the composite.

In numerous applications, particularly in aerospace structural design, materials are required to be as lightweight as possible while simultaneously possessing the strength to withstand forces and stresses under a variety of expected loading conditions. A typical objective involves minimizing the thickness or, equivalently, the number of plies in composite materials. This minimization is constrained by the expected stress and moment resultants  $\bar{N}$  and  $\bar{M}$ , alongside predefined failure criteria expressed through inequalities involving stresses, deformations, or the ABD-matrix. The optimization process becomes highly complex due to the non-linear relationship between the laminate's structural behavior and the ply angles, which serve as design variables in the optimization and typically range from tens to hundreds in number [84]. To effectively navigate this complexity, a bi-level design strategy is commonly employed for laminated composite materials, involving two distinct phases of optimization [30–36]. The first phase optimizes for the laminate thickness  $h$  and the stiffness characteristics, represented by lamination parameters  $v_l^A$ ,  $v_l^B$  and  $v_l^D$ , such that they meet predefined strength criteria. These design variables are typically assumed to be continuous, such that gradient descent methods can be employed effectively. The subsequent phase, termed *stacking sequence retrieval* (SSR), seeks a ply-angle sequence that approximates the lamination parameters derived from the first phase. The laminate thickness  $h$  is incorporated into the second phase as the total number of plies  $N$  in the laminate. In this work, we will focus on this second phase of stacking sequence retrieval.

Manufacturing constraints typically dictate the allowed ply-angle sequences in a laminate. Primary among these is the confinement of ply angles to a discrete set, with a common selection being  $\{0^\circ, +45^\circ, 90^\circ, -45^\circ\}$ . Alternatives can also be found, with divisions of the interval  $(-90^\circ, 90^\circ)$  at intervals of  $22.5^\circ$  or  $15^\circ$ . It is worth noting that fiber orientations, and consequently the lamination parameters, demonstrate a periodicity of  $180^\circ$ . Other prevalent manufacturing constraints include [39–42]:

- **Symmetric laminates:** The laminate exhibits symmetry about its midplane. Hence, plies  $n$  and  $N - n + 1$  share the same ply angle, leading to null coupling lamination parameters  $v_k^B$  and stiffness matrix  $\mathbf{B}$ .
- **Disorientation constraint:** The angular difference between adjacent plies is limited, often to  $45^\circ$ .
- **Balanced laminates:** Except for ply-angles  $0^\circ$  or  $90^\circ$ , the number of plies with angles  $\theta$  and  $-\theta$  are equal. This has the effect, that  $v_2^A$  and  $v_4^A$  vanish.
- **Contiguity constraint:** The number of consecutive same-angle plies is limited, for instance to 5 plies with the same orientation.
- **10%-rule:** Specific ply angles must constitute a minimum of 10% of the laminate. This rule is frequently applied to angles of  $0^\circ$ ,  $90^\circ$  and  $\pm 45^\circ$ .

For a laminate consisting of  $N$  plies and a set of  $d$  discrete angles, the solution space encompasses  $d^N$  potential configurations. This exponential explosion renders direct search or enumeration methods computationally prohibitive. Moreover, the imposition of manufacturing constraints further complicates the search landscape. Various strategies have been advanced to navigate this complex space [24,26,48,85], however, the optimization of more complicated structures remains a formidable challenge.

Lately, there has been a spotlight on breakthroughs in quantum computing, a technology that might offer new tools to address this complexity [5,7,86–88]. However, one must map the presently-described problem to a representation amenable to quantum information-theoretic methods. We start with formulating stacking sequence retrieval as an integer-optimization problem.

## 2.2. Formulating stacking sequence retrieval as an integer-optimization problem

In general, it is advantageous for optimization algorithms to be presented with a version of the problem that abstracts away all but the essential information. For our purposes, this level of abstraction is necessary to adapt the problem for quantum computing architectures. In this section, we present our redefinition of the problem that forms the basis for our analysis in this work.

The discrete nature of the ply angles in laminated composites is well-suited to be represented in terms of integer design variables, a method commonly employed in various stacking sequence retrieval methods [36,56,89]. Accordingly, we define the  $d$  discrete angles  $\theta = \{\theta_1, \theta_2, \dots, \theta_d\}$ , allowing each angle  $\theta_s$  to be represented by its corresponding label  $s \in \{1, \dots, d\}$ , effectively using these labels as integer design variables. Consequently, we can identify the ply angle sequence  $\vec{\theta} = (\theta_{s_1}, \theta_{s_2}, \dots, \theta_{s_N})$  with an ordered list of integers  $\vec{s} = (s_1, s_2, \dots, s_N)$ . Here we use  $\theta$  instead of the  $\phi$  from Eq. (3), to convey that we now explicitly choose the ply angles from the discrete set  $\theta$ . In the following, we denote the set of all possible stacking sequences as  $S = \{1, \dots, d\}^N$ , such that  $\vec{s} \in S$ . Considering Eq. (3), the lamination parameters can then be written in a common general form:

$$v_l^X(s_1, \dots, s_N) = \sum_{n=1}^N \alpha_n^X f_l(s_n), \quad (5)$$

where  $X = A, B, D$  and  $l = 1, 2, 3, 4$ , and the functions  $f_l : \{1, \dots, d\} \rightarrow [-1, 1]$  defined as:

$$\begin{aligned} f_1(s) &= \cos(2\theta_s), & f_2(s) &= \sin(2\theta_s), \\ f_3(s) &= \cos(4\theta_s), & f_4(s) &= \sin(4\theta_s). \end{aligned} \quad (6)$$

In this framework, a manufacturing constraint can be realized as a Boolean function  $c : S \rightarrow \{\text{True}, \text{False}\}$  on the lists of integers, where  $c(\vec{s}) = \text{True}$  exactly when ply-angle sequence  $\vec{\theta} = (\theta_{s_1}, \dots, \theta_{s_N})$  satisfies the constraint. The optimization towards target

lamination parameters can then be written as an integer-optimization problem. We choose an abstracted formulation of the problem that does not include the ply angles  $\theta_s$  but assumes their impact on the loss function to be encoded in the functions  $f_l$  that are defined on the according indices  $s = \{1, \dots, d\}$ :

**Stacking sequence retrieval problem (SSR):** Given

- Positive integers  $N, d \in \mathbb{N}$ ,
- The set of stacking sequences  $S = \{1, \dots, d\}^N$ ,
- Sets of weights  $\{\alpha_n^X\}_n$  for  $n = 1, \dots, N$  and  $X = 1, \dots, N_\alpha$  for some  $N_\alpha \in \mathbb{N}$ ,
- Functions  $f_l : \{1, \dots, d\} \rightarrow \mathbb{R}$  for  $l = 1, \dots, N_f$  for some  $N_f \in \mathbb{N}$ ,
- A set of constraints  $C = \{c_k : S \rightarrow \{\text{True}, \text{False}\}\}$  on the solution space, where  $k$  enumerates the constraints,
- Target values  $\bar{\xi} \in \mathbb{R}^{N_\alpha N_f}$ ,

find a stacking sequence  $\vec{s} = (s_1, \dots, s_N) \in S$  with  $c(\vec{s}) = \text{True}$  for all  $c \in C$  that minimizes the loss function:

$$H(s_1, \dots, s_N) = \sum_{X,l} \left( \sum_{n=1}^N \alpha_n^X f_l(s_n) - \bar{\xi}_l^X \right)^2. \quad (7)$$

Note, that for the loss function, we use the mean-square-error (MSE) rather than the root-mean-square-error (RMSE), due to its natural implementation in the quantum context. Nevertheless, the RMSE can prove useful to evaluate the quality of solutions, as it represents the Euclidean distance of the solution to the target parameters in lamination parameter space. The defined integer-optimization problem is more general than the original stacking sequence retrieval problem, and most of the following discussion does not depend on explicit definitions of  $\alpha_n^X$ ,  $N_\alpha$ ,  $f_l$  and  $N_f$ . However, we will generally assume the previously defined case of  $N_f = 4$ ,  $X = A, B, D$  such that  $N_\alpha = 3$ , and  $\alpha_n^X$  and  $f_l$  defined according to the lamination parameters in Eq. (3).

### 3. A formulation of the SSR problem in a quantum state space

In this section, we discuss how the stacking sequence retrieval problem can be represented with linear algebra concepts inline with quantum information theoretic methods. In particular, we represent stacking sequences as vectors in a complex-valued Hilbert space, and treat functions acting on stacking sequences, such as the loss function or constraints, as Hermitian operators on this space. The presented approach is applicable to various prevalent quantum computing methods, such as variational quantum algorithms (VQA) [64–67] and quantum annealing [68–70,90–92]. For readers unfamiliar with quantum computation, we recommend the book [7].

#### 3.1. SSR in terms of quantum states and operators

In the defined integer optimization problem in Section 2.2, the three primary elements are the stacking sequences  $\vec{s} \in S$ , the loss function  $H(\vec{s})$  in Eq. (7), and the constraints  $c \in C$ . Initially, we focus on the stacking sequences and loss function, and address the constraints later on.

We encode stacking sequences  $\vec{s} \in S$  using a basis-encoding within a composite complex-valued Hilbert space  $\mathcal{H} = \bigotimes_{n=1}^N \mathcal{H}_n$ . Here, each subsystem  $\mathcal{H}_n$  with dimension  $\dim(\mathcal{H}_n) = d$  represents a single ply, and consequently the composite system for the entire stacking sequence has dimension  $\dim(\mathcal{H}) = d^N$ . Accordingly, each basis vector  $|\vec{s}\rangle \in \mathcal{H}$  represents an individual stacking sequence  $\vec{s} \in S$ . These vectors are orthonormal, meaning for any two distinct sequences  $\vec{s}, \vec{t} \in S$ , their inner product satisfies:

$$\langle \vec{s} | \vec{t} \rangle = \delta_{\vec{s}, \vec{t}}. \quad (8)$$

To model the loss function  $H(\vec{s})$  mentioned in Eq. (7), we introduce a Hermitian operator  $\hat{H}$  in Hilbert space  $\mathcal{H}$ , termed the loss function Hamiltonian. This Hamiltonian is defined to be diagonal in the basis of stacking sequences with eigenvalues corresponding to the loss function evaluations  $\hat{H} |\vec{s}\rangle = H(\vec{s}) |\vec{s}\rangle$ :

$$\hat{H} = \sum_{\vec{s} \in S} H(\vec{s}) |\vec{s}\rangle \langle \vec{s}|. \quad (9)$$

Thus, finding the solution to the integer optimization problem is equivalent to identifying the ground state of this Hamiltonian, ignoring any additional constraints  $c \in C$  for the moment.

Due to its generality, the basis-state encoding together with the loss function Hamiltonian is applicable to various quantum methods. However, the specific form of the Hamiltonian and its application in quantum algorithms significantly influence the practical implementation. As we will discuss for the QAOA algorithm in Section 4.1.2, some algorithms require implementing the exponential of the Hamiltonian using the available gates on the quantum computer [65]. For feasible implementation with a limited number of quantum gates, the Hamiltonian should exhibit a low-degree polynomial expansion in terms of local operators on each subsystem. As a further example, quantum annealing requires the polynomial expansion of the Hamiltonian to be quadratic, since the individual terms are implemented as interactions between pairs of subsystems [68–70]. The choice of mean-square-error for the

loss function  $H(\vec{s})$  in Eq. (7) naturally results in a quadratic expression in the single-ply functions  $f_l : \{1, \dots, d\} \rightarrow \mathbb{R}$ , aligning with algorithmic requirements.

To make the polynomial expansion of the loss function and the according Hamiltonian explicit, as well as for later convenience, we absorb the target parameters  $\xi_l^X$  into the sum over the ply indices  $n$ , such that the loss function takes the form:

$$H(\vec{s}) = \sum_{X,l} \left( \sum_{n=1}^N H_{X,l}^{[n]}(s_n) \right)^2. \quad (10)$$

where the target parameters are incorporated into the ply-angle-dependent terms  $H_{X,l}^{[n]} : \{1, \dots, d\} \rightarrow \mathbb{R}$ . For the lamination parameters  $X = A, B, D$ , the summation over the absolute values of the weights is  $\sum_{n=1}^N |\alpha_n^X| = 1$ . Therefore, a natural choice for the terms  $H_{X,l}^{[n]}$  distributes the target parameters according to the weights:

$$H_{X,l}^{[n]}(s) = \alpha_n^X (f_l(s) - \text{sign}(\alpha_n^X) \xi_l^X). \quad (11)$$

For  $X = A, D$ , the sign function is redundant since all weights  $\alpha_n^A, \alpha_n^D > 0$  are positive for all ply-indices  $n$ .

Accordingly, the full loss function Hamiltonian is formulated to be quadratic in local operators  $\hat{H}_{X,l}^{[n]}$  on the individual subsystems  $n$ :

$$\hat{H} = \sum_{X,l} \left( \sum_{n=1}^N \hat{H}_{X,l}^{[n]} \right)^2, \quad (12)$$

where the local operators defined through the local functions  $H_{X,l}^{[n]}(s)$  as:

$$\hat{H}_{X,l}^{[n]} = \sum_{t=1}^d H_{X,l}^{[n]}(t) |t\rangle\langle t|_n. \quad (13)$$

Here,  $|t\rangle\langle t|_n$  denotes the outer product of state  $|t\rangle \in \mathcal{H}_n$  and satisfies  $(|t\rangle\langle t|_n) |\vec{s}\rangle = \delta_{t,s_n} |\vec{s}\rangle$ .

### 3.2. Enforcing manufacturing constraints

In addition to optimizing for the target lamination parameters, the stacking sequences must also adhere to manufacturing constraints, such as the ones listed in Section 2. Here, we examine how these constraints, generalized to larger classes of constraints, can be enforced.

Some constraints can be directly incorporated into the representation of the stacking sequences. For instance, to enforce symmetry around the midplane in a laminate,  $s_n = s_{N-n+1}$ , it suffices to model half of the stack and then implicitly mirror this to the other half. Due to the symmetry of the weights, the  $A$  and  $D$  lamination parameters in Eq. (3) can be adapted to this case, by only summing the upper half  $n = N/2 + 1, \dots, N$  of the stack and doubling the result. For a more convenient expression, we can reassign the ply indices  $n$  to start at the midplane by substituting  $n \rightarrow n - N/2$ , such that  $z_n = n$ . We also redefine the ply number  $N$  to just count half of the stack  $N/2 \rightarrow N$ . The reformulated lamination parameters only differ from Eq. (3) by the definition of  $z_n$  and the factor 4 preceding the summation in the  $D$ -parameters:

$$\begin{aligned} (v_1^A, v_2^A, v_3^A, v_4^A) &= \frac{1}{N} \sum_{n=1}^N (z_n - z_{n-1}) (\cos(2\theta_n), \sin(2\theta_n), \cos(4\theta_n), \sin(4\theta_n)) \\ (v_1^D, v_2^D, v_3^D, v_4^D) &= \frac{1}{N^3} \sum_{k=1}^N (z_n^3 - z_{n-1}^3) (\cos(2\theta_n), \sin(2\theta_n), \cos(4\theta_n), \sin(4\theta_n)) \\ z_n &= n \end{aligned} \quad (14)$$

As before, the weights:

$$\alpha_n^A = \frac{1}{N}, \quad \alpha_n^D = \frac{n^3 - (n-1)^3}{N^3} \quad (15)$$

sum to 1. The  $B$ -parameters vanish for symmetric laminates and can thus be neglected.

Enforcing the remaining constraints  $c : S \rightarrow \{\text{True}, \text{False}\}$  in Section 2 can be achieved by incorporating a penalty function  $H_c : S \rightarrow \mathbb{R}_{\geq 0}$  into the loss function:

$$H_{\text{total}}(\vec{s}) = H(\vec{s}) + H_c(\vec{s}), \quad (16)$$

where  $H_c(\vec{s}) > 0$  if the constraint is violated for stacking sequence  $\vec{s}$ ,  $c(\vec{s}) = \text{False}$ , and  $H_{\text{penalty}}(\vec{s}) = 0$  for valid states,  $c(\vec{s}) = \text{True}$ . This effectively turns the constraint optimization problem with loss function  $H$  into an unconstrained optimization problem with loss function  $H_{\text{total}}$  [93–95]. Often, a scalar factor is multiplied to the penalty function  $H_c$ , in order to control the magnitude of the penalty. However, this factor can also be regarded as part of the definition of the penalty function  $H_c$ . Accordingly, we include tunable parameters in the definition of the following penalty functions that we define for the different constraints in section Section 2.



As with the loss function  $H$ , the penalty function  $H_c$  is also transformed into a linear operator  $\hat{H}_c$ :

$$\hat{H}_c = \sum_{\vec{i} \in S} H_c(\vec{i}) |\vec{i}\rangle\langle\vec{i}| \quad (17)$$

for its quantum representation. Consequently, the total Hamiltonian takes the form

$$\hat{H}_{\text{total}} = \hat{H} + \hat{H}_c. \quad (18)$$

As discussed for the Hamiltonian  $\hat{H}$ , the interactions between subsystems in  $\hat{H}_c$  significantly influence the performance of various quantum algorithms. In the following, we explore possible penalty terms for the remaining manufacturing constraints apart from the symmetry constraint in Section 2. For this purpose, we examine general categories of constraints that encompass the listed constraints.

### 3.2.1. The disorientation constraint as nearest-neighbor coupling

The simplest constraint listed in Section 2 is the disorientation constraint, which limits the ply angle difference between neighboring plies. In general, a constraint with nearest neighbor coupling can be enforced with a penalty function  $\eta : \{1, \dots, d\}^2 \rightarrow \mathbb{R}_{\geq 0}$  on each pair of neighboring subsystems  $n$  and  $n+1$ :

$$\eta(s_n, s_{n+1}) = \begin{cases} \gamma, & \text{if } (s_n, s_{n+1}) \text{ violates the constraint,} \\ 0, & \text{else,} \end{cases} \quad (19)$$

where the adjustable parameter  $\gamma > 0$  is the added penalty for a single constraint violation. The function  $\eta$  is used to define an operator:

$$\hat{H}_{nn}^{[n,n+1]} = \sum_{t,t'=1}^d \eta(t, t') |tt'\rangle\langle tt'|_{n,n+1}. \quad (20)$$

As before, the indices of  $|tt'\rangle\langle tt'|_{n,n+1}$  signify the local action of the operator on subsystems  $\mathcal{H}_n$  and  $\mathcal{H}_{n+1}$ . Subsequently, these operators are summed over all nearest-neighbor pairs to yield the total penalty:

$$\hat{H}_{nn} = \sum_{n=1}^{N-1} \hat{H}_{nn}^{[n,n+1]}. \quad (21)$$

The penalty operator exhibits strong parallels with quantum systems that feature nearest-neighbor coupling, such as spin lattice systems, and extensive research has been conducted on these systems [6,68,91,96–102].

### 3.2.2. The contiguity constraint as $k$ -local coupling

The contiguity constraint restricts the number of consecutive same-angle plies to be equal or less than some small integer  $N_{\text{same}}$ . A suitable penalty function evaluates  $N_{\text{same}} + 1$  neighboring plies, and adds a penalty, if all plies have the same angle. The contiguity constraint is therefore a special case of a  $k$ -local constraint with  $k = N_{\text{same}} + 1$ , that encompasses interactions in a neighborhood of  $k$  subsystems in a linear chain. The treatment of nearest-neighbor interactions above can be extended to accommodate this case, by defining:

$$\eta(s_n, s_{n+1}, \dots, s_{n+k-1}) = \begin{cases} \gamma, & \text{if } (s_n, s_{n+1}, \dots, s_{n+k-1}) \text{ violates the constraint,} \\ 0, & \text{else,} \end{cases} \quad (22)$$

on  $k$  consecutive subsystems. As before, the parameter  $\gamma$  controls the magnitude of the added penalty. The corresponding  $k$ -local operator is given by:

$$\hat{H}_{k\text{-local}}^{[n, \dots, n+k-1]} = \sum_{t_1, \dots, t_k=1}^d \eta(t_1, \dots, t_k) |t_1 \dots t_k\rangle\langle t_1 \dots t_k|_{n, \dots, n+k-1}. \quad (23)$$

The total penalty takes the form:

$$\hat{H}_{k\text{-local}} = \sum_{n=1}^{N-k+1} \hat{H}_{k\text{-local}}^{[n, \dots, n+k-1]}. \quad (24)$$

As discussed before, algorithms such as QAOA (Section 4.1.2) or quantum phase estimation (QPE) [71,72] require the implementation of interactions as operations. These higher-order terms typically require decomposition into two-qubit operations, leading to a significant increase in the total number of operations. Similarly, quantum annealers, which inherently support only second-order interactions, necessitate the transformation of higher-order terms into quadratic forms [68–70,90–92]. This often involves the introduction of auxiliary qubits as additional variables [103–106]. Nonetheless, with advancements in quantum technology, future devices could be capable of managing these heightened operational demands, especially if  $k$  remains within a manageable range.



### 3.2.3. Balanced laminates and ensuring the same ply count for two distinct states

The constraint for balanced laminates requires that the ply angle sequence contains an equal number of plies with angle  $+\theta$  and angle  $-\theta$ , provided  $\theta$  is neither  $0^\circ$  nor  $90^\circ$ . This represents a special case of constraints where two states, denoted with  $t$  and  $t'$ , appear an equal number of times in the stacking sequence:

$$\sum_{n=1}^N \delta_{s_n,t} = \sum_{n=1}^N \delta_{s_n,t'}. \quad (25)$$

To implement such a penalty function, we can take the squared difference, similar to the loss function of the lamination parameters in Eq. (7):

$$H_{\text{balanced}}(\vec{s}) = \gamma \left( \sum_{n=1}^N \delta_{s_n,t} - \sum_{n=1}^N \delta_{s_n,t'} \right)^2 = \gamma \left( \sum_{n=1}^N (\delta_{s_n,t} - \delta_{s_n,t'}) \right)^2. \quad (26)$$

As before, the magnitude of the penalty can be adjusted with the parameter  $\gamma \in \mathbb{R}_{>0}$ . In order to define the corresponding operator, we replace a Kronecker delta  $\delta_{s_n,t}$  with the dyadic product  $|t\rangle\langle t|_n$  on subsystem  $H_n$ , which turns the classical penalty function into an operator:

$$\hat{H}_{\text{balanced}} = \gamma \left( \sum_{n=1}^N (|t\rangle\langle t|_n - |t'\rangle\langle t'|_n) \right)^2, \quad (27)$$

where  $|t\rangle\langle t|_n - |t'\rangle\langle t'|_n$  is a local operator acting on subsystem  $n$ . This leads to a structure analogous to the Hamiltonian in Eq. (12) for the lamination parameters, featuring interactions between each pair of subsystems. Consequently, the earlier considerations regarding quantum algorithms are also relevant here.

### 3.2.4. The 10%-rule and constraints on the ply count for a given state

The last manufacturing constraint listed in Section 2 is the 10%-rule, which necessitates that certain ply angles, typically  $0^\circ$ ,  $90^\circ$  or  $\pm 45^\circ$ , make up at least 10% of the plies. This constitutes a special case of constraints that, for some state  $t$ , require at least  $N_t$  plies to be in state  $t$ . A suitable penalty function can be defined as:

$$H_{10\%-rule}(\vec{s}) = \begin{cases} 0, & \text{if } \sum_{n=1}^N \delta_{s_n,t} \geq N_t, \\ p\left(N_t - \sum_{n=1}^N \delta_{s_n,t}\right), & \text{else,} \end{cases} \quad (28)$$

where the function  $p(x)$  characterizes the behavior for constraint violations. Examples include constant,  $p(x) = \gamma$ , linear  $p(x) = \gamma x$  or power-law  $p(x) = \gamma x^\alpha$  behavior for some  $\alpha, \gamma > 0$ . The according operator is defined as:

$$\hat{H}_{10\%-rule} = \sum_{\vec{i} \in S} H_{10\%-rule}(\vec{i}) |\vec{i}\rangle\langle \vec{i}| \quad (29)$$

As before, decomposing the operator into local operators provides insight into dependencies between individual subsystems. As we show in the appendix A.1, any penalty operator suitable for the 10%-rule contains terms of order  $N - N_t + 1$  or larger. For the 10%-rule specifically,  $N_t = 0.1N$ , the operator contains terms that include roughly 90% of all subsystems. These high-order interactions present a notable challenge for many algorithms. Further investigation is vital to determine if and how these interactions can be efficiently managed in quantum algorithms.

## 4. Selected algorithms in the context SSR

In the previous section, we discussed the representation of the stacking sequence retrieval problem using a basis-state encoding, along with a Hamiltonian that encodes the loss function and penalties. This approach is general and applicable to a broad range of algorithms. This is especially true for *variational quantum algorithms* (VQA) [64–67,88] and quantum annealing [68–70,90–92], which are anticipated to be among the first methods to find real-world applications on near-term quantum hardware. Especially VQA are a diverse family of algorithms with various state-space parametrization, measurement and optimization strategies [66,67,88], with ongoing research rapidly extending the amount of available methods. Identifying the optimal strategy for stacking sequence retrieval and integrating novel approaches will be crucial for demonstrating the effectiveness of quantum computation in this area. Here, we restrict the discussion to basic standard algorithms in order to illustrate how the presented quantum representation can be utilized for quantum computation. Specifically, we selected two different VQA for which we conduct state-vector simulations: a hardware efficient approach and the standard *quantum approximate optimization algorithm* (QAOA) [65,73]. Additionally, we utilize a tensor-network algorithm, the *density matrix renormalization group* (DMRG) algorithm [74–79,107], which is a classical variational algorithm closely related to one of the VQAs under investigation.

What all these variational algorithms share is a common objective: to minimize the expectation value of the Hamiltonian:

$$|\psi_{\text{opt}}\rangle = \arg \min_{|\psi\rangle \in \mathcal{H}} \langle \psi | \hat{H} | \psi \rangle. \quad (30)$$

The resulting state  $|\psi_{\text{opt}}\rangle$  represents either the basis state corresponding to the optimal solution or – in cases of multiple optimal configurations with identical minimal loss – a potential superposition of optimal basis states. Although this objective unifies variational algorithms, they are differentiated by the specific strategies used to traverse the state space and achieve minimization.

In our demonstrations, we employ the four commonly used angles:

$$\theta_1 = 0^\circ, \quad \theta_2 = +45^\circ, \quad \theta_3 = 90^\circ, \quad \theta_4 = -45^\circ. \quad (31)$$

Assuming a symmetric laminate, we utilize the corresponding lamination parameters as detailed in Eq. (14). This section explores the implications of various constraints outlined in Section 3.2 within the context of these algorithms. For our numerical demonstrations in Section 5, we include a disorientation constraint that limits the ply-angle difference between adjacent plies to  $45^\circ$  or less. This constraint is breached when neighboring plies are  $\theta_1$  and  $\theta_3$ , or  $\theta_2$  and  $\theta_4$ , and is enforced via a penalty term as discussed in Section 3.2.1.

In the remainder of this section, we delve into algorithm-specific considerations. To conclude, we discuss the similarities between the chosen hardware-efficient approach and DMRG, in contrast to QAOA. The results of our numerical demonstrations are detailed in the following Section 5.

#### 4.1. Variational quantum algorithms

*Variational quantum algorithms (VQA)* employ parameterized quantum circuits to define a subspace of the Hilbert space  $\mathcal{H}$  using real, continuous parameters [66,67]. By configuring a quantum circuit with  $N_p$  parameters  $\vec{p} \in \mathbb{R}^{N_p}$ , we can prepare the state  $|\psi(\vec{p})\rangle \in \mathcal{H}$  on a quantum computer. This setup enables the repeated preparation and measurement of this state to estimate the Hamiltonian's expectation value,  $\langle \hat{H} \rangle_\psi(\vec{p}) = \langle \psi(\vec{p}) | \hat{H} | \psi(\vec{p}) \rangle$ . The minimization itself is carried out by a classical optimizer, which iteratively selects parameters  $\vec{p}$  and executes calls to the function  $\langle \hat{H} \rangle_\psi(\vec{p})$  based on its specific strategy. Leveraging limited quantum resources for essential computations while relying on classical systems for optimization makes VQAs particularly suitable for near-term quantum devices.

Most contemporary quantum computers are based on qubits, which are characterized by two basis states  $|0_q\rangle$  and  $|1_q\rangle$ . To distinguish the qubit states from the ply states, we apply a subscript  $q$  to denote qubit states. The basis states  $|\vec{s}\rangle \in \mathcal{H}$  therefore need to be represented in terms of qubit states. In the case of the above mentioned 4 ply-angles, a subspace  $\mathcal{H}_n$  can be implemented as 2 qubits using a binary encoding, which we specifically choose to be:

$$|1\rangle \equiv |00_q\rangle, \quad |2\rangle \equiv |01_q\rangle, \quad |3\rangle \equiv |11_q\rangle, \quad |4\rangle \equiv |10_q\rangle. \quad (32)$$

In this ply-state encoding, flipping one of the qubits corresponds to a ply-angle change of  $+45^\circ$  or  $-45^\circ$ , while changing both qubits results in a  $90^\circ$  rotation.

For a general non-diagonal Hamiltonian, the expectation value is typically determined by expressing the Hamiltonian using Pauli operators  $\hat{X}$ ,  $\hat{Y}$ , and  $\hat{Z}$  applied to the qubits:

$$\hat{X} = |0_q\rangle\langle 1_q| + |1_q\rangle\langle 0_q|, \quad \hat{Y} = i(|0_q\rangle\langle 1_q| - |1_q\rangle\langle 0_q|), \quad \hat{Z} = |0_q\rangle\langle 0_q| - |1_q\rangle\langle 1_q|. \quad (33)$$

The expectation value for individual terms is measured using appropriate basis rotations. Due to linearity, the total expectation value equates to the sum of these individual contributions. In our specific scenario, however, the Hamiltonian is diagonal in the basis of stacking sequences and, by extension, in the qubit basis. This configuration allows for straightforward repeated measurements of a state and aggregation of loss function values from individual outcomes, divided by the total number of counts. Although this simplifies the measurement process, formulating the Hamiltonian in terms of Pauli operators remains a necessity for specific algorithms, as demonstrated for QAOA in Section 4.1.2.

As mentioned earlier, VQAs exhibit significant variability in state preparation circuits – ranging from problem-inspired [64,65, 108–110] to hardware efficient [111,112] – but also in state measurements and optimization strategies. Recent developments includes adaptive approaches to enhance the performance of the algorithm [113–118], and identifying VQAs that are optimally suited for problems like stacking sequence retrieval will necessitate extensive research and will likely hinge on ongoing advancements in this dynamic field. In this work, we focus on demonstrating our methodology using two specific standard VQAs: a hardware-efficient approach and the standard QAOA.

##### 4.1.1. A hardware-efficient approach

Among variational quantum algorithms, hardware-efficient approaches are distinguished by their utilization of short-depth quantum circuits. These circuits efficiently utilize available gates within a specific quantum computing architecture to parameterize a substantial portion of the state space. In this study, we employ a state preparation circuit, as illustrated in Fig. 2, which incorporates  $\hat{R}_{y(p)}$  single-qubit rotations to facilitate dependency on the continuous parameters  $p$ , and CNOT gates to generate entanglement among the qubits. Typically, in hardware-efficient approaches, the state preparation circuit features a block of gates that is repeated a certain number of times. Adjusting the number of repetitions modulates the number of parameters, enabling exploration of a larger subspace at the cost of increasing the complexity of optimization. A particular feature of our selected state preparation circuit is the exclusive connection of neighboring qubits by the CNOT gates. This particular state preparation circuit has previously been employed, for instance in [116].

A significant obstacle in hardware-efficient VQA is the optimization of a large number of parameters [119] and several strategies have been developed to tackle this issue. These include techniques that optimize only a subset of parameters at a time, such as

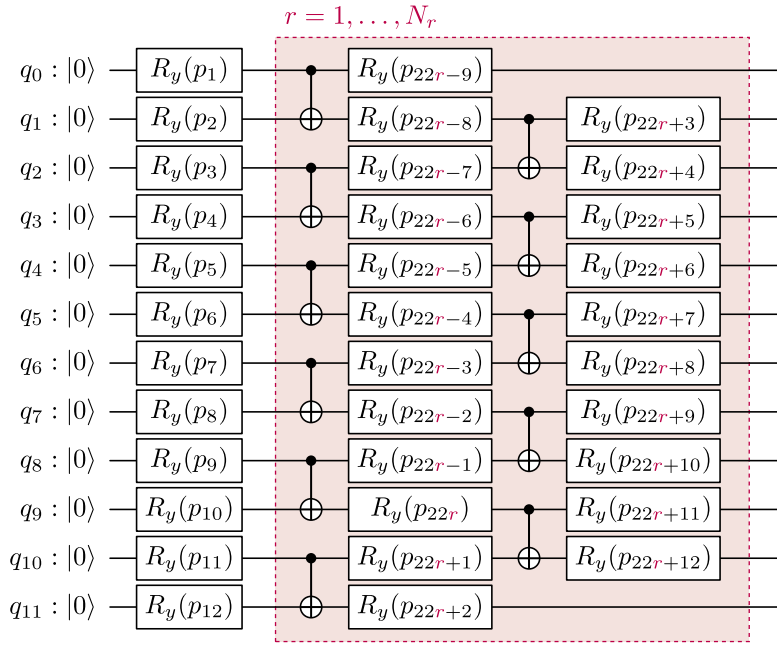


Fig. 2. Parameterized quantum circuit for hardware-efficient approach.

optimizing different segments of the parameterized circuit sequentially [120–122]. In our approach, we adopt an optimization strategy that sequentially optimizes two neighboring qubits at a time, starting either from the qubits corresponding to plies  $n = 1$  or  $n = N$ . This sequence progresses one qubit at a time, culminating after  $N - 1$  pairwise optimizations. The process involves multiple sweeps of local optimizations across the entire stack until the optimization converges. This method bears notable similarities to the DMRG algorithm, which we discuss subsequent to the QAOA.

#### 4.1.2. QAOA

The most recognized variational quantum algorithm for combinatorial optimization is the *quantum approximate optimization algorithm* (QAOA) [65,123], which employs two distinct types of operators in its state preparation circuit. The problem unitary is defined by the exponential of the Hamiltonian:

$$U_C(\gamma) = \exp(-i\gamma\hat{H}), \quad (34)$$

and the mixer unitary is articulated through the exponential of  $\hat{X}$  operators applied to all  $N_q = 2N$  qubits:

$$U_M(\beta) = \exp\left(-i\beta \sum_{j=1}^{N_q} \hat{X}_j\right), \quad (35)$$

where  $\hat{X}_j$  represents the operator  $\hat{X}$  on qubit  $j$ . Each operator introduces a single parameter  $\beta \in \mathbb{R}$  or  $\gamma \in \mathbb{R}$  into the circuit. The entire circuit begins by applying single-qubit rotations to each qubit to prepare the state  $|+\rangle^{\otimes N_q}$ , where  $|+\rangle = \frac{1}{\sqrt{2}}(|0_q\rangle + |1_q\rangle)$ . Subsequently, a defined number of  $N_r$  repetitions of the problem and mixer unitaries are alternately executed on this initial state:

$$|\psi(\vec{\beta}, \vec{\gamma})\rangle = U_M(\beta_{N_r})U_C(\gamma_{N_r}) \dots U_M(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes N_q}. \quad (36)$$

Thus, the complete quantum circuit that prepares  $|\psi(\vec{\beta}, \vec{\gamma})\rangle$  comprises  $2N_r$  continuous parameters that need optimization.

To implement the state preparation circuit on a quantum computer, the problem and mixer unitaries must be converted into a sequence of available gates. As common practice, we focus on the single-qubits rotations  $\hat{R}_x$ ,  $\hat{R}_y$ , and  $\hat{R}_z$ , and the CNOT gate. Given that shorter circuits are quicker to evaluate and less susceptible to errors, the total number of gates in the decomposition significantly influences the performance of the algorithm. The mixer unitary, which is always the same for standard QAOA, is typically decomposed into a layer of parallel single-qubit rotations and therefore only has a minor impact on the circuit depth.

Conversely, the specific form of the Hamiltonian significantly affects the gate count required for the problem unitary. Initially, we analyze the loss function Hamiltonian in Eq. (12), which is expanded to take the form:

$$\hat{H} = \sum_{X,d} \sum_{n,n'=1}^N \hat{H}_{X,d}^{[n]} \hat{H}_{X,d}^{[n']}. \quad (37)$$

Furthermore, we can substitute the definition of the local matrices in Eq. (13) to obtain:

$$\hat{H} = \sum_{X,I} \sum_{n,n'=1}^N \sum_{t,t'=1}^d \hat{H}_{X,I}(t_n) \hat{H}_{X,I}(t'_{n'}) |tt'\rangle \langle tt'|_{nn'}. \quad (38)$$

The operators  $|tt'\rangle \langle tt'|_{nn'} = |t\rangle \langle t|_n |t'\rangle \langle t'|_{n'}$  act on subsystems  $n$  and  $n'$ . Following the encoding in Eq. (32), we express an operator  $|s\rangle \langle s|$  on a single subsystem with qubit operators as follows:

$$\begin{aligned} |1\rangle \langle 1| &\equiv |00_q\rangle \langle 00_q| = \frac{1}{4} (\hat{I} \otimes \hat{I} + \hat{I} \otimes \hat{Z} + \hat{Z} \otimes \hat{Z} + \hat{Z} \otimes \hat{I}), \\ |2\rangle \langle 2| &\equiv |01_q\rangle \langle 01_q| = \frac{1}{4} (\hat{I} \otimes \hat{I} - \hat{I} \otimes \hat{Z} - \hat{Z} \otimes \hat{Z} + \hat{Z} \otimes \hat{I}), \\ |3\rangle \langle 3| &\equiv |11_q\rangle \langle 11_q| = \frac{1}{4} (\hat{I} \otimes \hat{I} - \hat{I} \otimes \hat{Z} + \hat{Z} \otimes \hat{Z} - \hat{Z} \otimes \hat{I}), \\ |4\rangle \langle 4| &\equiv |10_q\rangle \langle 10_q| = \frac{1}{4} (\hat{I} \otimes \hat{I} + \hat{I} \otimes \hat{Z} - \hat{Z} \otimes \hat{Z} - \hat{Z} \otimes \hat{I}). \end{aligned} \quad (39)$$

Consequently, an operator  $|tt'\rangle \langle tt'|_{nn'}$  contains terms with up to  $d = 4$   $\hat{Z}$ -operators. After this substitution, all terms in the Hamiltonian with the same operators can be collected.

Given that terms of the Hamiltonian  $\hat{H}$  only contain  $\hat{Z}$ -operators, they all commute. The exponential function in the problem unitary  $U_C$  in Eq. (34) can therefore be separated into the exponential function of the individual terms. The number of terms therefore has a direct impact on the depth of the decomposed circuit. Furthermore, the exponential function of a term comprising only identity operators and a number of  $N_Z$  Pauli- $\hat{Z}$  operators can be implemented with  $2N_Z$  CNOT-gates and a single-qubit  $\hat{R}_z$  rotation [7]. It is therefore favorable to only have low-order terms. Due to its construction and the chosen qubit-encoding, the loss function Hamiltonian only contains terms up to fourth order. The expansion also includes a scalar term, only consisting of identity operators, which merely adds a phase to the exponential and can thus be ignored.

We can calculate the number of possible terms by examining the different scenarios: There are  $2N$  ways to place a single  $\hat{Z}$  operator. For second-order terms, there are  $N$  ways to place  $\hat{Z}$  on a pair of qubits corresponding to one ply and  $4N(N-1)$  ways to place two  $\hat{Z}$  operators on different plies. For third-order terms, there are  $2N(N-1)$  ways to place a pair of  $\hat{Z}$  operators on one ply and a single one on another ply. Lastly, there are  $N(N-1)$  possible fourth-order terms resulting from placing two pairs on two different plies. Consequently, the unitary  $U_C$  contains at most  $2N$  terms with one,  $4N^2 - 3N$  terms with two,  $2N(N-1)$  terms with three, and  $N(N-1)$  terms with four  $\hat{Z}$  operators. The quadratic nature of the Hamiltonian results in an  $O(N^2)$  behavior in both the number of terms and the required number of two-qubit gates in the problem unitary  $U_C$ . Given that this circuit is executed  $N_r$  times, the complexity can significantly increase with a larger number of plies  $N$ , potentially exceeding the capabilities of near-future quantum devices. Nevertheless, the quadratic scaling and the lower order of the terms are generally viewed as favorable when compared to the potential complexity of terms in a general Hamiltonian.

We can similarly analyze the penalty operators described in Section 3.2. The nearest-neighbor constraint, detailed in Eq. (21), comprises solely second-order terms. Once the qubit encoding is applied, these terms yield at most four  $\hat{Z}$  operators. Since these operators are already incorporated into the loss function Hamiltonian, the integration of the penalty terms and subsequent simplification of the Hamiltonian do not increase the number of terms or their order. Consequently, the circuit depth remains unchanged upon adding the penalty term. Similarly, the penalty for balanced laminates, as specified in Eq. (27), is quadratic and can be added to the Hamiltonian without increasing the gate count.

The situation changes when considering the penalty operators for the  $k$ -local constraint in Eq. (24). These terms involve operators on  $k$  consecutive plies, and after encoding, the number of  $\hat{Z}$  operators can escalate to  $2k$ , necessitating up to  $4k$  two-qubit gates per term. However, since the operators in a term are restricted to consecutive plies, the total number of possible terms scales as  $O(N)$ , assuming  $k$  remains constant. Thus, if  $k$  is not excessively large, the gate count required to implement this operator might still be manageable on near-term quantum computers.

As previously discussed in Section 3.2.4, any expansion of the penalty operator for the 10%-rule, as delineated in Eq. (29), introduces high-order terms that scale with the number of plies  $N$ . Implementing the exponential function of these high-order terms presents a significant challenge to enforcing the 10%-rule in QAOA using a penalty term.

In our demonstrations in section Section 5, we focus exclusively on implementing the disorientation constraint as a nearest-neighbor constraint.

#### 4.2. DMRG

The hardware-efficient approach, along with the selected optimization procedure, shares many similarities with a specific tensor network method known as the *density matrix renormalization group* (DMRG) algorithm [74–79,107]. A principal challenge in simulating quantum-based calculations on classical computers is the exponentially large state space, making it impractical to represent a state by storing all its components. Tensor networks address this challenge by enabling the representation of high-order tensors through a network of interconnected low-order tensors [124–131]. Although these methods have limitations and are unlikely to efficiently simulate full-scale fault-tolerant quantum computations, they have proven highly effective in studying many-body quantum mechanics and simulating quantum circuits. Remarkably, some quantum-computing experiments that claimed to demonstrate quantum advantage were later classically simulated with tensor network methods [6,132–134] and [135–137].

The DMRG algorithm is a specialized tensor network method designed to identify the ground state of one-dimensional many-body quantum systems. It employs specific tensor network structures – *matrix product states (MPS)* and *matrix product operators (MPO)* – to represent quantum states and operators, respectively. The optimization procedure of the DMRG algorithm mirrors that of our hardware-efficient approach described in Section 4.1.1. Specifically, much like only adjacent qubits are interconnected by CNOT gates in our approach, the MPS configuration connects local tensors across neighboring subsystems. Moreover, the optimization in DMRG involves local optimizations on neighboring subsystem pairs, which is akin to the localized pairwise optimization of adjacent qubits used in our hardware-efficient strategy. In fact, and as we discuss later on, the hardware-efficient circuit can easily be converted into an MPS. A key distinction from the quantum algorithms is that MPS and MPO representations are defined directly on the ply-subsystems  $\mathcal{H}_n$ , eliminating the need for qubit decomposition. Thus, optimizations are conducted on neighboring subsystems rather than qubits.

The main challenge of applying the DMRG algorithm to the SSR problem is finding an MPO representation of the loss function Hamiltonian and the penalty operators. We begin our discussion by developing an MPO representation for the loss function, followed by an examination of the various penalty operators in Section 3.2.

#### 4.2.1. An MPO representation of the loss function Hamiltonian

In this section, we construct an MPO representation of the loss function Hamiltonian  $\hat{H}$  in Eq. (12). As we discuss further below, a summation of matrix product operators can straightforwardly be implemented in MPO form. Thus, we initially concentrate on a single term  $\hat{H}_{X,I}$ , corresponding to the lamination parameter  $v_I^X$ :

$$\hat{H} = \sum_{X,I} \hat{H}_{X,I}, \quad \hat{H}_{X,I} = \left( \sum_{n=1}^N \hat{H}_{X,I}^{[n]} \right)^2. \quad (40)$$

For clarity, we define the matrix elements for states  $|\vec{s}\rangle, |\vec{s}'\rangle \in \mathcal{H}$  of the diagonal operator  $\hat{H}_{X,I}$  as:

$$h_{\vec{s},\vec{s}'} := \langle \vec{s} | \hat{H}_{X,I} | \vec{s}' \rangle = \delta_{\vec{s},\vec{s}'} h_{\vec{s}}, \quad h_{\vec{s}} = \langle \vec{s} | \hat{H} | \vec{s} \rangle, \quad (41)$$

implying:

$$\hat{H}_{X,I} = \sum_{\vec{i},\vec{i}' \in \mathcal{S}} h_{\vec{i},\vec{i}'} |\vec{i}\rangle \langle \vec{i}'| = \sum_{\vec{i} \in \mathcal{S}} h_{\vec{i}} |\vec{i}\rangle \langle \vec{i}|. \quad (42)$$

Similarly, we denote the eigenvalues of the local operators  $\hat{H}_{X,I}^{[n]}$  as:

$$h_{s_n}^{[n]} = \langle s_n | \hat{H}_{X,I}^{[n]} | s_n \rangle = H_{X,I}^{[n]}(s_n), \quad (43)$$

where, according to the definition of the loss function, we have:

$$h_{\vec{s}} = \left( \sum_{n=1}^N h_{s_n}^{[n]} \right)^2. \quad (44)$$

To construct the MPO representation, we aim to define a tensor  $W^{[1]}$  sized  $d \times d \times b_1$ , a tensor  $W^{[N]}$  sized  $d \times d \times b_{N-1}$ , and tensors  $W^{[n]}$  for  $1 < n < N$  sized  $d \times d \times b_{n-1} \times b_n$ , such that:

$$h_{\vec{s},\vec{s}'} = W_{s_1 s'_1}^{[1]} W_{s_2 s'_2}^{[2]} \dots W_{s_{N-1} s'_{N-1}}^{[N-1]} W_{s_N s'_N}^{[N]}, \quad (45)$$

where the bond dimensions  $b_1, \dots, b_{N-1}$  are parameters yet to be determined based on the specific MPO representation required. Note, that for specific  $\vec{s}, \vec{s}' \in \mathcal{S}$ , the matrices  $W_{s_n s'_n}^{[n]}$  are multiplied with matrix multiplication, with the ones for  $n = 1$  and  $n = N$  taken as a row and column vector respectively. Given that the Hamiltonian is diagonal, the tensors  $W^{[n]}$  are also assumed diagonal in the indices  $s_n$  and  $s'_n$ :

$$W_{s_n s'_n}^{[n]} = \delta_{s_n s'_n} W_{s_n}^{[n]}, \quad W_{s_n}^{[n]} := W_{s_n s_n}^{[n]}. \quad (46)$$

Thus, to reflect Eq. (44), matrices  $W_{s_n}^{[n]}$  must be defined such that:

$$\left( \sum_{n=1}^N h_{s_n}^{[n]} \right)^2 = W_{s_1}^{[1]} W_{s_2}^{[2]} \dots W_{s_N}^{[N]}, \quad (47)$$

and the entire MPO representation for the matrix elements  $h_{\vec{s},\vec{s}'}$  can be realized by multiplying a Kronecker delta  $\delta_{s_n, s'_n}$  with the resulting matrices  $W_{s_n}^{[n]}$ .

Our approach decomposes  $h_{\vec{s}}$  using a matrix whose repeated multiplication yields  $h_{\vec{s}}$  as the top-right element. Specifically, we define:

$$M(x) = \begin{bmatrix} 1 & \sqrt{2}x & x^2 \\ 0 & 1 & \sqrt{2}x \\ 0 & 0 & 1 \end{bmatrix}, \quad (48)$$

which follows the multiplication rule:

$$M(x)M(y) = M(x + y). \quad (49)$$

By induction, multiplying  $N$  of these matrices, each associated with the eigenvalues  $h_{s_n}^{[n]}$ , results in:

$$\prod_{n=1}^N M(h_{s_n}^{[n]}) = M\left(\sum_n h_{s_n}^{[n]}\right) = \begin{bmatrix} 1 & \sqrt{2}\left(\sum_n h_{s_n}^{[n]}\right) & \left(\sum_n h_{s_n}^{[n]}\right)^2 \\ 0 & 1 & \sqrt{2}\left(\sum_n h_{s_n}^{[n]}\right) \\ 0 & 0 & 1 \end{bmatrix}. \quad (50)$$

An MPO representation of  $\hat{H}_{X,l}$  as in Eq. (45) can thus be obtained by breaking it down into these matrices:

$$W_{s_n s'_n}^{[n]} = \delta_{s_n s'_n} M(h_{s_n}^{[n]}) \quad (51)$$

for  $1 < n < N$ , with the first row and third column of the first and last matrix selected to yield  $\left(\sum_n h_{s_n}^{[n]}\right)^2$ :

$$\begin{aligned} W_{s_1 s'_1}^{[1]} &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \delta_{s_1 s'_1} M(h_{s_1}^{[1]}), \\ W_{s_N s'_N}^{[N]} &= \delta_{s_N s'_N} M(h_{s_N}^{[N]}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \end{aligned} \quad (52)$$

The total loss function Hamiltonian in Eq. (12) compiles  $\hat{H}_{X,l}$  for all lamination parameters  $X = A, B, D$  and  $l = 1, 2, 3, 4$ , excluding  $X = B$  for symmetric laminates. This sum of matrix product operators (MPOs) can theoretically be realized by stacking the matrices  $W_{s_n s'_n}^{[n]}$  for each  $n$  into block-diagonal matrices. However, thanks to the linearity of the expectation value  $\langle \psi | \hat{H} | \psi \rangle$ , the DMRG method can efficiently iterate over each term individually by calculating an environment tensor for each term and subsequently summing these tensors element-wise to construct the overall environment tensor.

#### 4.2.2. An MPO representation for nearest-neighbor constraints

As discussed in Section 3.2, we can implement nearest-neighbor constraints as penalty functions, which can be expressed as operators and added to the Hamiltonian. The sum of the operators in Eq. (21) effectively counts the constraint violations. There, we defined the added penalty for one constraint violation to be

$$\eta(s_n, s_{n+1}) = \begin{cases} \gamma, & \text{if } (s_n, s_{n+1}) \text{ violates constraint,} \\ 0, & \text{else,} \end{cases} \quad (53)$$

where  $\gamma > 0$  is the penalty for a single constraint violation. Similar to other terms in the Hamiltonian, the penalty function needs to be represented as an MPO. As before, we construct a matrix that calculates the desired result, which is the count of constraint violations, in the top-right corner. For this purpose, we represent the function  $\eta(s_n, s_{n+1})$  as the dot product of two vectors  $\vec{p}(s_n)$  and  $\vec{q}(s_{n+1})$ , both of dimension  $d$ :

$$\eta(s_n, s_{n+1}) = \vec{p}(s_n) \cdot \vec{q}(s_{n+1}) = \sum_{k=1}^d p_k(s_n) q_k(s_{n+1}). \quad (54)$$

We achieve this by using  $\vec{p}(s_n)$  as an indicator for the state  $s_n$ , and  $\vec{q}(s_{n+1})$  to indicate which neighboring ply of state  $s_n$  leads to a constraint violation:

$$p_k(s_n) = \delta_{k, s_n}, \quad q_k(s_{n+1}) = \eta(k, s_{n+1}). \quad (55)$$

If we define a matrix  $M(s)$  based on  $\vec{p}$  and  $\vec{q}$  as:

$$M(s) = \begin{bmatrix} 1 & p_1(s) & \cdots & p_d(s) & 0 \\ 0 & 0 & \cdots & 0 & q_1(s) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & q_d(s) \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad (56)$$

it can be shown by induction that the product of these matrices yields the count of constraint violations in the top-right corner of the resulting matrix:

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \left( \prod_{n=1}^N M(s_n) \right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \sum_{n=1}^{N-1} \eta(s_n, s_{n+1}). \quad (57)$$

This matrix form is similar to the treatment of operators in [138,139]. To retrieve the penalty in MPO form, we can add Kronecker deltas analogously to before.

In the particular case of a disorientation constraint of  $45^\circ$ , the constraint is violated when neighboring plies are  $\theta_1$  and  $\theta_3$ , or  $\theta_2$  and  $\theta_4$ . In this case, we can define the vectors  $\vec{p}(s)$  and  $\vec{q}(s)$  for the MPO representation of the penalty as:

$$p_k(s) = \delta_{k,s}, \quad q_k(s) = \gamma(\delta_{k,s+2} + \delta_{k,s-2}), \quad (58)$$

where  $\gamma$  represents the penalty added for a single constraint violation.

#### 4.2.3. An MPO representation for the contiguity constraint

For the contiguity constraint, we extend the approach for nearest-neighbor constraints to  $k$ -local constraints. This extension involves modifying the penalty function, which is now defined for a sequence of  $k$  subsystems. Specifically, the penalty function  $\eta(s_n, s_{n+1}, \dots, s_{n+k-1})$  assigns a penalty of  $\gamma$  if the sequence  $(s_n, s_{n+1}, \dots, s_{n+k-1})$  violates the constraint, and 0 otherwise. To represent this penalty function, we seek  $k$  vectors  $\vec{p}^{(1)}(s), \dots, \vec{p}^{(k)}(s) \in \mathbb{C}^{\tilde{d}}$  for each ply angle  $s \in \{1, \dots, d\}$  such that:

$$\eta(s_n, s_{n+1}, \dots, s_{n+k-1}) = \sum_{j=1}^{\tilde{d}} p_j^{(1)}(s_n) p_j^{(2)}(s_{n+1}) \cdots p_j^{(k)}(s_{n+k-1}). \quad (59)$$

Here,  $d$  is the number of allowed ply angles, and  $\tilde{d}$ , a positive integer, denotes the size of these vectors. For the specific case of the contiguity constraint, which penalizes  $k$  consecutive plies with the same ply-angle, we find an effective representation with  $\tilde{d} = d$ . The vectors are defined as:

$$p_j^{(1)}(s) = \cdots = p_j^{(k-1)}(s) = \delta_{j,s}, \quad p_j^{(k)}(s) = \gamma \delta_{j,s}, \quad (60)$$

where we chose  $\gamma$  to included  $p_j^{(k)}(s)$ . Other possibilities include multiplying  $\gamma$  to another vector or distribute it by multiplying  $\gamma^{\frac{1}{k}}$  to each of the vector.

We extend the definition of the matrix  $M(s)$ , as outlined in Eq. (56), to adapt it for the  $k$ -local scenario. This adaptation involves constructing a square matrix of dimension  $\tilde{d}(k-1) + 2$ . The vectors  $\vec{p}^{(1)}(s)$  and  $\vec{p}^{(k)}(s)$  are incorporated in the first row and last column respectively, similar to Eq. (56). However, to cater to the  $k$ -local aspect, we integrate additional columns and rows. These new elements form a diagonal matrix  $\text{diag}(\vec{p}^{(2)}(s), \dots, \vec{p}^{(k-1)}(s))$  of dimension  $\tilde{d}(k-2)$ , with the elements of vectors  $\vec{p}^{(2)}(s)$  to  $\vec{p}^{(k-1)}(s)$  listed sequentially along the diagonal. The resulting matrix  $M(s)$  for the  $k$ -local case can thus be represented in block-matrix form as:

$$M(s) = \begin{bmatrix} 1 & (\vec{p}^{(1)}(s))^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \text{diag}(\vec{p}^{(2)}(s), \dots, \vec{p}^{(k-1)}(s)) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \vec{p}^{(k)}(s) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{bmatrix}. \quad (61)$$

In this matrix,  $(\vec{p}^{(1)}(s))^T$  is a row vector and  $\vec{p}^{(k)}(s)$  is a column vector, both of size  $\tilde{d}$ . The matrices  $\mathbf{0}$  contain only zeros and are sized appropriately. Single-element matrices containing 0 or 1 are denoted in regular font weight. As in Section 4.2.2, a product of matrices  $M(s_1) \cdots M(s_N)$  counts the penalties for constraint violations in the top right corner:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} M(s_1) \cdots M(s_N) \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \sum_{n=1}^{N-k+1} \eta(s_n, \dots, s_{n+k-1}), \quad (62)$$

where the vectors surrounding the product select for the first row and last column. Following, the element-wise (or Hadamard) product of two vectors,  $\vec{a}$  and  $\vec{b}$ , is denoted by  $\vec{a} \odot \vec{b}$ , and is defined as:

$$(\vec{a} \odot \vec{b})_j = a_j b_j. \quad (63)$$

To illustrate this, consider the action of a single matrix  $M(s)$  on a vector. For an arbitrary complex number  $a \in \mathbb{C}$  and vectors  $\vec{b}^{(j)} \in \mathbb{C}^{\tilde{d}}$  for  $j = 1, \dots, k-1$ , the matrix operation is given by:

$$M(s) \begin{bmatrix} a \\ \vec{b}^{(1)} \\ \vec{b}^{(2)} \\ \vdots \\ \vec{b}^{(k-2)} \\ \vec{b}^{(k-1)} \\ 1 \end{bmatrix} = \begin{bmatrix} a + \vec{p}^{(1)}(s) \odot \vec{b}^{(1)} \\ \vec{p}^{(2)}(s) \odot \vec{b}^{(2)} \\ \vec{p}^{(3)}(s) \odot \vec{b}^{(3)} \\ \vdots \\ \vec{p}^{(k-1)}(s) \odot \vec{b}^{(k-1)} \\ \vec{p}^{(k)}(s) \\ 1 \end{bmatrix}. \quad (64)$$

In this operation, each vector  $\vec{b}^{(j)}$  is elevated one position and then multiplied element-wise with the corresponding vector  $\vec{p}^{(j)}(s)$ . The top vector  $\vec{b}^{(1)}$  undergoes a dot product with  $\vec{p}^{(1)}(s)$ , which is added to the scalar  $a$ . The vector  $\vec{p}^{(k)}(s)$  is inserted into the bottom



slot that becomes vacant. Consequently, when the matrices in Eq. (62) are multiplied from right to left, we observe the sequential accumulation and elevation of the vectors  $\vec{p}^{(k)}$ . This is depicted in the following process:

$$\begin{aligned}
 & \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \xrightarrow{M(s_N)} \begin{bmatrix} 0 \\ \mathbf{0}_{\vec{d}(k-2)} \\ \vec{p}^{(k)}(s_N) \\ 1 \end{bmatrix} \xrightarrow{M(s_{N-1})} \begin{bmatrix} 0 \\ \mathbf{0}_{\vec{d}(k-3)} \\ \vec{p}^{(k-1)}(s_{N-1}) \odot \vec{p}^{(k)}(s_N) \\ \vec{p}^{(k)}(s_{N-1}) \\ 1 \end{bmatrix} \xrightarrow{M(s_{N-2})} \dots \\
 & \xrightarrow{M(s_{N-k+2})} \begin{bmatrix} 0 \\ \vec{p}^{(2)}(s_{N-k+2}) \odot \dots \odot \vec{p}^{(k)}(s_N) \\ \vdots \\ \vec{p}^{(k-1)}(s_{N-k+2}) \odot \vec{p}^{(k)}(s_{N-k+3}) \\ \vec{p}^{(k)}(s_{N-k+2}) \\ 1 \end{bmatrix} \xrightarrow{M(s_{N-k+1})} \begin{bmatrix} \eta(s_{N-k+1}, \dots, s_N) \\ \vec{p}^{(2)}(s_{N-k+1}) \odot \dots \odot \vec{p}^{(k)}(s_{N-1}) \\ \vdots \\ \vec{p}^{(k-1)}(s_{N-k+1}) \odot \vec{p}^{(k)}(s_{N-k+2}) \\ \vec{p}^{(k)}(s_{N-k+1}) \\ 1 \end{bmatrix}.
 \end{aligned} \tag{65}$$

In this sequence, the suffixes of the matrices  $\mathbf{0}$  indicate their size. Upon integrating the matrix  $M(s_{N-k+1})$ , the accurate penalty for the last  $k$  subsystems, from  $N - k + 1$  to  $N$ , is added to the first row. The elements below are positioned to include the correct penalty for the subsystems from  $N - k$  to  $N - 1$  when the subsequent matrix  $M(s_{N-k})$  is integrated into the product. By continuing this process for all matrices and then selecting the first row, the correct penalty for the entire stacking sequence  $\vec{s}$  is computed, validating Eq. (62).

The matrices resulting from this formulation, each having a dimension of  $\vec{d}(k-1) + 2$ , are notably larger compared to the other MPOs we have previously discussed. However, as this only effects tensor contractions and not the much more costly eigenvalue search in the DMRG algorithm, the impact on the runtime remains manageable. Nevertheless, it remains to be seen how DMRG will navigate these intricate constraints and what implications they may have on the solution's accuracy.

#### 4.2.4. An MPO representation for balanced laminates

As outlined in Section 3.2, the constraint for balanced laminates, as well as generalized constraints where two states,  $s$  and  $t$ , appear an equal number of times in a stacking sequence, can be expressed using a penalty operator:

$$\hat{H}_{\text{balanced}} = \gamma \left( \sum_{n=1}^N \hat{H}^{[n]} \right)^2, \tag{66}$$

where the local operators  $\hat{H}^{[n]}$  are defined for each subsystem  $n$  as:

$$\hat{H}^{[n]} = |s\rangle\langle s|_n - |t\rangle\langle t|_n = \sum_{s_n=1}^d \left( \delta_{s_n,s} - \delta_{s_n,t} \right) |s_n\rangle\langle s_n|_n. \tag{67}$$

Given that the penalty operator bears similarity to the Hamiltonian form for the lamination parameters, as seen in Eq. (40), the same methodology discussed in Section 4.2.1 can be applied. For this, we set the eigenvalues  $h_{s_n}^{[n]}$  of the system to:

$$h_{s_n}^{[n]} = \delta_{s_n,s} - \delta_{s_n,t}, \tag{68}$$

yielding a representation of the penalty operator  $\hat{H}_{\text{penalty}}^{(\text{balanced})}$  in MPO form. The factor  $\gamma$  can be included by multiplying it to any one of the local tensors.

#### 4.2.5. An MPO representation for the 10%-rule

As detailed in Section 3.2, constraints such as the 10%-rule require a minimum of  $N_s$  plies to be in a specific state  $s$ . For this constraint, we implement a linear penalty function:

$$H_{10\%-rule}(\vec{s}) = \begin{cases} 0, & \text{if } \sum_{n=1}^N \delta_{s_n,s} \geq N_s, \\ \gamma \left( N_s - \sum_{n=1}^N \delta_{s_n,s} \right), & \text{else,} \end{cases} \tag{69}$$

where the parameter  $\gamma > 0$  controls the magnitude of the added penalty. While an expansion of the according operator into single-site operators is more intricate, as discussed in Section 3.2, a straightforward representation as an MPO can be found via the decomposition:

$$H_{10\%-rule}(\vec{s}) = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} M(s_1) \dots M(s_N) \begin{bmatrix} \gamma N_s \\ -\gamma \\ \vdots \\ -\gamma \end{bmatrix}, \tag{70}$$

where each  $M(s_n)$  is a square matrix of dimension  $N_s + 1$ . For  $s_n = s$ ,  $M(s)$  is defined as:

$$M(s) = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (71)$$

and for  $s_n \neq s$ , it is equivalent to the identity matrix:

$$M(s_n) = \mathbf{I}_{N_s+1} \quad \text{for } s_n \neq s. \quad (72)$$

Consequently, multiplying  $M(s_n)$  with a vector  $\vec{a} \in \mathbb{C}^{N_s+1}$  leaves  $\vec{a}$  unchanged if  $s_n \neq s$ . However, for  $s_n = s$ , it shifts all elements below the first row up by one position, combines the first two elements in the first row, and appends a zero at the end:

$$M(s) \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N_s+1} \end{bmatrix} = \begin{bmatrix} a_1 + a_2 \\ a_3 \\ \vdots \\ a_{N_s+1} \\ 0 \end{bmatrix}. \quad (73)$$

Thus, applying this matrix to the column vector in Eq. (70) cumulatively adds  $-\gamma$  for each occurrence of  $s_n = s$ , effectively subtracting from the penalty. If the constraint is satisfied for  $\vec{s}$ , the penalty becomes zero after  $N_s$  such additions. Meanwhile, the remaining elements of the vector are now also 0, such that multiplying any further matrices leave the vector unchanged. Conversely, if the constraint is not met, the penalty remains non-zero, reflecting the penalty function  $H_{10\%-rule}(\vec{s})$ . This justifies the decomposition in Eq. (70), which can be used to construct the MPO similarly to previously discussed cases.

Similar to the contiguity constraint, these matrices are significantly larger than those of the other encountered MPOs. As discussed earlier, this should have a comparatively minor impact on the runtime, while the influence on the accuracy remains to be explored.

#### 4.3. The connection between the hardware-efficient approach and DMRG

In this section, we highlight the similarities between the hardware-efficient approach and the DMRG algorithm, and contrasting these with the properties of QAOA. From an implementation perspective, both QAOA and the hardware-efficient approach involve encoding states in the Hilbert space on qubits, performing state-vector simulations with a designated number of repetitions  $N_r$  of a parameterized quantum circuit, and extracting the Hamiltonian's expectation value. A classical optimizer is then employed to search for parameters that minimize this value. However, the specific state preparation circuits and optimization procedures lead to different landscapes for the Hamiltonian expectation value. Specifically, QAOA utilizes operators  $U_C$  and  $U_M$  that act simultaneously on all qubits with a single parameter per operator, allowing for the simultaneous optimization of a relatively small number of parameters. In contrast, the hardware-efficient approach optimizes only the parameters affecting neighboring qubits, much like how DMRG optimizes subsystems for neighboring plies.

Furthermore, the connection between the hardware-efficient approach and DMRG extends deeper [140–143]. Viewing the gates in a quantum circuit as matrices naturally leads to a tensor network representation of the circuit. The initial qubit states  $|0\rangle$  can be included in vector form to form a tensor network representing the states prepared by the circuit. In the hardware-efficient circuit, where two-qubit gates act only on neighboring qubits, performing singular value decompositions (SVD) on these gates and contracting all tensors associated with individual qubits effectively yields an MPS representation of the prepared states. This results in open indices for individual qubits, but by grouping the qubits associated with one ply, we can also achieve an MPS representation similar to those used in DMRG. This conceptualization allows us to view the hardware-efficient approach as a qubit implementation of a specifically parameterized MPS.

The bond dimension of this MPS can be inferred from the SVD of the CNOT gate. Typically, the CNOT is represented as a matrix with non-zero elements  $C_{q_1 q_2, q'_1 q'_2} := \langle q_1 q_2 | \text{CNOT} | q'_1 q'_2 \rangle$ :

$$C_{00,00} = C_{01,01} = C_{10,11} = C_{11,10} = 1,$$

which are applied to the current two-qubit state and return the state after the application of the gate. However, performing an SVD on this matrix would split the matrix along the time direction, and not separate the qubits. We therefore reorder the indices  $\tilde{C}_{q_1 q'_1, q_2 q'_2} = C_{q_1 q_2, q'_1 q'_2}$  to obtain non-zero elements:

$$\tilde{C}_{00,00} = \tilde{C}_{00,11} = \tilde{C}_{11,01} = \tilde{C}_{11,10} = 1.$$

Notably, the CNOT gate does not alter the control qubit, leading to a rank-2 matrix representation. Consequently, the SVD yields only 2 non-zero singular values, implying that each CNOT gate between two qubits doubles the bond dimension of the corresponding MPS. As each repetition in the hardware-efficient circuit introduces an additional CNOT gate between two qubits, the bond dimension of the MPS grows exponentially with  $2^{N_r}$ . With this result, we can compare this hardware-efficient circuit directly to an MPS of a specific bond-dimension that is utilized in DMRG. Conversely, QAOA uses numerous CNOT gates per repetition on non-neighboring qubits, stemming from the many terms in the Hamiltonian. Consequently, the tensor-network representation of the circuit has a

significantly more complicated form than a simple MPS. We can therefore expect the hardware-efficient approach and DMRG to have comparable behaviors, while QAOA will likely perform very differently.

However, there are key differences between the hardware-efficient approach and DMRG. The hardware-efficient approach optimizes neighboring qubits, whereas DMRG directly operates on the ply subsystems  $\mathcal{H}_n$ . In principle, the optimization in the hardware-efficient approach could also be performed by simultaneously optimizing the 4 qubits corresponding to 2 plies, which introduces the challenge of handling more parameters concurrently. Moreover, while the hardware-efficient approach utilizes a parameterized MPS with a classical optimizer, DMRG conducts optimization through eigendecomposition of an environment tensor followed by SVD to optimize local tensors. This method also impacts the control of bond dimensions, since this procedure naturally increases the bond-dimension and offers tools for its controlled reduction, unlike in the hardware-efficient approach where the bond dimension remains fixed and a method to reduce the number of repetitions  $N_r$  is not readily available. This analysis underscores how the exploration of tensor-network methods like DMRG can provide valuable insights into specific quantum algorithms such as our hardware-efficient approach.

## 5. Numerical demonstration

We performed numerical simulations of the algorithms outlined in Section 4 to demonstrate our quantum formulation of stacking sequence retrieval as presented in Section 3. Specifically, we conducted state-vector simulations for the two quantum algorithms discussed in Section 4.1: QAOA and the previously described hardware-efficient approach. As DMRG is a classical algorithm, it did not require simulation on quantum computing hardware and was implemented directly.

Given the limitations of simulating quantum computing on classical hardware, particularly with a large number of qubits, we limited our simulations to a symmetric laminate comprising a small number of  $N = 6$  plies in one half of the stack, corresponding to a total of 12 simulated qubits. However, the MPO and MPS representations used in DMRG allow for a more efficient management of the state space compared to the direct simulation of subsystems, enabling DMRG to efficiently handle a much larger number of subsystems. Consequently, we also conducted DMRG simulations for a substantially larger configuration of  $N = 200$  plies.

All simulations were performed both with and without the inclusion of a disorientation constraint of  $45^\circ$ .

### 5.1. Setup

The numerical calculations were carried out on a laptop equipped with an Intel Core i7-1185G7 processor (4 cores, 3.00 GHz), 16 GB RAM, running Windows 10 Professional. The complete code for these experiments is available at [144]. The algorithm was executed on the CPU without using parallelization techniques like multi-threading. All files generated from our experiments are available in HDF5 format at [145].

The quantum simulations were performed in *Python* in version 3.11 using the *Qiskit* library in version 1.0.2 [146]. This library facilitated the construction and simplification of the loss function and penalty Hamiltonian in expanded Pauli operator form, as detailed in Section 4.1.2. Additionally, the state preparation circuits for QAOA and the hardware-efficient approach were generated as quantum circuits, which *Qiskit* converts into an instance of a state vector class. This class provides a function for calculating the expectation value of operators such as the Hamiltonian. Optimization was conducted using the *Scipy* library in version 1.12.0 [147], employing the BFGS method as the optimizer.

The numerical implementation and trials were executed in the programming language *Julia*, version 1.8.4. The MPS and MPO were implemented using the *ITensors.jl* library, version 0.3.22 [148,149], which also includes an implementation of the DMRG algorithm. As this implementation supports only alternating sweeping directions, we modified the algorithm to allow for custom sweeping sequences. Additional functionality for recording data during the optimization was also integrated. Aside from the support for custom sweeping directions, the optimization process remains consistent with the original implementation provided by *ITensors*.

For all numerical trials, we selected the four commonly used angles  $0^\circ$ ,  $+45^\circ$ ,  $90^\circ$ ,  $-45^\circ$ , as discussed in Section 4. In the quantum simulations, these angles were encoded using binary encoding as described in section Section 4.1. We applied all three algorithms to a symmetric laminate with  $N = 6$  plies in one half of the stack and optionally included a disorientation constraint. For this constraint, we applied a penalty of  $\gamma = 0.25$  per constraint violation (sections Section 3.2.1 and Section 4.2.2). Additionally, we performed DMRG on a larger scale with  $N = 200$  plies in half the stack, assigning a penalty of  $\gamma = 0.005$ . All trials were conducted with and without the disorientation constraint.

To select a well-justified set of target lamination parameters, we considered two key factors. Firstly, we chose target parameters that correspond to exact solutions of feasible stacking sequences. This approach ensures that the loss function serves as an effective quality measure for any obtained solution, since an optimal solution with a minimal loss function value of 0 always exists. Suitable target parameters can be obtained from generated stacking sequences that comply with the constraints, such that these sequences also act as known optimal solutions for the target parameters.

Secondly, it is crucial that the target lamination parameters represent a diverse cross-section of potential lamination parameters. For  $N = 6$  layers, we achieved this by considering symmetry transformations of the constraint. As we discuss further in Appendix A.2, we anticipate that the algorithms will exhibit similar behavior in cases that transform into each other under transformations that map valid states to valid states and constraint-violating states to constraint-violating states. For the disorientation constraint, these symmetry transformations are generated by rotations that shift all states by the same number and one reflection that swaps 0 and 2, or equivalently 1 and 3. To explore a broad range of potential behaviors, we generated 15 stacking sequences that do not transform

into each other under these symmetry transformations, resulting in a set of 15 distinct target lamination parameters for our trials with  $N = 6$  plies.

For  $N = 200$ , the total number of valid stacking sequences is vastly greater than for  $N = 6$  plies, necessitating a different approach. Common methods for generating random stacking sequences often result in density variations within the corresponding point cloud in the 8-dimensional lamination parameter space, typically due to the specifics of the sequence generation method. To mitigate potential biases from our random generation method, we compiled a large dataset of 500,000 stacking sequences that comply with the disorientation constraint, subsequently converting these into an equivalent number of points in lamination parameter space. We then applied a kernel density estimator to evaluate the point density at each location [150–152]. Finally, we used the inverses of these densities as weights to randomly select points in a manner that more closely approximates a uniform distribution across the lamination parameter space. Employing this method, we sampled a set of 50 target lamination parameters for all our trials with  $N = 200$  plies.

For the QAOA trials, we conducted simulations for  $N_r = 1, 2, 3, 4, 6, 8$  repetitions of the problem and mixer unitaries in the state preparation circuit. For each number of repetitions  $N_r$ , we generated 6 random sets of initial parameters,  $\beta$  and  $\gamma$ , which were used across all 15 target lamination parameters.

In the hardware-efficient approach, we performed optimizations for  $N_r = 0, 1, 2, 3, 4, 5, 6$  repetitions in the state preparation circuit. For each repetition count, we generated 5 random sets of initial parameters. Additionally, for each combination of repetition number, initial parameters, and target, we conducted the optimization twice: once sweeping outward (from  $n = 1$  to  $n = N$ ) and once sweeping inward (from  $n = N$  to  $n = 1$ ). Multiple sweeps were performed consecutively until convergence was achieved.

For DMRG, we conducted optimizations for maximum bond dimensions of 2, 4, 8, 16, and 32. We performed optimizations using alternating sweeps, as is customary in DMRG, as well as only inward and only outward sweeps. In the final stages of optimization, we applied a technique to enforce a basis state corresponding to a single stacking sequence, which involved progressively halving the maximum bond dimension during the last few sweeps, ultimately achieving a bond dimension of 1 in the penultimate sweep. In the final sweep, we applied a truncation criterion that discarded singular values below 0.5 during the dimension reduction, effectively enforcing the representation of a basis state.

For  $N = 6$ , we based the number of sweeps for DMRG on the quick convergence observed in the hardware-efficient approach, as discussed in the following section Section 5.2. Ensuring sufficient sweeps to both grow and reduce the bond dimension as described above, we executed DMRG with a total of 10 sweeps. For  $N = 200$ , to better observe the optimization progression over a more extended series of sweeps, we opted for 69 sweeps, yielding a total of 70 data points per trial when including the initial state.

For each target lamination parameter, we generated 5 initial MPS with a bond dimension of 2. These initial MPS were consistently used across all trials, irrespective of bond dimension and sweep sequence. Given that the initial bond dimension was lower than most of the chosen maximum bond dimensions, the bond dimension of the MPS increased quickly during the initial sweeps of optimization until it reached the predetermined maximum. Therefore, the majority of the sweeps were conducted at this maximum bond dimension.

## 5.2. Results

The general concept of QAOA is to exploit quantum interference to suppress the amplitude of states associated with high losses and enhance the probability of measuring states with lower losses. Accordingly, our trials with QAOA did not yield a specific basis state corresponding to a particular stacking sequence, but resulted in a superposition of states encompassing a range of losses. For each set of resulting QAOA parameters  $\tilde{\beta}$  and  $\tilde{\gamma}$ , we calculated the loss function expectation value of the state and assessed the probability of obtaining the exact solution from the corresponding component of the state. The results are presented in Fig. 3, where we display the Hamiltonian expectation values and the probabilities of obtaining the exact state separately, comparing results with and without the inclusion of the penalty operator. In each plot, we display the overall average for each number of repetitions  $N_r$ , as well as individual traces for each of the 15 targets, averaging only over the 6 runs with different initial parameters. The figure also includes the stacking sequences corresponding to the target lamination parameters. Additionally, we calculated the expectation value of the penalty operator for the disorientation constraint, setting the penalty factor to  $\gamma = 1$ , thus the recorded expectation values represent the average number of constraint violations in the stack. For  $N_r = 1$  repetition in the QAOA circuit, we found an average expectation value of 0.309 constraint violations without the penalty operator, and 0.261 with it. As the number of repetitions  $N_r$  increased, these values decreased, with 0.184 and 0.085 recorded for  $N_r = 8$  repetitions, respectively, without and with the penalty operator.

For the hardware-efficient approach, the behavior observed was markedly different from that of QAOA. Specifically, the hardware-efficient approach effectively produced a basis state after the initial sweep, with subsequent sweeps primarily filtering out remnants of other states. The results are depicted in Fig. 4a, where we show the average RMSE to the target parameters for the final stacking sequences and the percentage that coincided with the optimal state. Additionally, we present results for sweeping inward and sweeping outward in separate plots. No clear trend emerged for different target lamination parameters; thus, we only show the average of all results for a specific number of repetitions  $N_r$ , with and without the constraint included in the same plot. We also tracked the number of resulting stacking sequences that violated the disorientation constraint. Without the penalty, inward sweeps at  $N_r = 0$  repetitions resulted in 65% constraint-violating stacks, while subsequent repetitions yielded between 24% and 39% constraint-violating stacks without a discernible trend with increasing repetitions. Similarly, outward sweeps resulted in 40% constraint violations at  $N_r = 0$  and values between 24% and 39% for other numbers of repetitions, also without showing a clear correlation to the inward sweep values. Including the penalty, we observed 29% and 27% constraint violations for inward and

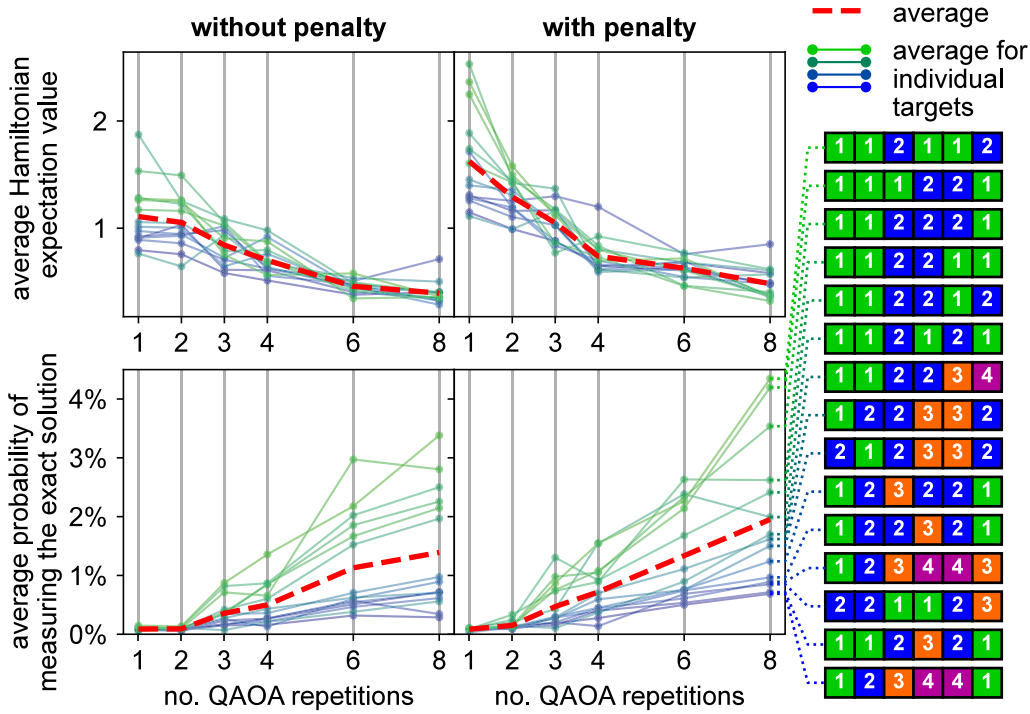


Fig. 3. Results from the simulations of QAOA. The top row shows the average Hamiltonian expectation value while the bottom row shows the average probability of obtaining the exact solution when measuring the final state. Shown are the average of the 6 trials for each individual set of target parameters (blue and green solid lines) and the average over all 15 sets of target parameters (dashed red lines). On the right, the stacking sequences corresponding to the exact solutions are shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

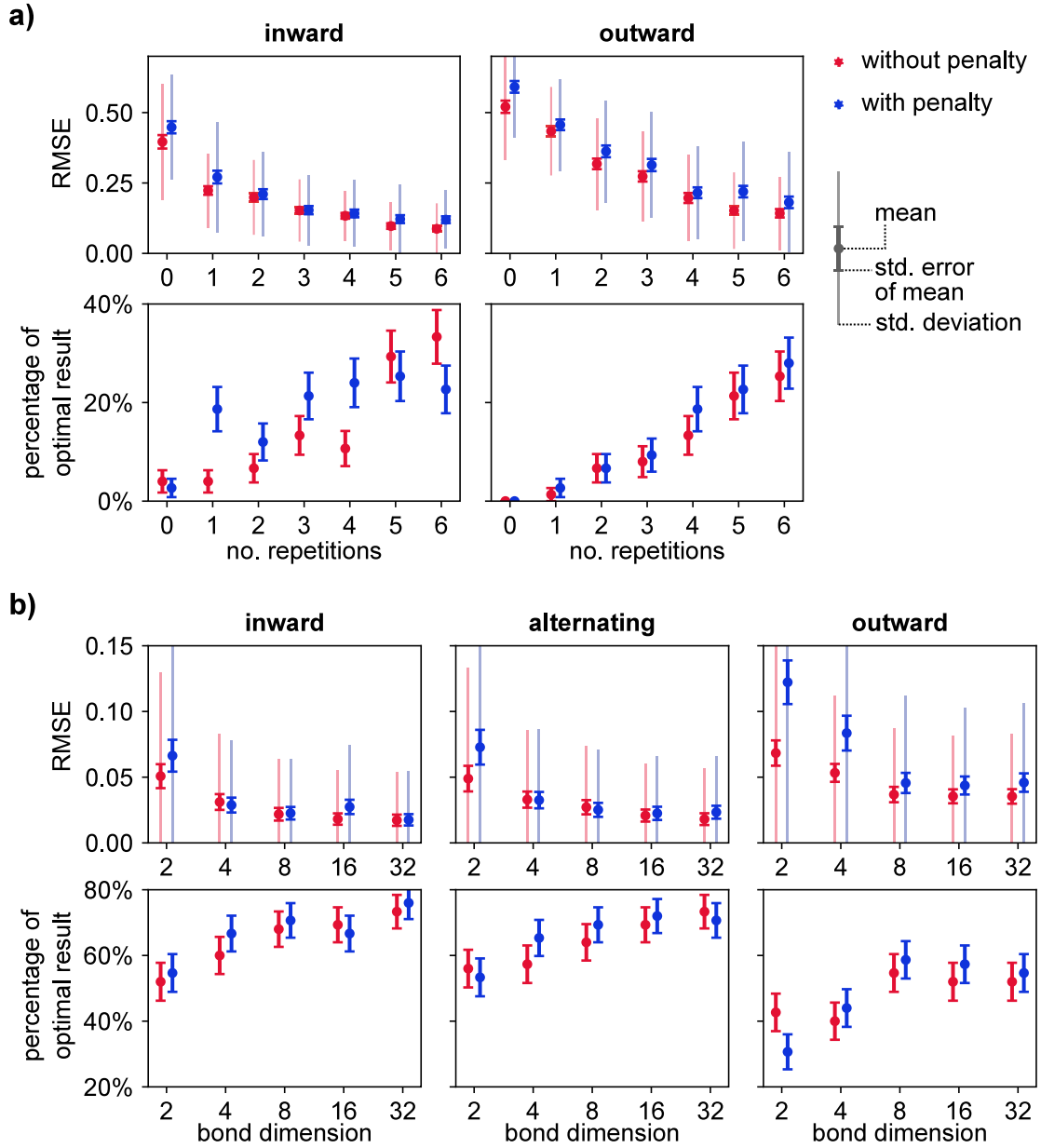
outward sweeps respectively at  $N_r = 0$ . However, with an increase in repetitions  $N_r$ , the percentage of constraint-violating stacks rapidly decreased, dropping below 1.5% at  $N_r = 2$  and  $N_r = 3$  for inward and outward sweeps respectively.

The reduction of the maximum bond dimension for the final sweeps in our implementation of DMRG is designed to enforce a basis state as a result. For  $N = 6$ , we display the results for DMRG in Fig. 4b in a similar fashion as for the hardware efficient approach. The main difference is, that we now have three different sweeping configurations: inward, outward and alternating sweeps. As before, we also counted the states containing constraint violations. Without the penalty, inward and alternating sweeps yielded 19% and 17% constraint violating stacks respectively for bond-dimension 2, which reduced to 12% in both cases for bond-dimension 32. Outward sweeps showed higher values, with 28% for bond-dimension 2 and 24% for bond-dimension 32. When including the penalty, we only observed a single constraint-violating stack for alternating sweeps and a single one for outward sweeps, both at bond-dimension 2, which equates to 1.3%.

For DMRG trials with  $N = 200$  plies, we offer a more comprehensive overview of performance. Firstly, we confirmed that all resulting stacking sequences were free of constraint violations when the disorientation constraint was included. Without the penalty, every combination of maximum bond dimension and sweeping direction yielded at least 96% constraint-violating stacks, indicating that nearly all resulting stacks violated the disorientation constraint.

Additionally, we documented the expectation value of the loss function Hamiltonian after each sweep in all trials. Fig. 5a shows the average across all target lamination parameters as a function of the number of sweeps. Each trace corresponds to a different configuration regarding bond dimension, sweeping direction, and whether or not the disorientation constraint was included. For a more detailed discussion on the distribution of the obtained solutions, Fig. 5b provides a histogram of the Root Mean Square Error (RMSE) across all samples for the most effective configuration (bond dimension 32 and sweeping left), both with and without the disorientation constraint. The bin size of the histograms is set to 0.002. A comprehensive overview of the results from all individual trials for DMRG with  $N = 200$  plies is available in Fig. 6 in Appendix A.3.

We also monitored the time duration of each sweep. Fig. 5c illustrates the average sweep duration as a function of the bond dimension, with separate traces to distinguish between different sweeping directions and comparisons of cases with and without the disorientation constraint. For calculating the averages and standard deviations, the first and last 10 sweeps were excluded. To minimize the impact of background processes on the operating system on the duration of isolated sweeps, the lowest and highest 10% of times for all considered sweeps were also excluded. In the DMRG algorithm, the most computationally intensive step involves finding the eigenvector with the lowest eigenvalue of a matrix, which is a critical component of the local optimizations of the site tensors. The computational complexity for solving the eigenvalue problem generally scales as  $O(b^3)$ , where  $b$  is the bond dimension. This scaling behavior is consistent with that of the Lanczos algorithm used for the eigenvalue computations, assuming the number



**Fig. 4.** Results for the hardware-efficient approach (a) and DMRG (b). In both cases, the top row shows the average RMSE for the final results, and the bottom row shows the ratio of resulting stacks that coincided with the exact solution. The averages are taking over the results from all 15 sets of target lamination parameters with 5 trials performed for each.

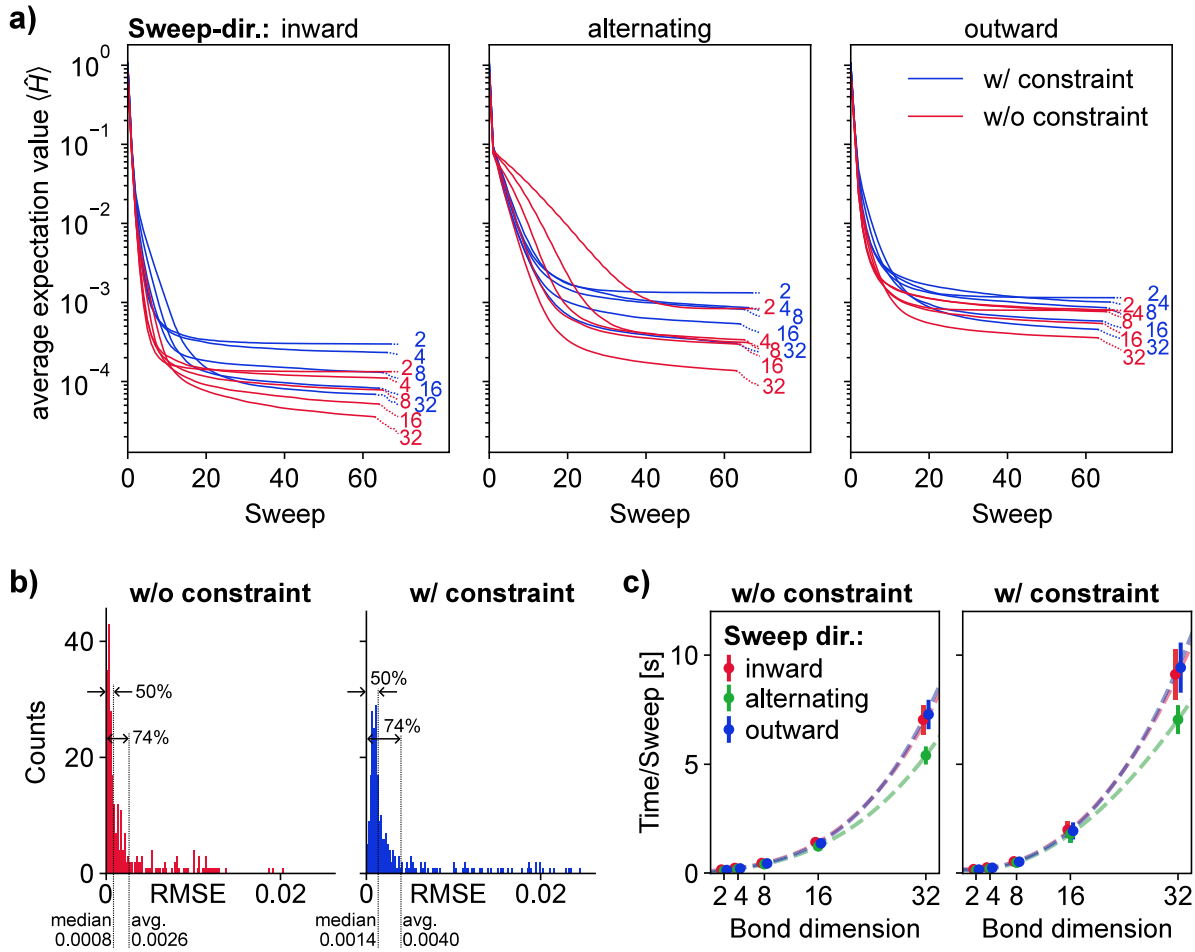
of non-zero matrix elements scales proportionally to the overall size of the matrix. However, due to the limited number of data points for different bond dimensions, definitive assertions about the actual observed scaling behavior are not possible. Nonetheless, for illustrative purposes, a third-order polynomial fit has been included in the figure to facilitate easier comparison of the different configurations.

### 5.3. Discussion

We will first discuss and compare the chosen methods for  $N = 6$  plies, then provide a more detailed analysis of DMRG for  $N = 200$  plies.

With  $N = 6$  plies, we observed for all three algorithms that increasing the number of repetitions  $N_r$  in the state-preparation circuit or raising the bond dimension in DMRG generally leads to better outcomes in terms of loss or RMSE, and the likelihood of





**Fig. 5.** Results from DMRG with  $N = 200$  plies. **(a)** Traces of expectation values of the loss function Hamiltonian, averaged over all samples and measurements after each sweep. Each line corresponds to a different bond dimension, denoted at the end of each trace. Red lines represent trials excluding the disorientation constraint and blue lines correspond to trials including the constraint. Solid lines represent the optimization phase with a constant maximum bond dimension. The successive reduction of the bond dimension at the final phase of the optimization is indicated by dotted lines that can be found at the right-hand end of each trace. **(b)** Histogram of the RMSE of the solutions for maximum bond-dimension 32 and inward direction. Counted were the points from all target lamination parameters and initial MPS. The bin-size is 0.002. The ranges of RMSE for the bottom half of the points, and up to the average RMSE are shown in both plots. Left: Without the disorientation constraint. Right: With the disorientation constraint. **(c)** Average sweep duration as a function of bond dimension. Left and right plots respectively correspond to scenarios without and with the disorientation constraint. Each sweep direction is represented by distinct points in different colors. A third-order polynomial fit is included as dashed lines, to easier distinguish trends for different configurations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

obtaining the exact state. Comparing Figs. 3 and 4, the hardware-efficient approach and DMRG appear to significantly outperform QAOA. Furthermore, QAOA is considerably more expensive to execute due to its substantially larger circuit depth. However, as the hardware-efficient approach and DMRG generally end up in a basis state, which often has insufficient accuracy according to the large standard deviations in Fig. 4, multiple trials with different initial parameters may be necessary to obtain a good solution [73,153–155]. Conversely, the final superposition of states in QAOA generally includes good solutions, which can be found with a sufficient number of measurements of the final superposition, making it less expensive than multiple trials of the entire optimization. Nevertheless, QAOA faces well-known issues like barren plateaus that may still require multiple trials and optimized strategies [114,115,117,156]. Additionally, we chose basic VQA examples for our demonstrations, and as more advanced methods are developed, further research will determine which strategies yield the best results when scaling up the number of plies to industry-relevant levels.

It is also important to note that we used exact state-vector simulation for the quantum algorithms. However, unlike DMRG, which is a classical algorithm, executing quantum algorithms on quantum hardware introduces various errors such as state-preparation and measurement (SPAM) errors and various noise channels during the execution of a quantum circuit [157]. Investigating strategies to manage these noise factors is a major research area, and future studies must address how these errors and mitigation strategies impact the design and effectiveness of quantum algorithms for the SSR problem.



We continue with a discussion of the results for  $N = 200$  plies. For  $N = 6$  plies, the solution space consists of  $d^N = 4096$  configurations, making it feasible to find the optimal solution. However, for larger numbers of plies, the configuration space expands exponentially, shifting the focus from identifying the optimal solution to finding a highly accurate solution. While current quantum hardware and quantum simulations are generally limited by the system size over which algorithms can be effectively executed. The scalable DMRG algorithm however can provide insights that are applicable to some VQA like the chosen hardware-efficient approach.

As shown in Fig. 5a, the high number of sweeps allow for studying the progression of the Hamiltonian expectation value during the optimization. In all configurations, there is a rapid decrease in the expectation value during the initial sweeps, followed by a significant slowdown in the optimization rate. The reduction in bond dimension, indicated by the dotted lines at the end of each trace in Fig. 5a, leads to a pronounced reduction in the expectation value of the loss function. This indicates that just before this reduction, the MPS represents a superposition of basis states that includes less optimal states, which are then filtered out during the bond dimension reduction.

While the averaged traces provide an overview, they do not reveal the actual distribution of all resulting lamination parameters for each specific configuration. The histograms in Fig. 5b and Fig. 6 demonstrate how the majority of the trial significantly outperform the worst. Furthermore, Fig. 6 shows that the performance does not necessarily correlate between different configurations in initial MPS, bond-dimension and sweeping direction. Consequently, warm-starting and multiple trials with various configurations can potentially enhance the accuracy of the result significantly.

Comparing the loss function expectation values of the results together with the duration of a sweep in Fig. 5c for distinct bond dimensions shows a clear trade-off between accuracy and runtime. The main contributor to the runtime is the eigen-decomposition, which has a cubic dependence on the bond dimension. This increase makes DMRG impractically demanding for high bond dimensions. In contrast, we only need a logarithmic increase in the circuit depth  $N$ , in the hardware-efficient circuit to grow the effective bond dimension. However, this does not yet consider the optimization procedure, and the large number of parameters to be optimized might counteract this gain.

Considering the practical aspects of our research approach, it is crucial to highlight that our numerical trials were conducted using the DMRG method, based on the implementation provided by the *ITensors.jl* library, on a CPU without optimizing for performance. Implementing parallelization techniques, GPU support, sparse matrix optimizations, or problem-specific methods could lead to substantial performance enhancements. Additionally, adjusting other DMRG settings, such as hyperparameters in the eigenvalue solver, might further impact performance. Furthermore, to take advantage of the varying performance of different operational configurations, techniques like warm-starting can potentially reduce the requirements for finding a good solution in the majority of cases [158].

In conclusion, our study lays the groundwork for further investigation into the potential of DMRG as a viable alternative for stacking sequence retrieval. To achieve this, it is essential to incorporate the specific manufacturing constraints discussed earlier, such as the contiguity constraint and the 10%-rule, which are crucial for real-world applications [39–41]. Alongside these additions, implementing performance enhancements such as parallelization, GPU support, and optimization of algorithm settings is critical for this method to be competitive with existing stacking sequence retrieval techniques. Nevertheless, the results from our study offer encouraging signs that tensor networks and quantum algorithms could play a significant role in advancing quantum and quantum-inspired methods for laminate design.

## 6. Conclusions and outlook

In this work, we have developed novel approaches based on quantum computing and DMRG to tackle the stacking sequence retrieval problem, a notable challenge in the field of laminated composite design.

We initiated our study by developing a quantum representation of the SSR problem, adapting it for quantum computing methods. Initially, the problem was formalized as an integer optimization problem, where ply-angle sequences were represented as a list of integers, and the loss function was defined to quantify the mean square error between the lamination parameters of a given configuration and the target parameters. These stacking sequences were subsequently embedded in a Hilbert space, with each basis state corresponding to a distinct stacking sequence, alongside a Hamiltonian that encapsulated the loss function as a Hermitian operator. Various manufacturing constraints and their implementation as penalty terms in the loss function were also examined. The translation of these penalty terms into Hamiltonian operators revealed diverse forms of subsystem interactions across individual plies.

We then proceeded to validate our approach through numerical simulations. For this purpose, we selected two variational quantum algorithms, QAOA and a hardware-efficient approach for which we performed state-vector simulations. The latter shares similarities with the classical DMRG algorithm which we selected as a third algorithm for our study. We explored relevant concepts regarding the implementation of these algorithms and assessed their impact on algorithm performance. Simulations were conducted for a modest number of  $N = 6$  plies, which exhibited distinct behaviors between QAOA and notable similarities between the hardware-efficient approach and DMRG. The efficient implementation of DMRG further facilitated simulations for a larger laminate with  $N = 200$  plies. Each algorithm demonstrated varying degrees of success in identifying good solutions, affirming that our quantum formulation of the SSR problem is a viable approach.

While our study on a single panel was conducted with the future goal of addressing more complex scenarios involving multiple panels, the results from employing the well-known DMRG algorithm in this new scenario are promising even for a single panel. However, to establish its competitiveness, several key aspects need to be addressed in future work. A critical aspect is the integration

of additional manufacturing constraints, which are vital for real-world applications, and evaluating whether the DMRG algorithm can efficiently manage these complexities. Moreover, implementing performance optimizations will be crucial in assessing whether this approach can rival existing classical methods in efficacy.

A thorough comparison with existing state-of-the-art algorithms will determine whether DMRG is a serious contender. Such comparisons might also reveal advantages of DMRG's heuristic nature that could benefit specialized SSR strategies. For example, employing MPS as in DMRG allows for efficiently representing a superposition of stacking sequences, which could enable novel specialized strategies for multi-panel designs. An MPS consisting of approximate solutions can be obtained, for instance, by stopping the DMRG algorithm early before convergence. Future approaches might also explore more sophisticated tensor network configurations for multi-panel laminates, making tensor network methods a contender to quantum algorithms in these complex cases. Conversely, a collaborative exploration of tensor networks and quantum algorithms may unveil new insights, propelling advancements that leverage the strengths of both methodologies.

The quantum algorithms selected for our simulations, though basic, are valuable for validating our approach and represent just two examples within a broad spectrum of existing and potential quantum computing methods, with new specialized strategies being developed rapidly. Therefore, our simulations offer only a limited view of the future potential of quantum methods for SSR. Nevertheless, a key motivation of this work is to establish it as a promising candidate for industry-relevant applications of quantum computation. By presenting our findings to the broader community, we aim to stimulate the development of quantum algorithms tailored to industry-relevant problems, thereby fostering real-world impact.

Our exploration into the application of quantum computing for stacking sequence retrieval, particularly the process of selecting optimal sequences from a set of discrete choices, may have implications beyond laminated composite design. For example, the definition of the lamination parameters in Section 2.2 resembles the weighted-sum model in multi-criteria decision making, which is employed in a variety of applications [159–161]. This suggests that the methodologies and insights gained from our study can potentially inform approaches to quantum computing in other domains, thereby contributing to the progress in the practical application of quantum computational methods.

In conclusion, this research establishes a foundation for a further exploration into the utility of quantum algorithms for stacking sequence retrieval and laminate design, by developing a quantum representation of the problem. Moreover, we introduced tensor networks as a potential quantum-inspired approach. Importantly, our numerical results substantiate the efficacy of our methods, and provide valuable insights into benefits and drawbacks of quantum and quantum-inspired methods for stacking sequence retrieval. Continued research is crucial to fully understand the role these advanced computational techniques might play in the future of laminate design.

## Code and data availability

The code for performing the experiments in Section 5 can be found in a [GitHub repository](#) [144]. The generated data is available can be found in on [4TU.ResearchData](#) [145].

## CRediT authorship contribution statement

**Arne Wulff:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Boyang Chen:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Matthew Steinberg:** Writing – review & editing, Methodology. **Yinglu Tang:** Writing – review & editing, Supervision, Funding acquisition. **Matthias Möller:** Writing – review & editing, Supervision, Funding acquisition. **Sebastian Feld:** Writing – review & editing, Supervision, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used OpenAI's ChatGPT exclusively for language suggestions and corrections. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

## Acknowledgments

We thank the remaining members of the QAIMS lab, Swapan Madabhushi Venkata, Koen Mesman and Philip Würzner, for valuable discussions regarding this work. We thank the Faculty for Aerospace Engineering at the Delft University of Technology for their funding support of the QAIMS lab.

## Appendix A

### A.1. Terms in the penalty for the 10%-rule

In this section, we have a closer look into penalties to ensure the 10%-rule, as discussed in Section 3.2. There, in Eq. (28), we defined a possible penalty function to depend on the number of missing plies to satisfy the constraint:

$$H_{10\%-rule}(\vec{s}) = \begin{cases} 0, & \text{if } \sum_{n=1}^N \delta_{s_n,t} \geq N_t, \\ p(N_t - \sum_{n=1}^N \delta_{s_n,t}), & \text{else,} \end{cases} \quad (74)$$

where  $N_t$  is the minimum required number of plies with state  $t$ .

An intuitive approach to construct a suitable penalty function stems from the following realization: If the constraint is violated, then there exists at least one subset with  $N - N_t + 1$  plies, that does not include any plies of state  $t$ . We can therefore simply define a suitable penalty function by checking all subsets of  $N - N_t + 1$  plies and add a penalty, if no plies with state  $t$  are present. For example, if we denote the set of all  $k$ -element subsets of  $\{1, \dots, N\}$  with  $C(N, k)$ , we can define the penalty function as:

$$H_{10\%-rule}(\vec{s}) = \sum_{A \in C(N, N-N_t+1)} \prod_{n \in A} (1 - \delta_{s_n,t}). \quad (75)$$

Here,  $(1 - \delta_{s_n,t}) = 1$  exactly if ply  $n$  is not in state  $t$ ,  $s_n \neq t$ , and otherwise the expression is 0. For the terms, we thus add a penalty  $\prod_{n \in A} (1 - \delta_{s_n,t}) = 1$ , exactly if for all plies  $n \in A$  we have  $s_n \neq t$ . Since this definition is symmetric under permutations of the plies, and therefore the result only depends on the difference in the number of current and required plies in state  $t$ , it represents a concrete example of the penalty function in Eq. (28). Furthermore, the individual terms translate to operators on  $N - N_t + 1$  subsystems in the quantum representation.

As discussed in Section 3.2, these high-order operators can act unfavorably in certain quantum algorithms, especially those that require implementing the Hamiltonian as operations on the qubits. The question now is: Can we do better? As we show in this section, this is unfortunately not the case.

First, we will show, that a penalty function, as defined in Eq. (28), will necessarily lead to terms of order  $N - N_t + 1$ . However, this definition is already somewhat restrictive, as it only allows penalty functions, that are symmetric under the permutation of the plies. From an optimization point of view, it makes sense that states with the same discrepancy to satisfying the constraint produce the same penalty. However, there might be penalty functions, that are not symmetric under permutation, but might be implementable with lower-order operators. In the final part of this section, we will show that any suitable operator for the 10%-rule will require operators acting on at least  $N - N_t + 1$  subsystem, regardless of symmetry considerations.

#### A.1.1. Using the defined penalty function in eq. (28)

In this section, we show, that a decomposition into local operators of the penalty function for the 10%-rule, as defined in Eq. (28) in Section 3.2:

$$H_{\text{penalty}}^{10\%-rule}(\vec{s}) = \begin{cases} 0, & \text{if } \sum_{n=1}^N \delta_{s_n,t} \geq N_t, \\ p(N_t - \sum_{n=1}^N \delta_{s_n,t}), & \text{else,} \end{cases} \quad (76)$$

always includes terms of order  $N - N_t + 1$ .

For this purpose, we extend the definition of  $p(x)$  also for valid states, where  $x \geq N_t$ . If we write  $p(x)$  as a polynomial:

$$p(x) = \sum_{k=0}^{\deg p} a_k x^k, \quad (77)$$

– any other form would immediately yield a Taylor expansion up to infinite order – then each  $x \in \{N_t, N_t + 1, \dots, N\}$  must be a root of the polynomial:

$$p(x) = 0. \quad (78)$$

Thus, since  $p(x)$  has at least  $N - N_t + 1$  roots, it must have at least an equivalent degree:

$$\deg p \geq N - N_t + 1. \quad (79)$$

For a state  $\vec{s} \in S$ , we have:

$$x = N_t - \sum_{n=1}^N \delta_{s_n,t}, \quad (80)$$

such that a power  $x^k$  yields:

$$x^k = \left( N_t - \sum_{n=1}^N \delta_{s_n,t} \right)^k \quad (81)$$

$$= (-1)^k \left( \sum_{n=1}^N \delta_{s_n,t} \right)^k + (-1)^{k-1} k N_t \left( \sum_{n=1}^N \delta_{s_n,t} \right)^{k-1} + (-1)^{k-2} \frac{1}{2} (k-1) k N_t^2 \left( \sum_{n=1}^N \delta_{s_n,t} \right)^{k-2} + \dots \quad (82)$$

Furthermore:

$$\left( \sum_{n=1}^N \delta_{s_n,t} \right)^k = \sum_{n_1, \dots, n_k=1}^N \delta_{s_{n_1},t} \delta_{s_{n_2},t} \dots \delta_{s_{n_k},t}. \quad (83)$$

Since  $p(x)$  contains terms with  $k = N - N_t + 1$ , the penalty function includes terms with  $\delta_{s_{n_1},t} \delta_{s_{n_2},t} \dots \delta_{s_{n_k},t}$ , which translate to operators  $|tt \dots t\rangle \langle tt \dots t|_{n_1 n_2 \dots n_k}$ . Therefore, the according Hamiltonian will have terms of order  $N - N_t + 1$ .

#### A.1.2. Any penalty function will include high-order terms

Above, we showed that the specified penalty function in Eq. (28) leads to terms of order  $N - N_t + 1$ . However, the penalty function is inherently symmetric under permutations of the elements in a state  $\vec{s} \in S$ . Here, we show that indeed any penalty function, regardless of symmetry considerations, must include terms of order  $N - N_t + 1$  or larger.

First, we note that non-diagonal local operators produce off-diagonal elements when combined to higher operators. We can therefore focus on diagonal operators. Furthermore, these diagonal operators directly translate to Kronecker-deltas in the classical penalty function, as can be seen from the matrix element of the operator:

$$\langle s | \left( \sum_{r=1}^d c_r |r\rangle \langle r| \right) | s \rangle = c_s = \sum_{r=1}^d c_r \delta_{r,s} \quad (84)$$

where  $r, s \in \{1, \dots, d\}$  are states on a single subsystems. We can therefore focus on expansions of the classical penalty function in terms of Kronecker-deltas.

In order to demonstrate that any penalty function for the 10%-rule requires at least  $N - N_t + 1$ -order terms, we show that any penalty function with lower order will be insufficient. We accomplish this by considering a function that only includes terms of order up to  $k = N - N_t$ :

$$Z(\vec{s}) = \sum_{A \in \mathcal{P}_k(N)} \sum_{\vec{a} \in \{1, \dots, d\}^{|A|}} \zeta_{A,\vec{a}} \delta_{\vec{s}[A],\vec{a}}, \quad (85)$$

where  $\zeta_{A,\vec{a}} \in \mathbb{R}$  are the coefficients of the expansion. Here we used the following notation:  $\mathcal{P}_k(N)$  is the power set of  $\{1, \dots, N\}$ , up to cardinality  $k$ :

$$\mathcal{P}_k(N) = \{A \subseteq \{1, \dots, N\} : |A| \leq k\}. \quad (86)$$

We use subsets  $A \in \mathcal{P}_k(N)$  to select subsequence of elements  $s_n$  of the state  $\vec{s}$  for  $n \in \{1, \dots, N\}$ , which we denote with:

$$\vec{s}[A] = (s_{n_1}, s_{n_2}, \dots, s_{n_{|A|}}) \in \{1, \dots, N\}^{|A|}, \quad (87)$$

where  $A = \{n_1, n_2, \dots, n_{|A|}\}$  and  $n_1 < n_2 < \dots < n_{|A|}$ . With the Kronecker-delta:

$$\delta_{\vec{s}[A],\vec{a}} = \delta_{s_{n_1},a_1} \delta_{s_{n_2},a_2} \dots \delta_{s_{n_{|A|}},a_{|A|}}, \quad (88)$$

we match the selected states of  $\vec{s}$  to a defined vector  $\vec{a} \in \{1, \dots, d\}^{|A|}$ . In an operator representation, these Kronecker-deltas are converted into operators  $|a_1 a_2 \dots a_{|A|}\rangle \langle a_1 a_2 \dots a_{|A|}|_{n_1 n_2 \dots n_{|A|}}$ . For our proof, we show that if this function  $Z(\vec{s})$  has roots on all valid states, it will necessarily also have roots for at least one constraint-violating state, which makes it unsuitable as a penalty function.

Before we continue, we first note, that:

$$\delta_{s_n,\alpha} = \delta_{s_n,\alpha} \sum_{\beta=1}^d \delta_{s_n',\beta} = \sum_{\beta=1}^d \delta_{s_n,\alpha} \delta_{s_n',\beta}, \quad (89)$$

from which follows, that terms with lower order can be converted into terms with higher order. Therefore, instead of allowing terms with up to order  $k$  in  $Z(\vec{s})$ , we can focus on  $Z(\vec{s})$  containing only terms of exactly order  $k$ :

$$Z(\vec{s}) = \sum_{A \in C(N,k)} \sum_{\vec{a} \in \{1, \dots, d\}^{|A|}} \zeta_{A,\vec{a}} \delta_{\vec{s}[A],\vec{a}}, \quad (90)$$

where  $C(N, k)$  denotes the set of all combinations of  $k$  object from  $\{1, \dots, N\}$  without replacement:

$$C(N, k) = \{A \in \mathcal{P}(N) : |A| = k\}. \quad (91)$$

The valid states  $\vec{s}$ , with:

$$\sum_{n=1}^N \delta_{s_n,t} \geq N_t = N - k, \quad (92)$$

must not add a penalty:

$$Z(\vec{s}) = 0, \quad (93)$$

which defines a system of linear equations in  $\zeta_{A,\vec{a}}$  in Eq. (90), for which the rows of the according matrix are given by the  $\delta_{\vec{s}[A],\vec{a}}$ . We will show, that there exists a state  $\vec{r} \in S$  with less than  $N_t$  occurrences of state  $t$ , and thus should add penalty, but evaluates to  $Z(\vec{r}) = 0$ . We do so, by showing, that  $\delta_{\vec{r}[A],\vec{a}}$  can be written as a linear combination of  $\delta_{\vec{s}[A],\vec{a}}$  with valid states  $\vec{s}$ .

For this purpose, we define vectors  $\vec{w}^{(\Gamma)} \in S$  containing only state  $t$  and another state  $u \in \{1, \dots, d\}$  with  $u \neq t$ , where the last  $N - k - 1$  elements are all  $t$ :

$$\vec{w}^{(\Gamma)} = \vec{x}^{(\Gamma)} \oplus (t, \dots, t). \quad (94)$$

Here,  $\Gamma \in \mathcal{P}(k+1)$  is a subset of  $\{1, \dots, k+1\}$  and  $\vec{x}^{(\Gamma)} \in \{t, u\}^{k+1}$  is defined by:

$$x_n^{(\Gamma)} = \begin{cases} t, & \text{if } n \in \Gamma, \\ u, & \text{if } n \notin \Gamma. \end{cases} \quad (95)$$

We will show that  $\delta_{\vec{r}[A],\vec{a}}$  for:

$$\vec{r} = \vec{w}^{(\emptyset)} = (u, \dots, u, t, \dots, t), \quad (96)$$

which contains  $N - k - 1 = N_t - 1$  occurrences of  $t$ , is a linear combination of  $\delta_{\vec{w}^{(\Gamma)}[A],\vec{a}}$  with  $\Gamma \neq \emptyset$ , such that  $\vec{w}^{(\Gamma)}$  constraints at least  $N - k = N_t$  occurrences of  $t$ .

For a given  $A = \{n_1, n_2, \dots, n_k\} \in C(N, k)$  and  $\vec{a} \in \{1, \dots, d\}^{|A|}$ , we assume that  $n_1, \dots, n_m \leq k+1$  and  $n_{m+1}, \dots, n_k > k+1$ . Then for any  $\Gamma \in \mathcal{P}(k+1)$ :

$$\delta_{\vec{w}^{(\Gamma)}[A],\vec{a}} = \delta_{x_{n_1}^{(\Gamma)},a_1} \delta_{x_{n_2}^{(\Gamma)},a_2} \dots \delta_{x_{n_m}^{(\Gamma)},a_m} \delta_{t,a_{m+1}} \dots \delta_{t,a_k}. \quad (97)$$

Note that for  $\vec{r}$ , where  $\Gamma = \emptyset$ , we get:

$$\delta_{\vec{r}[A],\vec{a}} = \delta_{u,a_1} \dots \delta_{u,a_m} \delta_{t,a_{m+1}} \dots \delta_{t,a_k}. \quad (98)$$

If we sum the Kronecker-deltas for all  $\vec{x}^{(\Gamma)}$  with the same number  $\alpha = |\Gamma|$  of  $t$ , we obtain:

$$\sum_{\Gamma \in C(k+1,\alpha)} \delta_{\vec{w}^{(\Gamma)}[A],\vec{a}} = \delta_{t,a_{m+1}} \dots \delta_{t,a_k} \sum_{\Gamma \in C(k+1,\alpha)} \delta_{x_{n_1}^{(\Gamma)},a_1} \delta_{x_{n_2}^{(\Gamma)},a_2} \dots \delta_{x_{n_m}^{(\Gamma)},a_m}. \quad (99)$$

We can evaluate the sum on the right side, by considering that  $\vec{x}^{(\Gamma)}$ , which has length  $k+1$ , has to match  $\vec{a}$  on the sites defined by  $n_1, \dots, n_m$ , and that  $\gamma = \text{count}(t, (a_1, \dots, a_m))$  occurrences of  $t$  are already bound in  $\vec{a}$ . Therefore, we need to count all combinations, where we place  $\alpha - \gamma$  instances of  $t$  on the remaining  $(k+1) - m$  sites:

$$\sum_{\Gamma \in C(k+1,\alpha)} \delta_{\vec{w}^{(\Gamma)}[A],\vec{a}} = \delta_{t,a_{m+1}} \dots \delta_{t,a_k} \binom{(k+1) - m}{\alpha - \gamma}, \quad (100)$$

where the brackets on the right side denote the binomial coefficient.

Now, we calculate the sum of all these expressions for  $\alpha = 1, 2, \dots, k+1$  with alternating signs:

$$\begin{aligned} & \sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \sum_{\Gamma \in C(k+1,\alpha)} \delta_{\vec{w}^{(\Gamma)}[A],\vec{a}} \\ &= \delta_{t,a_{m+1}} \dots \delta_{t,a_k} \sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \binom{(k+1) - m}{\alpha - \gamma} \\ &= \delta_{t,a_{m+1}} \dots \delta_{t,a_k} \sum_{\alpha=1-\gamma}^{k+1-\gamma} (-1)^{\alpha+\gamma-1} \binom{(k+1) - m}{\alpha} \\ &= \delta_{t,a_{m+1}} \dots \delta_{t,a_k} (-1)^{\gamma-1} \sum_{\alpha=\max(0,1-\gamma)}^{k+1-m} (-1)^{\alpha} \binom{(k+1) - m}{\alpha}, \end{aligned} \quad (101)$$

where in the second line, we shifted the index  $\alpha$  and in the third line, we trimmed to index  $\alpha$  to reflect, that the binomial coefficient vanishes for  $\alpha < 0$  or  $\alpha > (k+1) - m$ . We distinguish two cases: If the number of  $t$  in  $(a_1, \dots, a_m)$  is non-zero, then the expression vanishes:

$$\begin{aligned} & \sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \sum_{\Gamma \in C(k+1,\alpha)} \delta_{\vec{w}^{(\Gamma)}[A],\vec{a}} \\ &= \delta_{t,a_{m+1}} \dots \delta_{t,a_k} (-1)^{\gamma-1} \sum_{\alpha=0}^{k+1-m} (-1)^{\alpha} \binom{(k+1) - m}{\alpha} \\ &= 0, \end{aligned} \quad (102)$$

where we used the identity  $\sum_{j=1}^n (-1)^j \binom{n}{j} = 0$ . If  $(a_1, \dots, a_m)$  does not contain any  $t$ , then we need to make sure to include the term  $\alpha = 0$ , in order to use this identity:

$$\begin{aligned} & \sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \sum_{\Gamma \in C(k+1, \alpha)} \delta_{\vec{w}^{(\Gamma)}[A], \vec{a}} \\ &= \delta_{t, a_{m+1}} \cdots \delta_{t, a_k} (-1)^{\gamma-1} \sum_{\alpha=1}^{k+1-m} (-1)^\alpha \binom{(k+1)-m}{\alpha}, \\ &= \delta_{t, a_{m+1}} \cdots \delta_{t, a_k} (-1)^{\gamma-1} \left( \sum_{\alpha=0}^{k+1-m} (-1)^\alpha \binom{(k+1)-m}{\alpha} - (-1)^0 \binom{(k+1)-m}{0} \right) \\ &= \delta_{t, a_{m+1}} \cdots \delta_{t, a_k}, \end{aligned} \quad (103)$$

where in the last line, we included that the number of  $t$  in  $(a_1, \dots, a_m)$  vanishes in this case,  $\gamma = 0$  and thus  $(-1)^{\gamma-1} = -1$ . We can summarize the results for the two cases as:

$$\sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \sum_{\Gamma \in C(k+1, \alpha)} \delta_{\vec{w}^{(\Gamma)}[A], \vec{a}} = \delta_{u, a_1} \cdots \delta_{u, a_m} \delta_{t, a_{m+1}} \cdots \delta_{t, a_k}, \quad (104)$$

which is exactly  $\delta_{\vec{r}[A], \vec{a}}$  in Eq. (98):

$$\delta_{\vec{r}[A], \vec{a}} = \sum_{\alpha=1}^{k+1} (-1)^{\alpha-1} \sum_{\Gamma \in C(k+1, \alpha)} \delta_{\vec{w}^{(\Gamma)}[A], \vec{a}}. \quad (105)$$

Therefore  $\delta_{\vec{r}[A], \vec{a}}$  is linear combination of  $\delta_{\vec{w}^{(\Gamma)}[A], \vec{a}}$ . As a consequence, if we choose  $\zeta_{A, \vec{a}}$  in Eq. (90), such that  $Z(\vec{w}^{(\Gamma)}) = 0$  for all valid states with  $\Gamma \neq \emptyset$ , then  $Z(\vec{r}) = 0$  for the constraint violating state  $\vec{r}$ . Thus, a suitable penalty function  $Z(\vec{s})$  cannot be constructed with only terms of order less than  $N - N_t + 1$ . As a consequence, any penalty Hamiltonian will contain terms of order  $N - N_t + 1$  or more.

## A.2. Symmetries of stacking sequences with respect to lamination parameters and constraints

In Section 5, we selected target lamination parameters for  $N = 6$  plies that correspond to stacking sequences which cannot be transformed into each other through symmetry transformations of the lamination parameters and the disorientation constraint. In this section, we explore these symmetry transformations and their impact on SSR.

Given their definition, the set of lamination parameters is symmetric under specific rotations and reflections of the space. As long as an SSR algorithm inherently treats all ply-angles equally, it can be expected to perform similarly for target lamination parameters that can be transformed into each other through these symmetry transformations. For numerical trials of the algorithm, it therefore makes sense to choose target parameters that are distinctly invariant to ensure a broad representation of potential behaviors. We begin by examining the symmetries of the lamination parameter space with arbitrary continuous ply-angles and then discuss the scenario with discrete parameters.

The 4-dimensional rotation:

$$R(\chi) = \begin{pmatrix} \cos(2\chi) & -\sin(2\chi) & 0 & 0 \\ \sin(2\chi) & \cos(2\chi) & 0 & 0 \\ 0 & 0 & \cos(4\chi) & -\sin(4\chi) \\ 0 & 0 & \sin(4\chi) & \cos(4\chi) \end{pmatrix} \quad (106)$$

applied to the set of lamination parameters of a ply-angle sequence  $\vec{\phi} = (\phi_1, \dots, \phi_N)$  for a specific  $X = A, B, D$ :

$$\begin{pmatrix} v_1^X \\ v_2^X \\ v_3^X \\ v_4^X \end{pmatrix} = \sum_{n=1}^N \alpha^X \begin{pmatrix} \cos(2\phi_n) \\ \sin(2\phi_n) \\ \cos(4\phi_n) \\ \sin(4\phi_n) \end{pmatrix} \quad (107)$$

produces a constant shift of  $\chi$  for all ply angles  $\phi_n$ :

$$R(\chi) \begin{pmatrix} v_1^X \\ v_2^X \\ v_3^X \\ v_4^X \end{pmatrix} = \sum_{n=1}^N \alpha^X \begin{pmatrix} \cos(2(\phi_n + \chi)) \\ \sin(2(\phi_n + \chi)) \\ \cos(4(\phi_n + \chi)) \\ \sin(4(\phi_n + \chi)) \end{pmatrix}. \quad (108)$$

As the shifted angle sequence  $(\phi_1 + \chi, \dots, \phi_N + \chi)$  remains a valid ply angle sequence, the set of lamination parameters is symmetric under performing rotation  $R(\chi)$  on all  $X = A, B, D$  simultaneously with the same angle  $\chi$ .

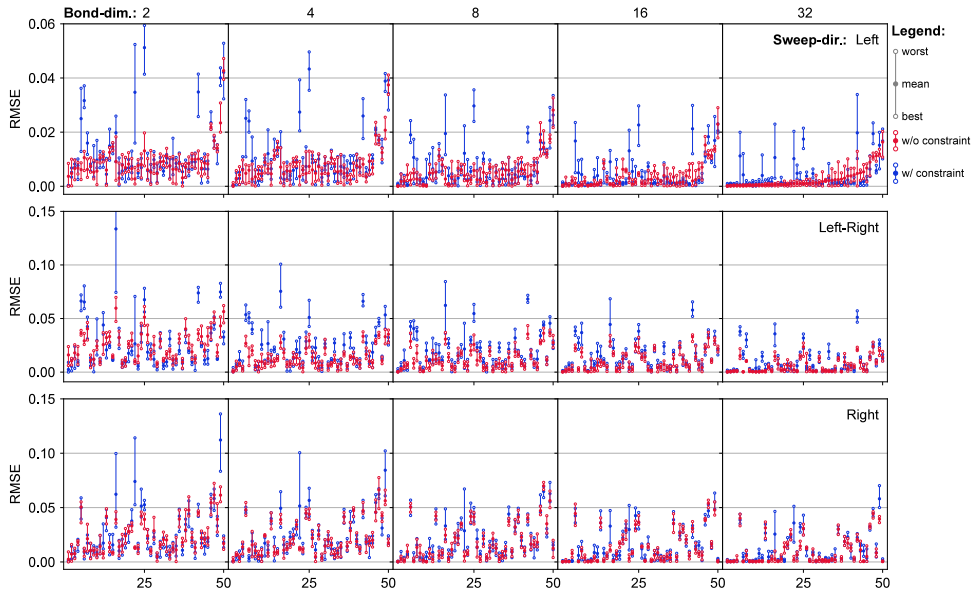


Fig. 6. Summary of all DMRG samples organized in a tabular format. Rows correspond to different sweep directions (inward, alternating, outward) and columns represent the bond dimensions ranging from 2 to 32. For each sample, the best, worst, and average results are shown. The samples are sorted by the average values for bond dimension 32 and sweep direction left, without the constraint (top right).

Additionally, the set of lamination parameters possesses mirror symmetry by applying the reflection:

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (109)$$

to all  $X = A, B, D$  simultaneously, which coincides with changing the sign of the ply angles:

$$T \begin{pmatrix} v_1^X \\ v_2^X \\ v_3^X \\ v_4^X \end{pmatrix} = \sum_{n=1}^N \alpha^n \begin{pmatrix} \cos(-2\phi_n) \\ \sin(-2\phi_n) \\ \cos(-4\phi_n) \\ \sin(-4\phi_n) \end{pmatrix} \quad (110)$$

Given that for  $\chi \in (90^\circ, 90^\circ]$  the rotations  $R(\chi)$  are distinct, the group generated by the rotations  $R(\chi)$  and the reflection  $T$  is isomorphic to the orthogonal group  $O(2)$ . Note that here the rotations have a periodicity of  $180^\circ$ .

Next, we examine the case of  $d$  evenly-spaced discrete angles  $\Theta = \{0^\circ, \Delta\theta, 2\Delta\theta, \dots, 180^\circ - \Delta\theta\}$  where  $\Delta\theta = \frac{180^\circ}{d}$ . This includes the commonly used four angles  $\{0^\circ, +45^\circ, 90^\circ, -45^\circ\}$ , with  $-45^\circ$  being equivalent to  $135^\circ$ . In this setting, the shift  $\chi$  induced by a rotation  $R(\chi)$  must also produce a valid ply-angle sequence, thus  $\chi$  is restricted to the set  $\Theta$ . The reflection  $T$  continues to be a symmetry transformation of the lamination parameters. With these restrictions, the group generated by the permissible rotations and the reflection forms a subgroup of the previous group, isomorphic to the dihedral group  $D_d$ , which is the symmetry group of a polygon with  $d$  vertices.

The symmetry transformations may be further constrained by manufacturing requirements. For the disorientation constraint, shifting all ply-angles does not alter the ply-angle difference between two plies; thus, a valid ply-angle sequence remains valid under these transformations, while a constraint-violating state remains invalid. However, this is not generally the case for the balanced laminates constraint. For instance, requiring an equal number of  $+45^\circ$  and  $-45^\circ$  plies, a rotation of  $45^\circ$  swaps these angles with  $0^\circ$  and  $90^\circ$ , which may result in a ply-angle sequence alternating between valid and invalid states. Consequently, an SSR algorithm may perform differently under transformations that are not symmetries of the constraint.

In our simulations involving the four angles  $\theta_1 = 0^\circ, \theta_2 = +45^\circ, \theta_3 = 90^\circ, \theta_4 = -45^\circ$  and only a disorientation constraint, the transformations of stacking sequences  $\vec{s} = (s_1, \dots, s_N) \in S$  corresponding to allowed rotations can be expressed as adding an integer modulo 4 to all ply-states  $s_n$ . Possible reflections include swapping states 1 and 3, or states 2 and 4. For the selected target lamination parameters for  $N = 6$ , these stacking sequences cannot be transformed into each other using any combination of these transformations.

At the beginning of this section, we noted that an SSR algorithm is expected to behave similarly for target parameters that are equivalent under these symmetry transformations, provided it does not discriminate between different ply angles. This is certainly the case for the DMRG algorithm, which directly operates on the ply-states. For the QAOA algorithm and the hardware-efficient



approach, consideration must be given to the qubit-encoding, the parameterized quantum circuit, and its optimization. In our state-vector simulations, qubit states  $|0\rangle$  and  $|1\rangle$  are considered interchangeable, but on actual quantum hardware, these states often do not exhibit completely symmetric behavior. For instance, one of these states may be a higher energy state prone to erroneously falling into the lower energy state ( $T_1$  decay), potentially leading to performance differences between encodings that involve different durations of the qubits being in the higher energy state.

Moreover, even within state-vector simulations, performance differences may occur for equivalent target parameters. The QAOA algorithm applies operations to all qubits simultaneously and implements eigenvalue-dependent phases, which should make it less susceptible to encoding-specific features. However, the local optimizations in the hardware-efficient approach may exhibit a dependency on the encoding, potentially resulting in distinct behaviors for target parameters that can be transformed into each other. For this reason, we have attempted to incorporate the rotational symmetries into our encoding at least partially. In this encoding, for the qubits representing one ply, flipping one qubit results in a rotation of  $+45^\circ$  or  $-45^\circ$ , while flipping both qubits results in a rotation of  $90^\circ$ . This remains consistent even under the aforementioned symmetry transformations.

### A.3. Diagram of all measured samples for DMRG with $N = 200$ plies

Fig. 6 offers a tabular depiction of the outcomes of all measured samples. The rows and columns respectively represent different sweep directions and diverse bond dimensions. Here, the rows signify different sweep directions, and the columns represent the diverse bond dimensions, ranging from 2 to 32. The samples are sorted by the average values for bond dimension 32 and sweep direction left, without the constraint.

## References

- [1] Jingqiao Zhang, A.C. (Arthur C.) Sanderson, Computational intelligence in expensive optimization problems, vol. 2, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-10700-9, 2010, p. 164, <http://dx.doi.org/10.1007/978-3-642-10701-6>.
- [2] Rajkumar Roy, Srichand Hinduja, Roberto Teti, Recent advances in engineering design optimisation: Challenges and future trends, CIRP Ann (ISSN: 00078506) 57 (2008) 697–715, <http://dx.doi.org/10.1016/j.cirp.2008.09.007>.
- [3] Richard M. Karp, Reducibility among combinatorial problems, Springer US, Boston, MA, 1972, pp. 85–103, [http://dx.doi.org/10.1007/978-1-4684-2001-2\\_9](http://dx.doi.org/10.1007/978-1-4684-2001-2_9).
- [4] Sukhpal Singh Gill, et al., Quantum computing: A taxonomy, systematic review and future directions, Softw. - Pract. Exp. (ISSN: 0038-0644) 52 (2022) 66–114, <http://dx.doi.org/10.1002/spe.3039>.
- [5] Yang Lu, et al., Quantum computing and industrial information integration: A review, J. Ind. Inf. Integr. (ISSN: 2452414X) 35 (2023) 100511, <http://dx.doi.org/10.1016/j.jii.2023.100511>.
- [6] Evidence for the utility of quantum computing before fault tolerance, Nature (ISSN: 0028-0836) 618 (2023) 500–505, <http://dx.doi.org/10.1038/s41586-023-06096-3>.
- [7] Michael A. Nielsen, Isaac L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, ISBN: 9781107002173, 2012, <http://dx.doi.org/10.1017/CBO9780511976667>.
- [8] Richard P. Feynman, Simulating physics with computers, Internat. J. Theoret. Phys. (ISSN: 0020-7748) 21 (1982) 467–488, <http://dx.doi.org/10.1007/BF02650179>.
- [9] Emily Grumbling, Mark Horowitz (Eds.), Quantum Computing, National Academies Press, Washington, D.C., ISBN: 978-0-309-47969-1, 2019, <http://dx.doi.org/10.17226/25196>.
- [10] Akshay Ajagekar, Fengqi You, Quantum computing for energy systems optimization: Challenges and opportunities, Energy (ISSN: 03605442) 179 (2019) 76–89, <http://dx.doi.org/10.1016/j.energy.2019.04.186>.
- [11] Sam McArdle, et al., Quantum computational chemistry, Rev. Modern Phys. (ISSN: 0034-6861) 92 (2020) 015003, <http://dx.doi.org/10.1103/RevModPhys.92.015003>.
- [12] Hari P. Paudel, et al., Quantum computing and simulations for energy applications: Review and perspective, ACS Eng. Au (ISSN: 2694-2488) 2 (2022) 151–196, <http://dx.doi.org/10.1021/acseengineeringau.1c00033>.
- [13] Annarita Giani, Zachary Eldredge, Quantum computing opportunities in renewable energy, SN Comput. Sci. (ISSN: 2662-995X) 2 (2021) 393, <http://dx.doi.org/10.1007/s42979-021-00786-3>.
- [14] Bela Bauer, et al., Quantum algorithms for quantum chemistry and quantum materials science, Chem. Rev. (ISSN: 0009-2665) 120 (2020) 12685–12717, <http://dx.doi.org/10.1021/acs.chemrev.9b00829>.
- [15] Martin P. Andersson, et al., Quantum computing for chemical and biomolecular product design, Curr. Opin. Chem. Eng. (ISSN: 22113398) 36 (2022) 100754, <http://dx.doi.org/10.1016/j.coche.2021.100754>.
- [16] Hongbin Liu, et al., Prospects of quantum computing for molecular sciences, Mater. Theory (ISSN: 2509-8012) 6 (2022) 11, <http://dx.doi.org/10.1186/s41313-021-00039-z>.
- [17] Yan Wang, Jungin E. Kim, Krishnan Suresh, Opportunities and challenges of quantum computing for engineering optimization, J. Comput. Inf. Sci. Eng. (ISSN: 1530-9827) 23 (2023) <http://dx.doi.org/10.1115/1.4062969>.
- [18] Yuri Alexeev, et al., Quantum-centric supercomputing for materials science: A perspective on challenges and future directions, 2023, URL <http://arxiv.org/abs/2312.09733>.
- [19] Eneko Osaba, Esther Villar-Rodriguez, Izaskun Oregi, A systematic literature review of quantum computing for routing problems, IEEE Access (ISSN: 2169-3536) 10 (2022) 55805–55817, <http://dx.doi.org/10.1109/ACCESS.2022.3177790>.
- [20] Naeimeh Mohseni, Peter L. McMahon, Tim Byrnes, Ising machines as hardware solvers of combinatorial optimization problems, Nat. Rev. Phys. (ISSN: 2522-5820) 4 (2022) 363–379, <http://dx.doi.org/10.1038/s42254-022-00440-8>.
- [21] Daniel J. Egger, et al., Quantum computing for finance: State-of-the-art and future prospects, IEEE Trans. Quantum Eng. (ISSN: 2689-1808) 1 (2020) 1–24, <http://dx.doi.org/10.1109/TQE.2020.3030314>.
- [22] Dylan Herman, et al., A survey of quantum computing for finance, 2022, URL <http://arxiv.org/abs/2201.02773>.
- [23] Daniel J. Egger, Jakub Mareček, Stefan Woerner, Warm-starting quantum optimization, Quantum (ISSN: 2521-327X) 5 (2021) 479, <http://dx.doi.org/10.22331/q-2021-06-17-479>.
- [24] Hossein Ghiasi, Damiano Pasini, Larry Lessard, Optimum stacking sequence design of composite materials part I: Constant stiffness design, Compos. Struct. (ISSN: 02638223) 90 (2009) 1–11, <http://dx.doi.org/10.1016/j.compstruct.2009.01.006>.

- [25] Hossein Ghiasi, et al., Optimum stacking sequence design of composite materials part II: Variable stiffness design, *Compos. Struct.* (ISSN: 02638223) 93 (2010) 1–13, <http://dx.doi.org/10.1016/j.compstruct.2010.06.001>.
- [26] S. Nikbakht, S. Kamarian, M. Shakeri, A review on optimization of composite structures part I: Laminated composites, *Compos. Struct.* (ISSN: 02638223) 195 (2018) 158–185, <http://dx.doi.org/10.1016/j.compstruct.2018.03.063>.
- [27] S. Nikbakht, S. Kamarian, M. Shakeri, A review on optimization of composite structures part II: Functionally graded materials, *Compos. Struct.* (ISSN: 02638223) 214 (2019) 83–102, <http://dx.doi.org/10.1016/j.compstruct.2019.01.105>.
- [28] IBM Research, How quantum computers could help design airplanes, 2023, URL <https://research.ibm.com/blog/boeing-case-study>.
- [29] Bryce Fuller, et al., Approximate solutions of combinatorial problems via quantum relaxations, 2021, URL <http://arxiv.org/abs/2111.03167>.
- [30] Koetsu Yamazaki, Two-level optimization technique of composite laminate panels by genetic algorithms, in: 37th Structure, Structural Dynamics and Materials Conference, vol. 37, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1996, pp. 1882–1887, <http://dx.doi.org/10.2514/6.1996-1539>.
- [31] J. Enrique Herencia, Paul M. Weaver, Michael I. Friswell, Optimization of long anisotropic laminated fiber composite panels with T-shaped stiffeners, *AIAA J.* (ISSN: 0001-1452) 45 (2007) 2497–2509, <http://dx.doi.org/10.2514/1.26321>.
- [32] Samuel T. Jusselmuiden, et al., Multi-step blended stacking sequence design of panel assemblies with buckling constraints, *Composites B* (ISSN: 13598368) 40 (2009) 329–336, <http://dx.doi.org/10.1016/j.compositesb.2008.12.002>.
- [33] Dianzi Liu, et al., Bilevel optimization of blended composite wing panels, *J. Aircr.* (ISSN: 0021-8669) 48 (2011) 107–118, <http://dx.doi.org/10.2514/1.C000261>.
- [34] Dianzi Liu, Vassili V. Toropov, A lamination parameter-based strategy for solving an integer-continuous problem arising in composite optimization, *Comput. Struct.* (ISSN: 00457949) 128 (2013) 170–174, <http://dx.doi.org/10.1016/j.compstruct.2013.06.003>.
- [35] Terence Macquart, Noud Werter, Roeland De Breuker, Aeroelastic design of blended composite structures using lamination parameters, *J. Aircr.* (ISSN: 0021-8669) 54 (2017) 561–571, <http://dx.doi.org/10.2514/1.C033859>.
- [36] Xiaoyang Liu, Carol A. Featherston, David Kennedy, Two-level layup optimization of composite laminate using lamination parameters, *Compos. Struct.* (ISSN: 02638223) 211 (2019) 337–350, <http://dx.doi.org/10.1016/j.compstruct.2018.12.054>.
- [37] S.W. Tsai, N.J. Pagano, Invariant Properties of Composite Materials, Defense Technical Information Center, Ft. Belvoir, 1968, URL <https://apps.dtic.mil/sti/citations/AD0668761>.
- [38] Mitsunori Miki, Yoshihiko Sugiyama, Optimum design of laminated composite plates using lamination parameters, in: 32nd Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1991, <http://dx.doi.org/10.2514/6.1991-971>.
- [39] Michael Chun-Yung Niu, *Airframe-Structural-Design*, Technical Book Co, ISBN: 962-7128-04-X, 1988.
- [40] J.A. Bailie, R.P. Ley, A. Pasricha, A Summary and Review of Composite Laminate Design Guidelines, NAS1-19347, Northrop Grumman-Military Aircraft Systems Division, 1997, URL <https://www.abbottaerospace.com/downloads/nasa-nas1-19347-summary-and-review-of-composite-laminate-design-guidelines/>.
- [41] Composite Materials Handbook (Volume 3) MIL-HDBK-17-3F, Department of Defense, USA, 2002, URL <https://www.abbottaerospace.com/downloads/mil-hdbk-17/>.
- [42] Mazen A. Albazzan, et al., Efficient design optimization of nonconventional laminated composites using lamination parameters: A state of the art, *Compos. Struct.* (ISSN: 02638223) 209 (2019) 362–374, <http://dx.doi.org/10.1016/j.compstruct.2018.10.095>.
- [43] Y. Narita, Layerwise optimization for the maximum fundamental frequency of laminated composite plates, *J. Sound Vib.* (ISSN: 0022460X) 263 (2003) 1005–1016, [http://dx.doi.org/10.1016/S0022-460X\(03\)00270-0](http://dx.doi.org/10.1016/S0022-460X(03)00270-0).
- [44] Julien van Campen, Zafer Gürdal, Retrieving variable stiffness laminates from lamination parameters distribution, in: 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, (ISSN: 02734508) ISBN: 978-1-60086-975-4, 2009, <http://dx.doi.org/10.2514/6.2009-2183>.
- [45] Akira Todoroki, Yuichiro Terada, Improved fractal branch and bound method for stacking-sequence optimizations of laminates, *AIAA J.* (ISSN: 0001-1452) 42 (2004) 141–148, <http://dx.doi.org/10.2514/1.9038>.
- [46] Yuichiro Terada, Akira Todoroki, Yoshinobu Shimamura, Stacking sequence optimizations using fractal branch and bound method for laminated composites, *JSME Int. J. Ser. A* (ISSN: 1344-7912) 44 (2001) 490–498, <http://dx.doi.org/10.1299/jsmea.44.490>.
- [47] Akira Todoroki, Masato Sekishiro, New iteration fractal branch and bound method for stacking sequence optimizations of multiple laminates, *Compos. Struct.* (ISSN: 02638223) 81 (2007) 419–426, <http://dx.doi.org/10.1016/j.compstruct.2006.09.001>.
- [48] Noémie Fedon, et al., A method using beam search to design the lay-ups of composite laminates with many plies, *Composites C Open Access* (ISSN: 26666820) 4 (2021) 100072, <http://dx.doi.org/10.1016/j.jcocom.2020.100072>.
- [49] Jung-Seok Kim, Chun-Gon Kim, Chang-Sun Hong, Optimum design of composite structures with ply drop using genetic algorithm and expert system shell, *Compos. Struct.* (ISSN: 02638223) 46 (1999) 171–187, [http://dx.doi.org/10.1016/S0263-8223\(99\)00052-5](http://dx.doi.org/10.1016/S0263-8223(99)00052-5).
- [50] Grant Soremekun, et al., Stacking sequence blending of multiple composite laminates using genetic algorithms, *Compos. Struct.* (ISSN: 02638223) 56 (2002) 53–62, [http://dx.doi.org/10.1016/S0263-8223\(01\)00185-4](http://dx.doi.org/10.1016/S0263-8223(01)00185-4).
- [51] Birna P. Kristinsdottir, et al., Optimal design of large composite panels with varying loads, *Compos. Struct.* (ISSN: 02638223) 51 (2001) 93–102, [http://dx.doi.org/10.1016/S0263-8223\(00\)00128-8](http://dx.doi.org/10.1016/S0263-8223(00)00128-8).
- [52] David B. Adams, et al., Genetic algorithm optimization and blending of composite laminates by locally reducing laminate thickness, *Adv. Eng. Softw.* (ISSN: 09659978) 35 (2004) 35–43, <http://dx.doi.org/10.1016/j.advengsoft.2003.09.001>.
- [53] Zelda B. Zabinsky, Mark E. Tuttle, Charoenchai Khompatraporn, A case study: Composite structure design optimization, Springer US, 2006, pp. 507–528, [http://dx.doi.org/10.1007/0-387-30927-6\\_21](http://dx.doi.org/10.1007/0-387-30927-6_21).
- [54] Julien van Campen, et al., General blending definitions for stacking sequence design of composite laminate structures, in: 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference; 16th AIAA/ASME/AHS Adaptive Structures Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, ISBN: 978-1-60086-993-8, 2008, <http://dx.doi.org/10.2514/6.2008-1798>.
- [55] Akira Todoroki, Raphael Haftka, Lamination parameters for efficient genetic optimization of the stacking sequences of composite panels, in: 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1998, pp. 870–879, <http://dx.doi.org/10.2514/6.1998-4816>.
- [56] M. Autio, Determining the real lay-up of a laminate corresponding to optimal lamination parameters by genetic search, *Struct. Multidiscip. Optim.* (ISSN: 1615-147X) 20 (2000) 301–310, <http://dx.doi.org/10.1007/s001580050160>.
- [57] Mark Bloomfield, J. Herencia, Paul Weaver, Optimisation of anisotropic composite plates incorporating non-conventional ply orientations, in: 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference; 16th AIAA/ASME/AHS Adaptive Structures Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, ISBN: 978-1-60086-993-8, 2008, <http://dx.doi.org/10.2514/6.2008-1918>.
- [58] Mark W. Bloomfield, J. Enrique Herencia, Paul M. Weaver, Enhanced two-level optimization of anisotropic laminated composite plates with strength and buckling constraints, *Thin-Walled Struct.* (ISSN: 02638231) 47 (2009) 1161–1167, <http://dx.doi.org/10.1016/j.tws.2009.04.008>.
- [59] Mark W. Bloomfield, J. Enrique Herencia, Paul M. Weaver, Analysis and benchmarking of meta-heuristic techniques for lay-up optimization, *Comput. Struct.* (ISSN: 00457949) 88 (2010) 272–282, <http://dx.doi.org/10.1016/j.compstruct.2009.10.007>.

- [60] F.-X. Irisarri, et al., Optimal design of laminated composite structures with ply drops using stacking sequence tables, *Compos. Struct.* (ISSN: 02638223) 107 (2014) 559–569, <http://dx.doi.org/10.1016/j.compstruct.2013.08.030>.
- [61] Yasser M. Meddaikar, François-Xavier Irisarri, Mostafa M. Abdalla, Laminate optimization of blended composite structures using a modified shepard's method and stacking sequence tables, *Struct. Multidiscip. Optim.* (ISSN: 1615-147X) 55 (2017) 535–546, <http://dx.doi.org/10.1007/s00158-016-1508-0>.
- [62] Yudong Cao, et al., Quantum chemistry in the age of quantum computing, *Chem. Rev.* (ISSN: 0009-2665) 119 (2019) 10856–10915, <http://dx.doi.org/10.1021/acs.chemrev.8b00803>.
- [63] Thomas Ayrar, et al., Quantum computing with and for many-body physics, *Eur. Phys. J. A* (ISSN: 1434-601X) 59 (2023) 227, <http://dx.doi.org/10.1140/epja/s10050-023-01141-1>.
- [64] Alberto Peruzzo, et al., A variational eigenvalue solver on a photonic quantum processor, *Nature Commun.* (ISSN: 2041-1723) 5 (2014) 4213, <http://dx.doi.org/10.1038/ncomms5213>.
- [65] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, A quantum approximate optimization algorithm, 2014, URL <http://arxiv.org/abs/1411.4028>.
- [66] M. Cerezo, et al., Variational quantum algorithms, *Nat. Rev. Phys.* (ISSN: 2522-5820) 3 (2021) 625–644, <http://dx.doi.org/10.1038/s42254-021-00348-9>.
- [67] Suguru Endo, et al., Hybrid quantum-classical algorithms and quantum error mitigation, *J. Phys. Soc. Japan* (ISSN: 0031-9015) 90 (2021) 032001, <http://dx.doi.org/10.7566/JPSJ.90.032001>.
- [68] Tadashi Kadowaki, Hidetoshi Nishimori, Quantum annealing in the transverse ising model, *Phys. Rev. E* (ISSN: 1063-651X) 58 (1998) 5355–5363, <http://dx.doi.org/10.1103/PhysRevE.58.5355>.
- [69] Tadashi Kadowaki, Study of optimization problems by quantum annealing, 2002, URL <http://arxiv.org/abs/quant-ph/0205020>.
- [70] Philipp Hauke, et al., Perspectives of quantum annealing: methods and implementations, *Rep. Progr. Phys.* (ISSN: 0034-4885) 83 (2020) 054401, <http://dx.doi.org/10.1088/1361-6633/ab85b8>.
- [71] A. Yu. Kitaev, Quantum measurements and the abelian stabilizer problem, 1995, URL <http://arxiv.org/abs/quant-ph/9511026>.
- [72] Daniel S. Abrams, Seth Lloyd, Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors, *Phys. Rev. Lett.* (ISSN: 0031-9007) 83 (1999) 5162–5165, <http://dx.doi.org/10.1103/PhysRevLett.83.5162>.
- [73] Leo Zhou, et al., Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, *Phys. Rev. X* (ISSN: 2160-3308) 10 (2020) 021067, <http://dx.doi.org/10.1103/PhysRevX.10.021067>.
- [74] Steven R. White, Density matrix formulation for quantum renormalization groups, *Phys. Rev. Lett.* (ISSN: 0031-9007) 69 (1992) 2863–2866, <http://dx.doi.org/10.1103/PhysRevLett.69.2863>.
- [75] Steven R. White, Density-matrix algorithms for quantum renormalization groups, *Phys. Rev. B* (ISSN: 0163-1829) 48 (1993) 10345–10356, <http://dx.doi.org/10.1103/PhysRevB.48.10345>.
- [76] Tomotoshi Nishino, Kouichi Okunishi, Corner transfer matrix renormalization group method, *J. Phys. Soc. Japan* (ISSN: 0031-9015) 65 (1996) 891–894, <http://dx.doi.org/10.1143/JPSJ.65.891>.
- [77] F. Verstraete, D. Porras, J.I. Cirac, Density matrix renormalization group and periodic boundary conditions: A quantum information perspective, *Phys. Rev. Lett.* (ISSN: 0031-9007) 93 (2004) 227205, <http://dx.doi.org/10.1103/PhysRevLett.93.227205>.
- [78] Ulrich Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Physics* (ISSN: 00034916) 326 (2011) 96–192, <http://dx.doi.org/10.1016/j.aop.2010.09.012>.
- [79] Garnet Kin-Lic Chan, et al., Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms, 2016, URL <http://arxiv.org/abs/1605.02611>.
- [80] Fiber Technology for Fiber-Reinforced Composites, Elsevier, ISBN: 9780081018712, 2017, <http://dx.doi.org/10.1016/C2015-0-05497-1>.
- [81] E. Reissner, Y. Stavsky, Bending and stretching of certain types of heterogeneous aeolotropic elastic plates, *J. Appl. Mech.* (ISSN: 0021-8936) 28 (1961) 402–408, <http://dx.doi.org/10.1115/1.3641719>.
- [82] S.B. Dong, K.S. Pister, R.L. Taylor, On the theory of laminated anisotropic shells and plates, *J. Aerosp. Sci.* (ISSN: 1936-9999) 29 (1962) 969–975, <http://dx.doi.org/10.2514/8.9668>.
- [83] Stephen W. Tsai, H. Thomas Hahn, Introduction to Composite Materials, Routledge, ISBN: 9780203750148, 1980, <http://dx.doi.org/10.1201/9780203750148>.
- [84] Samuel T. IJsselmuiden, Optimal design of variable stiffness composite structures using lamination parameters, 2011, URL <http://resolver.tudelft.nl/uuid:973a564b-5734-42c4-a67c-1044f1e25f1c>.
- [85] Moritz Sprengholz, et al., Rapid transformation of lamination parameters into stacking sequences, *Compos. Struct.* (ISSN: 02638223) 276 (2021) 114514, <http://dx.doi.org/10.1016/j.compstruct.2021.114514>.
- [86] Michele Mosca, Quantum algorithms, 2008, URL <http://arxiv.org/abs/0808.0369>.
- [87] Ashley Montanaro, Quantum algorithms: an overview, *npj Quantum Inf.* (ISSN: 2056-6387) 2 (2016) 15023, <http://dx.doi.org/10.1038/npjqi.2015.23>.
- [88] Kishor Bharti, et al., Noisy intermediate-scale quantum algorithms, *Rev. Modern Phys.* (ISSN: 0034-6861) 94 (2022) 015004, <http://dx.doi.org/10.1103/RevModPhys.94.015004>.
- [89] G. Ntourmas, et al., Mixed integer linear programming formulations of the stacking sequence and blending optimisation of composite structures, *Compos. Struct.* (ISSN: 02638223) 264 (2021) 113660, <http://dx.doi.org/10.1016/j.compstruct.2021.113660>.
- [90] Hartmut Neven, Georgie Rose, William G. Macready, Image recognition with an adiabatic quantum computer I. Mapping to quadratic unconstrained binary optimization, 2008, URL <http://arxiv.org/abs/0804.4457>.
- [91] Zhengbing Bian, et al., The Ising Model: Teaching an Old Problem New Tricks, D-Wave, 2010, URL <https://www.dwavesys.com/resources/publication/the-ising-model-teaching-an-old-problem-new-tricks/>.
- [92] Eleanor G. Rieffel, et al., A case study in programming a quantum annealer for hard operational planning problems, *Quantum Inf. Process.* (ISSN: 1570-0755) 14 (2015) 1–36, <http://dx.doi.org/10.1007/s11128-014-0892-x>.
- [93] Andrew Lucas, Ising formulations of many NP problems, *Front. Phys.* (ISSN: 2296-424X) 2 (2014) 1–14, <http://dx.doi.org/10.3389/fphy.2014.00005>.
- [94] Itay Hen, Federico M. Spedalieri, Quantum annealing for constrained optimization, *Phys. Rev. Appl.* (ISSN: 2331-7019) 5 (2016) 034007, <http://dx.doi.org/10.1103/PhysRevApplied.5.034007>.
- [95] Fred Glover, et al., Quantum bridge analytics I: a tutorial on formulating and using QUBO models, *Ann. Oper. Res.* (ISSN: 0254-5330) 314 (2022) 141–183, <http://dx.doi.org/10.1007/s10479-022-04634-2>.
- [96] John Parkinson, Damian J.J. Farnell, An Introduction to Quantum Spin Systems, vol. 816, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-13289-6, 2010, <http://dx.doi.org/10.1007/978-3-642-13290-2>.
- [97] John B. Kogut, An introduction to lattice gauge theory and spin systems, *Rev. Modern Phys.* (ISSN: 0034-6861) 51 (1979) 659–713, <http://dx.doi.org/10.1103/RevModPhys.51.659>.
- [98] Daniel S. Abrams, Seth Lloyd, Simulation of many-body Fermi systems on a universal quantum computer, *Phys. Rev. Lett.* (ISSN: 0031-9007) 79 (1997) 2586–2589, <http://dx.doi.org/10.1103/PhysRevLett.79.2586>.
- [99] David Poulin, Pawel Wocjan, Preparing ground states of quantum many-body systems on a quantum computer, *Phys. Rev. Lett.* (ISSN: 0031-9007) 102 (2009) 130503, <http://dx.doi.org/10.1103/PhysRevLett.102.130503>.
- [100] J. Eisert, M. Cramer, M.B. Plenio, Colloquium : Area laws for the entanglement entropy, *Rev. Modern Phys.* (ISSN: 0034-6861) 82 (2010) 277–306, <http://dx.doi.org/10.1103/RevModPhys.82.277>.

- [101] Alba Cervera-Lierta, Exact Ising model simulation on a quantum computer, *Quantum* (ISSN: 2521-327X) 2 (2018) 114, <http://dx.doi.org/10.22331/q-2018-12-21-114>.
- [102] Guido Pagano, et al., Quantum approximate optimization of the long-range ising model with a trapped-ion quantum simulator, *Proc. Natl. Acad. Sci.* (ISSN: 0027-8424) 117 (2020) 25396–25401, <http://dx.doi.org/10.1073/pnas.2006373117>.
- [103] I.G. Rosenberg, Reduction of bivalent maximization to the quadratic case, *Cahiers Centre d'Études Recherche Opérationnelle* 17 (1975) 71–74.
- [104] Martin Anthony, et al., Quadratic reformulations of nonlinear binary optimization problems, *Math. Program.* (ISSN: 0025-5610) 162 (2017) 115–144, <http://dx.doi.org/10.1007/s10107-016-1032-4>.
- [105] Nike Dattani, Quadraticization in discrete optimization and quantum mechanics, 2019, URL <http://arxiv.org/abs/1901.04405>.
- [106] Endre Boros, Yves Crama, Elisabeth Rodríguez-Heck, Compact quadraticizations for pseudo-Boolean functions, *J. Comb. Optim.* (ISSN: 1382-6905) 39 (2020) 687–707, <http://dx.doi.org/10.1007/s10878-019-00511-0>.
- [107] J. Dukelsky, et al., Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains, *Europhys. Lett.* (EPL) (ISSN: 0295-5075) 43 (1998) 457–462, <http://dx.doi.org/10.1209/epl/i1998-00381-x>.
- [108] Stuart Hadfield, et al., From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, *Algorithms* (ISSN: 1999-4893) 12 (2019) 34, <http://dx.doi.org/10.3390/a12020034>.
- [109] Jarrod R. McClean, et al., The theory of variational hybrid quantum-classical algorithms, *New J. Phys.* (ISSN: 1367-2630) 18 (2016) 023023, <http://dx.doi.org/10.1088/1367-2630/18/2/023023>.
- [110] Joonho Lee, et al., Generalized unitary coupled cluster wave functions for quantum computation, *J. Chem. Theory Comput.* (ISSN: 1549-9618) 15 (2019) 311–324, <http://dx.doi.org/10.1021/acs.jctc.8b01004>.
- [111] Abhinav Kandala, et al., Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, *Nature* (ISSN: 0028-0836) 549 (2017) 242–246, <http://dx.doi.org/10.1038/nature23879>.
- [112] Quantum optimization using variational algorithms on near-term quantum devices, *Quantum Sci. Technol.* (ISSN: 2058-9565) 3 (2018) 030503, <http://dx.doi.org/10.1088/2058-9565/aab822>.
- [113] Harper R. Grimsley, et al., An adaptive variational algorithm for exact molecular simulations on a quantum computer, *Nature Commun.* (ISSN: 2041-1723) 10 (2019) 3007, <http://dx.doi.org/10.1038/s41467-019-10988-2>.
- [114] Linghua Zhu, et al., Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer, *Phys. Rev. Res.* (ISSN: 2643-1564) 4 (2022) 033029, <http://dx.doi.org/10.1103/PhysRevResearch.4.033029>.
- [115] Sergey Bravyi, et al., Classical algorithms for forrelation, 2021, URL <http://arxiv.org/abs/2102.06963>.
- [116] David Amaro, et al., Filtering variational quantum algorithms for combinatorial optimization, *Quantum Sci. Technol.* (ISSN: 2058-9565) 7 (2022) 015021, <http://dx.doi.org/10.1088/2058-9565/ac3e54>.
- [117] Yash J. Patel, et al., Reinforcement learning assisted recursive QAOA, *EPJ Quantum Technol.* (ISSN: 2662-4400) 11 (2024) 6, <http://dx.doi.org/10.1140/epjqt/s40507-023-00214-w>.
- [118] Artur Garcia-Saez, Jordi Riu, Quantum observables for continuous control of the quantum approximate optimization algorithm via reinforcement learning, 2019, URL <http://arxiv.org/abs/1911.09682>.
- [119] Lennart Bittel, Martin Kliesch, Training variational quantum algorithms is NP-hard, *Phys. Rev. Lett.* (ISSN: 0031-9007) 127 (2021) 120502, <http://dx.doi.org/10.1103/PhysRevLett.127.120502>.
- [120] Chufan Lyu, Victor Montenegro, Abolfazl Bayat, Accelerated variational algorithms for digital quantum simulation of many-body ground states, *Quantum* (ISSN: 2521-327X) 4 (2020) 324, <http://dx.doi.org/10.22331/q-2020-09-16-324>.
- [121] Andrea Skolik, et al., Layerwise learning for quantum neural networks, *Quantum Mach. Intell.* (ISSN: 2524-4906) 3 (2021) 5, <http://dx.doi.org/10.1007/s42484-020-00036-4>.
- [122] Xiaoyuan Liu, et al., Layer VQE: A variational approach for combinatorial optimization on noisy quantum computers, *IEEE Trans. Quantum Eng.* (ISSN: 2689-1808) 3 (2022) 1–20, <http://dx.doi.org/10.1109/TQE.2021.3140190>.
- [123] Edward Farhi, Aram W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, 2016, URL <http://arxiv.org/abs/1602.07674>.
- [124] Román Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, *Ann. Physics* (ISSN: 00034916) 349 (2014) 117–158, <http://dx.doi.org/10.1016/j.aop.2014.06.013>.
- [125] Román Orús, Tensor networks for complex quantum systems, *Nat. Rev. Phys.* (ISSN: 2522-5820) 1 (2019) 538–550, <http://dx.doi.org/10.1038/s42254-019-0086-7>.
- [126] Jacob C. Bridgeman, Christopher T. Chubb, Hand-waving and interpretive dance: an introductory course on tensor networks, *J. Phys. A* (ISSN: 1751-8113) 50 (2017) 223001, <http://dx.doi.org/10.1088/1751-8121/aa6dc3>.
- [127] Jacob Biamonte, Ville Bergholm, Tensor networks in a nutshell, 2017, URL <http://arxiv.org/abs/1708.00006>.
- [128] J. Ignacio Cirac, et al., Matrix product states and projected entangled pair states: Concepts, symmetries, theorems, *Rev. Modern Phys.* (ISSN: 0034-6861) 93 (2021) 045003, <http://dx.doi.org/10.1103/RevModPhys.93.045003>.
- [129] Glen Evenbly, A practical guide to the numerical implementation of tensor networks I: Contractions, decompositions and gauge freedom, 2022, URL <http://arxiv.org/abs/2202.02138>.
- [130] I.V. Oseledets, E.E. Tyrtshnikov, Breaking the curse of dimensionality, or how to use SVD in many dimensions, *SIAM J. Sci. Comput.* 31 (5) (2009) 3744–3759, <http://dx.doi.org/10.1137/090748330>.
- [131] I.V. Oseledets, Tensor-train decomposition, *SIAM J. Sci. Comput.* 33 (5) (2011) 2295–2317, <http://dx.doi.org/10.1137/090752286>.
- [132] Frank Arute, et al., Quantum supremacy using a programmable superconducting processor, *Nature* (ISSN: 0028-0836) 574 (2019) 505–510, <http://dx.doi.org/10.1038/s41586-019-1666-5>.
- [133] Johnnie Gray, Stefanos Kourtis, Hyper-optimized tensor network contraction, *Quantum* (ISSN: 2521-327X) 5 (2021) 410, <http://dx.doi.org/10.22331/q-2021-03-15-410>.
- [134] Feng Pan, Pan Zhang, Simulating the sycamore quantum supremacy circuits, 2021, URL <http://arxiv.org/abs/2103.03074>.
- [135] Joseph Tindall, et al., Efficient tensor network simulation of IBM's eagle kicked ising experiment, *PRX Quantum* (ISSN: 2691-3399) 5 (2024) 010308, <http://dx.doi.org/10.1103/PRXQuantum.5.010308>.
- [136] Tomislav Begušić, Johnnie Gray, Garnet Kin-Lic Chan, Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance, *Sci. Adv.* (ISSN: 2375-2548) 10 (2024) 4321, <http://dx.doi.org/10.1126/sciadv.adk4321>.
- [137] Siddhartha Patra, et al., Efficient tensor network simulation of IBM's largest quantum processors, *Phys. Rev. Res.* (ISSN: 2643-1564) 6 (2024) 013326, <http://dx.doi.org/10.1103/PhysRevResearch.6.013326>.
- [138] Daniel E. Parker, Xiangyu Cao, Michael P. Zaletel, Local matrix product operators: Canonical form, compression, and control theory, *Phys. Rev. B* (ISSN: 2469-9950) 102 (2020) 035147, <http://dx.doi.org/10.1103/PhysRevB.102.035147>.
- [139] Ian P. McCulloch, From density-matrix renormalization group to matrix product states, *J. Stat. Mech. Theory Exp.* (ISSN: 1742-5468) 2007 (2007) P10014, <http://dx.doi.org/10.1088/1742-5468/2007/10/P10014>.
- [140] Manuel S. Rudolph, et al., Decomposition of matrix product states into shallow quantum circuits, 2022, URL <http://arxiv.org/abs/2209.00595>.
- [141] Manuel S. Rudolph, et al., Synergy between quantum circuits and tensor networks: Short-cutting the race to practical quantum advantage, 2022, URL <http://arxiv.org/abs/2208.13673>.



- [142] Shi-Ju Ran, Encoding of matrix product states into quantum circuits of one- and two-qubit gates, *Phys. Rev. A* (ISSN: 2469-9926) 101 (2020) 032310, <http://dx.doi.org/10.1103/PhysRevA.101.032310>.
- [143] Daniel Malz, et al., Preparation of matrix product states with log-depth quantum circuits, *Phys. Rev. Lett.* (ISSN: 0031-9007) 132 (2024) 040404, <http://dx.doi.org/10.1103/PhysRevLett.132.040404>.
- [144] Arne Wulff, et al., Research code accompanying the publication: "Quantum computing and tensor networks for laminate design: A novel approach to stacking sequence retrieval", 2024, GitHub. URL <https://github.com/ArneWulff/ssr-with-qc-and-tn>.
- [145] Arne Wulff, et al., Research data accompanying the publication: "Quantum computing and tensor networks for laminate design: A novel approach to stacking sequence retrieval", 2024, <http://dx.doi.org/10.4121/ae276609-55b0-4af1-88c0-1102b1b58990>, 4TU.ResearchData.
- [146] Ali Javadi-Abhari, et al., Quantum computing with qiskit, 2024, <http://dx.doi.org/10.48550/arXiv.2405.08810>, [arXiv:2405.08810](https://arxiv.org/abs/2405.08810) [quant-ph].
- [147] Pauli Virtanen, et al., SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nature Methods* 17 (2020) 261–272, <http://dx.doi.org/10.1038/s41592-019-0686-2>.
- [148] Matthew Fishman, Steven R. White, E. Miles Stoudenmire, The ITensor Software Library for Tensor Network Calculations, *SciPost Phys. Codebases* (2022) 4, <http://dx.doi.org/10.21468/SciPostPhysCodeb.4>.
- [149] Matthew Fishman, Steven R. White, E. Miles Stoudenmire, Codebase release 0.3 for ITensor, *SciPost Phys. Codebases* (2022) 4–r0.3, <http://dx.doi.org/10.21468/SciPostPhysCodeb.4-r0.3>.
- [150] Murray Rosenblatt, Remarks on some nonparametric estimates of a density function, *Ann. Math. Stat.* (ISSN: 0003-4851) 27 (1956) 832–837, <http://dx.doi.org/10.1214/aoms/1177728190>.
- [151] Emanuel Parzen, On estimation of a probability density function and mode, *Ann. Math. Stat.* (ISSN: 0003-4851) 33 (1962) 1065–1076, <http://dx.doi.org/10.1214/aoms/1177704472>.
- [152] Stanisław Węglarczyk, Kernel density estimation and its application, in: W. Zieliński, et al. (Eds.), in: *ITM Web of Conferences*, vol. 23, EDP Sciences, (ISSN: 2271-2097) 2018, p. 00037, <http://dx.doi.org/10.1051/itmconf/20182300037>.
- [153] Jarrod R. McClean, et al., Barren plateaus in quantum neural network training landscapes, *Nature Commun.* (ISSN: 2041-1723) 9 (2018) 4812, <http://dx.doi.org/10.1038/s41467-018-07090-4>.
- [154] Edward Grant, et al., An initialization strategy for addressing barren plateaus in parametrized quantum circuits, *Quantum* (ISSN: 2521-327X) 3 (2019) 214, <http://dx.doi.org/10.22331/q-2019-12-09-214>.
- [155] Guillaume Verdon, et al., Learning to learn with quantum neural networks via classical neural networks, 2019, URL <http://arxiv.org/abs/1907.05415>.
- [156] Sami Khairy, et al., Learning to optimize variational quantum circuits to solve combinatorial problems, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, (ISSN: 2374-3468) 2020, pp. 2367–2375, <http://dx.doi.org/10.1609/aaai.v34i03.5616>.
- [157] John Preskill, Quantum computing in the NISQ era and beyond, *Quantum* (ISSN: 2521-327X) 2 (2018) 79, <http://dx.doi.org/10.22331/q-2018-08-06-79>.
- [158] Felix Truger, et al., Warm-starting and quantum computing: A systematic mapping study, 2023, URL <http://arxiv.org/abs/2303.06133>.
- [159] Peter C. Fishburn, Letter to the editor—Additive utilities with incomplete product sets: Application to priorities and assignments, *Oper. Res.* (ISSN: 0030-364X) 15 (1967) 537–542, <http://dx.doi.org/10.1287/opre.15.3.537>.
- [160] C. West Churchman, Russell L. Ackoff, An approximate measure of value, *J. Oper. Res. Soc. Am.* (ISSN: 0096-3984) 2 (1954) 172–187, <http://dx.doi.org/10.1287/opre.2.2.172>.
- [161] José Figueira, Salvatore Greco, Matthias Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys*, vol. 78, Springer New York, New York, NY, ISBN: 978-0-387-23067-2, 2005, <http://dx.doi.org/10.1007/b100605>.