Quantum Science and Technology

PAPER

Quantum state preparation using tensor networks

Ar A Melnikov[1], A A Termanova[1], S V Dolgov[2], F Neukart[1,3] and M R Perelshtein[1,4,*]

[1] Terra Quantum AG, Kornhausstrasse 25, 9000 St. Gallen, Switzerland
[2] University of Bath, Claverton Down, Bath BA2 7AY, United Kingdom
[3] LIACS, Leiden University Leiden, Leiden, The Netherlands
[4] QTF Centre of Excellence, Department of Applied Physics, Aalto University School of Science, PO Box 15100, FI-00076 AALTO, Finland
* Author to whom any correspondence should be addressed.

E-mail: mpe@terraquantum.swiss

Keywords: tensor networks, quantum computing, variational circuits, quantum state preparation, Riemannian optimization

Supplementary material for this article is available online

OPEN ACCESS

RECEIVED
5 December 2022

REVISED
16 May 2023

ACCEPTED FOR PUBLICATION
30 May 2023

PUBLISHED
19 June 2023

Abstract

Quantum state preparation is a vital routine in many quantum algorithms, including solution of linear systems of equations, Monte Carlo simulations, quantum sampling, and machine learning. However, to date, there is no established framework of encoding classical data into gate-based quantum devices. In this work, we propose a method for the encoding of vectors obtained by sampling analytical functions into quantum circuits that features polynomial runtime with respect to the number of qubits and provides $>99.9\%$ accuracy, which is better than a state-of-the-art two-qubit gate fidelity. We employ hardware-efficient variational quantum circuits, which are simulated using tensor networks, and matrix product state representation of vectors. In order to tune variational gates, we utilize Riemannian optimization incorporating auto-gradient calculation. Besides, we propose a 'cut once, measure twice' method, which allows us to avoid barren plateaus during gates' update, benchmarking it up to 100-qubit circuits. Remarkably, any vectors that feature low-rank structure—not limited by analytical functions—can be encoded using the presented approach. Our method can be easily implemented on modern quantum hardware, and facilitates the use of the hybrid-quantum computing architectures.

1. Introduction

Quantum state preparation is a key subroutine in the overwhelming majority of quantum algorithms. Such algorithms as quantum linear system solvers [1–5], quantum Monte-Carlo [6, 7], quantum machine learning [8–10], and general variational quantum approaches [11] require fast and accurate encoding of classical data into quantum circuits. The state preparation subroutine is often considered the main bottleneck limiting the performance of quantum algorithms [9, 12, 13].

In order to prepare a generic n-qubit quantum state, a circuit with depth $O(2^n)$ is required since all 2^n inherent parameters corresponding to amplitudes must be encoded using different gates [14]. However, the depth can be reduced to $O(n^2)$ by using $O(2^n)$ ancillary qubits [15]. Those approaches do not solve the problem of exponential circuit growth with increasing the number of coding qubits and are impractical for large n, especially in case of noisy intermediate-scale quantum (NISQ) devices with finite fidelity and limited connectivity [16]. On the other hand, one can slightly decrease the circuit depth in exchange for an approximation error [17] using, for instance, the entanglement reduction technique [18]—specific classes of quantum states, e.g. normal distributions, can be prepared utilizing relatively shallow circuits [19], however, such methods still suffer from an exponential circuit width and a complex gate realization.

General engineering of quantum circuits specifically for the generation of particular states is challenging and impractical. On the other hand, variational quantum circuits with fixed ansatz, which serves as a general state engineering tool [11], provide a promising approach to state preparation. The general class of quantum states that can be efficiently prepared via variational circuits using classical computations is Hierarchical

tensor formats [20]—only such states allow for the contraction with a variational circuit in linear time with respect to the number of qubits [21–23]. Here, we focus on one of the most well-studied and straightforward tensor formats, the matrix product states (MPSs) [21, 22, 24], which is at the core of a variety of tensor networks algorithms [25–27]. In contrast to existing MPS-based algorithms for quantum state preparation, our work contributes three novel aspects:

- We mainly focus on the encoding of vectors obtained by sampling analytical functions, since they are of the most interest for quantum practical applications, including partial differential equations [28, 29], two-electron integral calculation [30], sampling tasks [31], calculation of expectation values [6], which is required e.g. in finance [12];
- We introduce and utilize the Riemannian optimization for the update of quantum gates, which allows us to avoid spurious maxima of the fidelity and achieve higher accuracy and speed;
- We develop a scheme, which is resistant to barren plateaus [32], and numerically verify the robustness up to 100 qubits using `QMware` system (see appendix A).

1.1. Comparison with previous works

The matrix product state formalism, which emerged from the description of many-body quantum systems [21, 24], can be utilized for efficient quantum state preparation [33–35]. Although such methods are accurate, they require unitary operations acting immediately on the $(\lfloor \log(r) \rfloor + 1)$ qubits, where r is a rank (bond dimension) of the corresponding MPS [24], which is undesirable for modern quantum computers considering the limited connectivity and high gate errors. To reduce the hardware requirements, a matrix product disentangler algorithm was developed in [36], which prepares the state approximately and requires the implementation only of two-qubit gates acting on neighbor qubits. The disadvantage of this algorithm is that updating the gate parameters occurs sequentially layer-by-layer, and not on all gates simultaneously, which causes a higher approximation error [37]. Also, the quantum circuit is fixed and cannot be tuned accordingly to the given quantum hardware topology.

To facilitate the change in the architecture of the quantum circuit and simultaneously optimize all the gates in it, a projected gradient descent algorithm based on the automatic differentiation [38] was introduced in [37]. Firstly, the approximation error is minimized over unconstrained latent gates in a general matrix space, which are automatically differentiable. Secondly, the polar decomposition is applied to each latent gate independently to turn it into the actual (unitary) gate. However, the latter step is not variational, i.e. it does not take into account the gradient direction, and may produce a suboptimal solution.

In this work, we use a more natural update procedure for variational circuits aimed at MPS preparation based on the Riemannian optimization [39], where the gates are updated within the manifold of unitary matrices (see appendix B). This approach was shown to provide a higher speed and accuracy in some regimes [40] (see detailed comparison in supplemental material). Similarly to [41], we prepare vectors obtained by sampling analytical and smooth functions, However, in contrast to [41], we prepare states featuring an arbitrary rank allowing for the encoding of a wider class of functions. We approximate quantum states utilizing variational circuits with the native quantum gate set and [42] artificially restricting the connectivity between qubits to adjust the algorithm for NISQ hardware. We demonstrate that despite the limited connectivity high approximation accuracy is achieved by shallow circuits. Moreover, our method features the logarithmic complexity in the vector size and, in combination with the simple circuit structure, can be easily implemented on modern quantum hardware, to be applied to a variety of problems (see appendix C).

2. Results

2.1. The protocol idea—reducing everything to tensors

The matrix product state is a compressed representation of a vector that was originally introduced to approximate the ground state of a many-body Hamiltonian in quantum theory. The main characteristic of such a representation is the rank (bond dimension) r, which expresses the correlations and entanglement of the state vector. In fact, using singular value decomposition (SVD) [43] or cross-approximation [44], any 2^n vector can be represented as MPS with the number of elements $2nr^2$. In the worst scenario, the rank r depends exponentially on n. However, considering vectors arising from sampling of analytical functions on an discretized interval, often a small and n-independent rank is sufficient to approximate the function with high accuracy [41]. Moreover, there is an exact MPS-decomposition for trigonometric, exponential, polynomial and rational functions [45]. That is why the idea outlined in this work is not to consider the preparation of arbitrary states, but vectors that can be represented as MPS with a small rank. It is important to note that the set of such states, although it may seem small, nevertheless covers a wide range of the applied problems (see Appendix C).

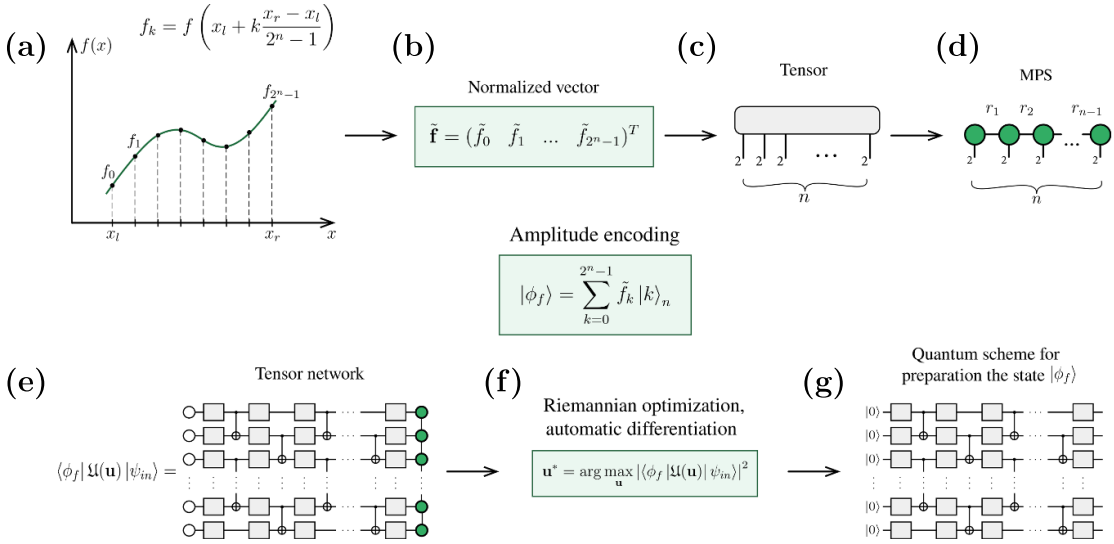

Figure 1. Overall quantum state preparation scheme. (a) The function to be prepared is sampled in 2^n points on the interval $[x_l, x_r]$—a vector of function values is obtained. (b) This vector is normalized and (c) reshaped into a tensor. (d) This tensor is converted into an MPS with visible index dimension equal to 2, and ranks (bond dimensions) equal to r_i [21]. The rank of the MPS is $r = \max(r_i)$. (e) A variational n-qubits quantum circuit with neighborhood interactions is represented as a tensor network and then contracted with the MPS to obtain the value of projection of the output quantum state onto the target one (1). (f) The optimal gates of the variational circuit are found using Riemannian optimization and automatic differentiation, yielding (g) the output state as close to the target one as possible.

The key quality of the MPS is weak entanglement. Indeed, considering the entropy of the reduced subsystem $\rho = Tr_2[|\psi\rangle\langle\psi|]$ of an MPS $|\psi\rangle$ with rank r, the following bound on entropy $S(\rho)$ holds: $S(\rho) \leqslant \log r$ [21]. Since such states feature a small rank, hence weak entanglement, we expect that they can be efficiently approximated by a circuit where only neighbor qubits are coupled and the depth is restricted. Such circuits possess limited entanglement and are similar to circuits specifically generating states belonging to the MPS class [34, 42].

In order to perform the MPS approximation using a quantum circuit, we represent the quantum circuit also as a tensor network allowing us to optimize the gate parameters by processing tensors directly. Leveraging the locality of operations in hardware-efficient circuits, we apply the tensor networks method for circuit simulation and optimization [46, 47]. Such a method allows us to process the circuit with polynomial runtime scaling in the number of qubits.

We emphasize that the learning of the circuit parameters for the state preparation does not need to be run on quantum hardware—the parameters can be learned using tensor networks on a classical computer, which are subsequently embedded in a circuit executed on a quantum processor.

2.2. Encoding and learning

Here, we provide a step-by-step description of the protocol, which is depicted in figure 1. In order to prepare a quantum state $|\phi_f\rangle$ we utilize a variational circuit, which is represented as an action of a variational unitary $\mathfrak{U}(\mathbf{a})$ over a state $|\psi_{\text{in}}\rangle = |0\rangle_n$. The problem of quantum state preparation is finding a set of parameters $\mathbf{a}$ such that the output state $\mathfrak{U}(\mathbf{a})|\psi_{\text{in}}\rangle$ approximates the desired $|\phi_f\rangle$:

$$\max_{\mathbf{a}} |\langle\phi_f|\mathfrak{U}(\mathbf{a})|\psi_{\text{in}}\rangle|^2. \tag{1}$$

2.2.1. Approximation of a function by an MPS: steps (a) to (d)

The desired state $|\phi_f\rangle$ is produced from an analytical function $f(x)$, which is assumed to have a small MPS rank. One way to encode a function into a quantum state is *the amplitude encoding*. The function is sampled on a given interval $[x_l,\ x_r]$ at 2^n equispaced points $x_0, \ldots, x_{2^n-1}$. Then a normalized vector of function values at these points is encoded in the quantum state amplitudes:

$$|\phi_f\rangle = \sum_{k=0}^{2^n-1} \tilde{f}_k |k\rangle_n, \quad \tilde{f}_k = \frac{f(x_k)}{Z}, \quad Z = \sum_{k=0}^{2^n-1} |f(x_k)|^2.$$

Further, we represent the obtained normalized vector $|\phi_f\rangle$ in the MPS format. For this, the vector is reshaped into a tensor of an order n and then sequential SVD applications allow us to represent it as an MPS

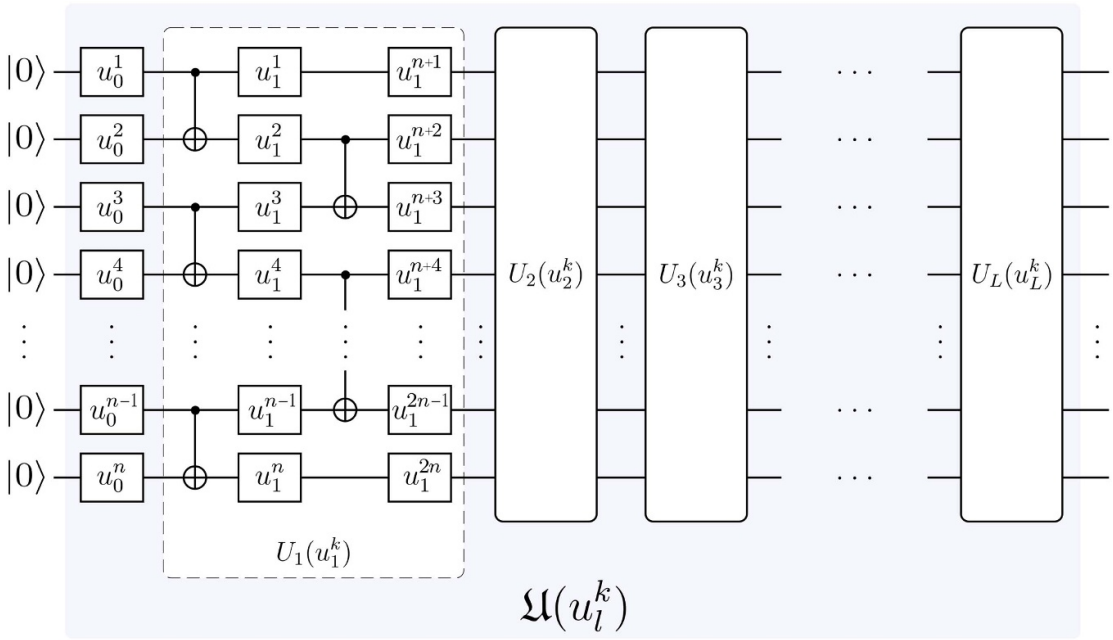

Figure 2. The variational quantum circuit used in all presented experiments. The unitary $\mathfrak{U}(u_l^k)$ is expressed as a product of L unitaries $U_l(u_l^k)$ sequentially acting on an input state. Each unitary U_l is a sequence of non-parametric CNOT gates and variational gates u_l^k.

[21, 48, 49]. However, the disadvantage of this approach is that it requires 2^n function calls, which is impractical in case of large vectors. Alternatively, the tensor-train cross approximation technique [44] allows one to recover an MPS by an adaptive sampling of just $O(nr^2)$ points. In this work, we utilize the cross approximation algorithm.

2.2.2. Processing a circuit as a tensor network: step (e)

In order to prepare states using NISQ devices we consider a unitary $\mathfrak{U}(\mathbf{a})$ consisting of sequentially applied single-qubit variational gates u_l^k mixed with CNOTs, as shown in figure 2.

As we mentioned in the previous section, the quantum circuit itself should be represented as a tensor network, with gates acting as tensors. The projection of the desired state on the state generated by this circuit $\langle\phi_f|\mathfrak{U}(\mathbf{a})|\psi_{\text{in}}\rangle$ is equivalent to the contraction of two tensor networks: $|\phi_f\rangle$ as an MPS and utilized circuit as a tensor network. Weak entanglement in shallow circuits with fixed depth, considered here, leads to the linear scaling of the simulation runtime with respect to the number of qubits. To achieve this, we utilize hyper-optimized tensor network contraction introduced in [46] that allows us to find a near-optimal contraction sequence. We show the numerical runtime of the projection of a random MPS $|\phi_f\rangle$ with rank 2 onto a quantum circuit with different number of layers up to 300 qubits in figure 3(a).

2.2.3. Optimization of the quantum circuit: step (f)

One of the most commonly used ways to solve the optimization problem equation (1) is to use the gradient-based parameter-shift rule [50]. Usually in that case, variational gates u_l^k depend on the parameters θ_l^k and have the form of $\exp(-i\theta_l^k\sigma)$, with σ being a Pauli operator. However, such a direct optimization of θ_l^k may suffer from a barren plateau of vanishing gradients due to the nonlinearity of the exponential function.

A more general initial circuit can be constructed from arbitrary unitary operators as variational gates. For this purpose, the Riemannian optimization on the Stiefel manifold of $m_1 \times m_2$ isometric matrices V_{m_1,m_2} can be used [51], where $m_1 = 2,\ m_2 = 2$ in case of single-qubit gates.

In order to compare those methods, we optimize a 5-qubit circuit with 2 layers of the said structure for preparing a random MPS with rank 2 and show how the approximation error decreases with the number of iterations of gradient descent in figure 3(b). The shift rule achieves only an approximation error worse than 10^{-2}, while the Riemannian optimization provides an error below 10^{-4}. Since the Riemannian optimization provides a better accuracy compared to the parameter-shift rule, we proceed with using the former.

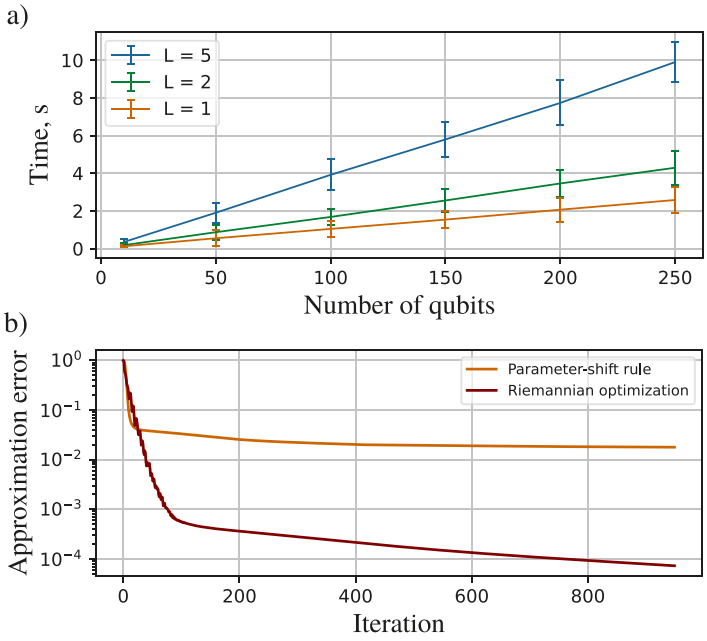

Figure 3. (a) Runtime of a single objective function calculation for varying numbers of qubits for 1-, 2-, and 5-layer circuit. The objective function equation (1) is computed as a tensor network contraction of a quantum circuit with the given number of layers and qubits and an MPS with rank 2, both populated with random parameters. Times are averaged over 100 runs of objective function calculation. (b) The approximation error obtained during an objective function minimization using the parameter-shift rule and the Riemannian optimization. Comparison between mentioned methods is done for a circuit containing 5 qubits and 2 layers approximating a random MPS with rank 2 sampled at 32 points. Since the Riemannian optimization provides a significantly lower error we focus on this method hereinafter.

For convenience, let us reformulate the fidelity maximization problem equation (1) into a minimization problem of infidelity (or approximation error)

$$C(u_l^k) = 1 - |\langle\phi|\mathfrak{U}(u_l^k)|\psi_{\text{in}}\rangle|^2 \to \min, \tag{2}$$

for a unitary 2×2 matrix u_l^k for each $l = 1 \ldots L$ and $k = 1 \ldots 2n$. In order to find a local minimum of some differentiable function $C(\mathbf{u})$, where $\mathbf{u}$ is $m_1 \times m_2$ isometric matrix, the gradient descent algorithm can be applied

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \alpha \nabla C(\mathbf{u}_t), \tag{3}$$

where $\mathbf{u}_t$ is a matrix $\mathbf{u}$ on step t, and α is a step size. Here, we consider the case of single unitary matrix dependence, however this analysis is valid in case of Cartesian product of such matricies [51].

The optimization problem in equation (2) must be constrained to $\mathbf{u} \in V_{m_1,m_2}$. This prevents us from using a conventional gradient descent algorithm—we cannot guarantee $\mathbf{u}_{t+1} \in V_{m_1,m_2}$ even if $\mathbf{u}_t \in V_{m_1,m_2}$. Addressing this issue, we use the Riemannian gradient and a retraction R on the manifold [51]. Firstly, instead of a standard gradient $\nabla C(\mathbf{u})$ we use the Riemannian gradient $\nabla_R C(\mathbf{u})$ which is a vector in the tangent space to the point $\mathbf{u} \in V_{m_1,m_2}$. The construction of a Riemannian gradient $\nabla_R C(\mathbf{u})$ as an orthogonal projection of the Euclidean gradient $\nabla C(\mathbf{u})$ onto the tangent space is described in the appendix B. Secondly, the next point of approximation $\mathbf{u}_{t+1}$ is determined by the SVD-retraction

$$\mathbf{u}_{t+1} = R^{\text{SVD}}_{\mathbf{u}_t}[-\alpha \nabla_R C(\mathbf{u}_t)].$$

The SVD-retraction is defined as $R^{\text{SVD}}_X(Y) = UV^\dagger$ after computing the SVD $X + Y = USV^\dagger$. By definition, this is just a projection of the matrix $(\mathbf{u}_t - \alpha \nabla_R C(\mathbf{u}_t))$ onto the manifold of $m_1 \times m_2$ isometric matrices. Using the retraction instead of a linear step equation (3) allows for the movement along the manifold in a desired direction, satisfying the constraint $\mathbf{u}_t \in V_{m_1,m_2}$ automatically for each step t.

Throughout the optimization, we calculate derivatives of the function with respect to gates using the automatic differentiation technique [38]. While for large and complex networks an analytical differentiation becomes impractical, the automatic differentiation computes derivatives of tensor network programs efficiently with machine precision, in contrast to numerical differentiation. The automatic differentiation,

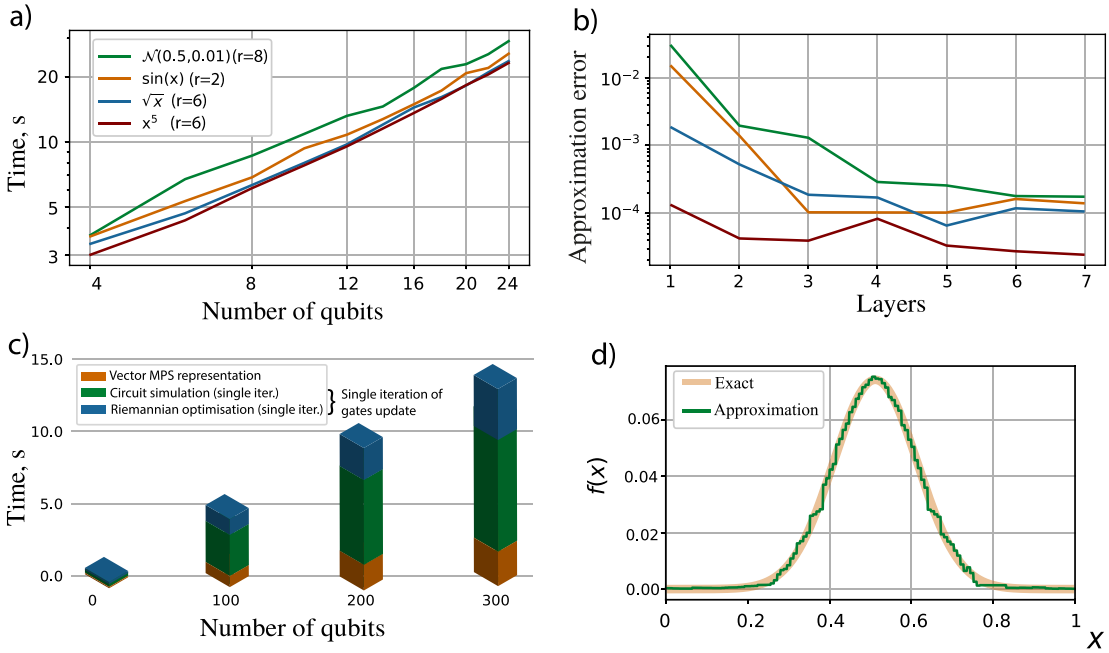

Figure 4. Results on runtime and approximation error for quantum state preparation using MPS formalism. (a) Runtime of our algorithm for the quantum circuit with 3 layers to prepare functions with 10^{-2} approximation error with respect to the number of qubits. Linear growth in the log–log scale indicates that the preparation is done in a polynomial time. Here, we do not perform simulations with larger circuits, since $2^{20} - 2^{30}$ points are enough for the one-dimensional function discretization. (b) Approximation error of the function approximation for different number of layers for 20 qubits and a fixed number of optimization iterations equal to 500. Overall, the deeper is the circuit, the higher is the approximation accuracy, however, starting from 5 layers, we observe a stagnation in the accuracy. (c) Runtimes needed to convert a function into MPS (orange) and to perform a single iteration of the circuit optimization (to contract tensor network (circuit simulation) in green and to perform a single iteration of the Riemannian optimization in blue). The approximation accuracy depends on the chosen number of iterations. The runtime features a linear scaling in the number of qubits. The number of layers in all numerical simulations is set to 3. (d) Exact Gaussian density function $\mathcal{N}(0.5, 0.01)$ and its approximation with an error $6 \cdot 10^{-4}$ obtained after 500 iterations of the algorithm using 10 qubits and 3 layers.

being usually applied for computer programs, is also applicable for tensor networks since the contraction of a tensor network can be represented as a computational graph [52]. Its algorithmic complexity is theoretically guaranteed to be not greater than the algorithmic complexity of the original program, which is equivalent to the complexity of a tensor network contraction [52].

Ultimately, optimization produces a set of unitary matrices that minimize $C(\mathbf{u})$. Such matrices define single-qubit gates in a quantum circuit for the approximation of the desired state.

2.3. Numerical simulations

In order to demonstrate the capabilities of our method, we consider the frequently-used analytical functions from different classes and prepare them in a many-qubit quantum state. In this work, we consider the polynomial x^5, the power function $\sqrt{x}$, the trigonometric function $\sin x$, and the Gaussian probability density function. Polynomials and trigonometric functions are often encountered when considering partial differential equations [28] and two-electron integral calculation [30]. Distribution functions are needed for sampling tasks [31] and the calculation of expectation values [6], which is necessary, e.g. in finance [12]. We sample the function over N points on an equidistant grid in $[0, 1]$ interval and form the function values in a N-dimensional vector, which we approximate using N amplitudes of $n = \log N$-qubit state.

In figure 4(a), we show the runtime of the whole state preparation routine as a function of the number of qubits. Here, we set the fidelity to 99% and the number of layers to 3, which is sufficient to approximate the considered vectors having low MPS ranks (also depicted in figure 4(a)). For all studied functions, our quantum state preparation protocol features a polylogarithmic scaling in the function discretization accuracy: the runtime scales as $O(\text{poly}(\log N))$. The number of layers can be increased which leads to an improvement in the approximation error—the infidelity decrease with the number of layers increase is presented in figure 4(b) for a 20-qubit circuit. The runtime breakdown is shown in figure 4(c). As an example, the approximation of a Gaussian density function sampled over 1024 points is depicted in figure 4(d), where a 10-qubit circuit with 3 layers was used to achieve a 99.94% fidelity.

The most significant advantage of our method is the ability to achieve a low approximation error using a shallow circuit. While deeper circuits with more parameters improve the accuracy, as shown in figure 4(b), a

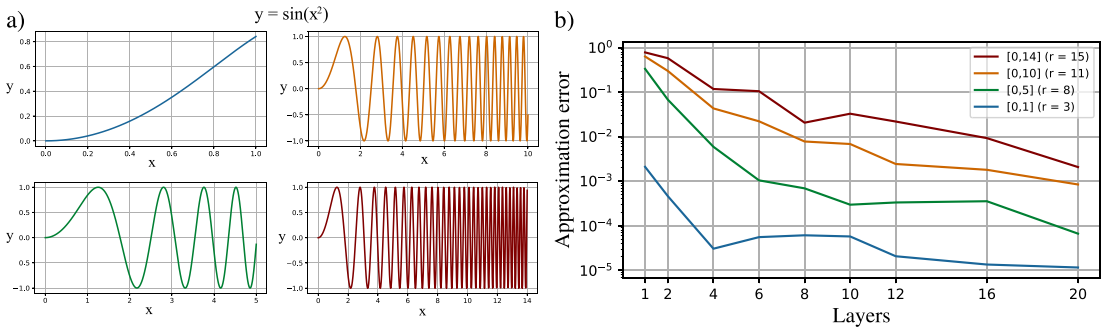

Figure 5. (a) Function $\sin(x^2)$ shown over four different intervals. As the interval increases, the function shows faster oscillations, which leads to an increase in its MPS rank. (b) Dependence of the approximation error of $\sin(x^2)$ via 10 qubits (1024 grid points) on the number of layers of the quantum circuit for different intervals. Corresponding ranks are shown in parentheses. As expected, with a larger interval (higher rank), more layers are needed to approximate the function.

low error of 0.01%–0.1% can be achieved using only 5 layers. Such an error is of the same order of magnitude as an error of state-of-the-art single two-qubit gates [53].

3. Discussion

3.1. Scaling analysis

Here, we analyze the complexity of the whole algorithm.

In order to investigate how the rank of MPS affects the approximation of our algorithm, we consider a $\sin(x^2)$ function specified on an interval $[0, b]$. As b increases, the function features faster oscillations and becomes more complex, as shown in figure 5(a), which leads to an increase of MPS ranks. In figure 5(b), we plot the approximation error for various intervals and number of layers. As expected, larger rank requires more layers and the error is larger for a given number of layers for higher ranks. As our goal is to approximate a function, we can fix the number of layers that provides certain accuracy. Generally, it is challenging to specify the exact number of layers for a given approximation as it strongly depends on the approximated function. Practically, for the functions investigated above shallow circuits are sufficient to provide decent accuracy.

With fixed number of layers L, the remaining parameters are the number of qubits n and the MPS rank r, which we assume to be restricted, i.e. $r \ll 2^{n/2}$. Each step of the algorithm, then, features the following complexity.

(i) *Representation of a function as an MPS* is done in $O(nr^2)$ time using the cross-approximation algorithm [44]. Moreover, since this procedure is required to be done only once in the whole algorithm (in contrast to optimization), the runtime contribution of this step can often be neglected.

(ii) *Circuit simulation* depends linearly on the number of qubits n with fixed number of layers according to figure 3(a). At the same time, the contraction time almost does not depend on the rank r, because all the complexity mainly contributed by the contraction of the quantum circuit, since the dimensions arising from the contraction of the quantum circuit are much larger than reasonable values of ranks (supplemental material, section 3). The treewidth [54] of the used circuit is constant with a fixed number of layers (supplemental material, section 3), thus the complexity of this step is reduced to be proportional to the total number of gates, i.e. $O(nL)$.

(iii) *Riemannian optimization* features the complexity of $O(nL)$, where the most computationally demanding part is the SVD-retraction, which has to be done $n(2L + 1)$ times for 2×2 matricies. For the fixed number of layers the runtime scaling is $O(n)$.

Figure 4(c) shows our experimental investigation of the complexity and confirms the scaling is indeed $O(n)$. It is clear that the most time-consuming part is the circuit simulation since it is has to be done in all iterations and requires an order of magnitude more operations than Riemannian optimization step. Thus, for further improvement of the algorithm, first of all, one should focus on this particular step.

3.2. Cut once, measure twice: a technique against barren plateaus

The problem, which remains to be considered, is that hardware-efficient quantum circuits suffer from the barren plateaus problem [32]. Namely, the number of circuit parameters to be optimized grows linearly with the number of qubits, but the gradients decrease exponentially. If random values of the parameters in

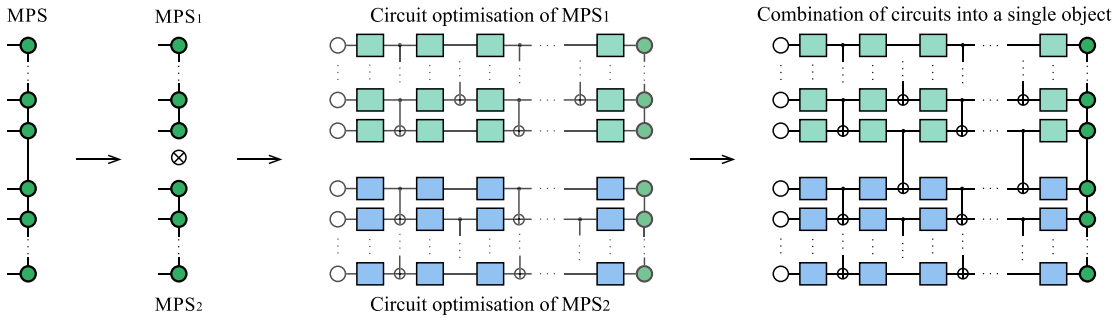

Figure 6. 'Cut once, measure twice' method to combat the barren plateaus problem. First, we partition an MPS into two halves and truncate the rank at the border of the partition to 1 (left). Middle: we optimize the resulting circuits for each of the halves. Right: we use the obtained gates as an initial guess in the optimization of the whole circuit. This method can be considered as a clever initialization of the circuit, which is known to be a cure against barren plateaus.

variational gates are chosen as an initial guess for the optimization, the initial fidelity can be less than the machine precision ($\sim 1/2^n$).

However, as was proposed in [32], selecting the appropriate starting optimization gates may help to avoid this issue. Combining the said idea with our MPS-based method, we modify the state preparation routine in the 'cut once, measure twice' way, shown in figure 6:

(i) We cut the MPS into two parts and truncate the rank at the border of the partition to 1.
(ii) For each of the resulting MPS vectors, we apply the state preparation algorithm for circuits with the same number of layers as for the initial MPS, but with fewer qubits. As a result, gates for the preparation of the 'small' MPS vectors are obtained.
(iii) We use thus obtained gates as an initial guess for the optimization of the original circuit for preparation of the initial MPS.

If a single cut is not enough, we cut each MPS again into halves and apply this algorithm recursively.

In the numerical experiments, we have found that a single run of this procedure is sufficient to optimize the 100-qubit circuit with 3 layers for the preparation of the Gaussian density function $\mathcal{N}(0.5, 0.01)$ on the interval $[0, 1]$ with a 99.6% approximation fidelity. At the same time, the state preparation algorithm implemented on the full 100-qubit circuit without any cuts prepares the same state with the fidelity close to zero.

4. Conclusion

We have extended the fruitful idea of MPS-based encoding schemes by considering variational hardware-efficient circuits as an encoder for high-dimensional vectors. The derived protocol allows us to solve the problem of efficient encoding of structured classical data into a multi-qubit quantum state. Examples of such data are smooth functions [45], probability distributions [55], videos [56], and image classifiers [57, 58].

As an illustrative example, we consider the preparation of states generated by specific analytical functions utilizing variational circuits with the canonical quantum gate set and restricting the connectivity to neighbor qubits. The latter assumption benefits not only physical realization, but also tensor-based simulations [47, 54], which we exploit to optimize the circuit parameters. We show that shallow circuits suffice to approximate considered functions with a high accuracy.

Our method can be applied to a wide variety of quantum algorithms. Firstly, partial differential equations (PDEs) after discretization are reduced to a system of linear equations $Ax = b$, where b is usually obtained by sampling an analytical function on an interval. All quantum linear system solvers [1–5] require preparation of b as the first step. Secondly, preparation of probability distribution functions is the main subroutine in calculation of the expectation values, which is a central problem in the finance [7, 12], e.g. risk analysis [6], and option pricing [59]. In addition, preparation of distributions can be applied to the sampling tasks [53, 60, 61]in computational statistics and quantum mechanics. One of the main bottlenecks in quantum machine learning [8, 62, 63] is the absence of quantum random access memory [64]. Our approach can be applied to the encoding of structured data, such as images [41, 65] and videos [56], into quantum circuits. Besides, our method can be utilized to prepare quantum states for sensing to maximise the detection efficiency [66, 67]. A more detailed explanation of these applications is given in appendix C.

It is worth mentioning that $10^6 - 10^9$ points are sufficient for the discretization of a single-variable function in the majority of problems, which requires 20–30 qubits. Higher number of qubits is necessary for multi-parameter functions, where our approach works as well in case when the correlations between variables are relatively small. Such multi-parameters functions are arising in the multi-dimensional PDEs [68], option pricing [69] and other problems [70]. The preparation of high-dimensional vectors requires large quantum circuits, which are generally hard to optimize due to the barren plateaus. Here we proposed a 'cut once, measure twice' approach that facilities the circuit optimization by proper initialization.

We also investigated circuits of different structures and found that there is no superior architecture, leaving the choice of the circuit to the user or hardware topology. Our algorithm is generic and can be applied to an arbitrary circuit, where variational gates may be applied over multiple qubits and/or defined by the hardware, e.g. rf-pulses, laser radiation, etc.

Following the paradigm of hybrid classical–quantum computing, we expect tensor-based algorithms to be the method of choice since the classical tensor networks subroutines feature logarithmic complexity in memory consumption and runtime and can be easily translated to quantum hardware for further manipulations. Mainly, the classical part can be done via tensor networks and then its output is transferred to a quantum computer with the help of our algorithm. Besides, the output quantum state vector that is well described as an MPS can be obtained using logarithmic—in the vector size—number of measurements [71]. Such an approach concludes the full cycle of hybrid computation MPS $\rightarrow$ QC $\rightarrow$ MPS, where all steps performed in logarithmic complexity.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

The work of Ar A M, A A T, F N, and M R P was supported by Terra Quantum A G. The work of S V D was financially supported by Terra Quantum A G through the consultancy agreement.

Ar A M, A A T and M R P devised the algorithm. A A T wrote the code and performed the experiments. Ar A M, S V D, M R P, A A T analyzed and improved the algorithm. F N and M R P supervised the project. All authors discussed the results, wrote the manuscript and contributed to the work.

Appendix A. Software

We perform quantum circuits simulation using the `QMware` system [72]. The circuits are assembled using QASM language and are, therefore, library-agnostic, i.e. can be run using Cirq, Qiskit, PennyLane, etc. We implement the proposed algorithm as Python library `QPrep`, which is publicly available in [73]. The library requires a function, number of sampling points and layers as an input and provides a circuit for the corresponding quantum state preparation.

Appendix B. Riemannian optimization

Riemannian optimization allows us to optimize a function constrained on a Riemannian manifold. In this work, our purpose is to minimize the infidelity equation (2) on the manifold of isometric matrices. We calculate the (Euclidean) gradient, find its projection onto the tangent space, and project the updated matrix onto the space of isometric matrices.

Here, we denote the manifold of isometric matrices of size $m_1 \times m_2$ as $\mathcal{M}$ and the tangent space at point $X \in \mathcal{M}$ as $T_X\mathcal{M}$. The Riemannian gradient of some differentiable function $f(X)$ at point $X \in \mathcal{M}$ of isometric matrices is calculated as (see supplemental material, section 4 for further details)

$$\nabla_R f(X) = \nabla f(X) - \frac{1}{2} X \left[(\nabla f(X))^\dagger X + X^\dagger \nabla f(X) \right].$$

By definition, $\nabla_R f(X)$ is an orthogonal projection of the Euclidean gradient $\nabla f(X)$ onto the tangent space $T_X\mathcal{M}$. The last step is the projection of the updated matrix $(X - \alpha \nabla_R f(X))$ on the manifold of isometric matrices $\mathcal{M}$, which is performed via SVD-retraction:

$$R_X^{\mathrm{SVD}}(-\alpha \nabla_R f(X)) = UV^\dagger,$$

where $X - \alpha \nabla_R f(X) = USV^\dagger$ is the SVD decomposition.

Appendix C. Applications

C.1. Sampling

The most straightforward way to apply quantum state preparation is the sampling task [31]. Our algorithm allows for the preparation of probability distribution functions, and then measurement of qubit's states provides us samples. For instance, the function of interest can be the Gaussian distribution, that has restricted MPS ranks in case of good structure of the covariance matrix (small ranks of off-diagonal blocks of the matrix) [74].

C.2. Solution of linear systems of equations

Linear systems of equations usually arise considering partial differential equations after their discretization. For instance, simple one-dimensional Poisson equation

$$-u''(x) = f(x) \quad x \in [0,1]\ u(0) = u(1),$$

gives the system of linear equations that can be discretized on the uniform grid

$$Ax = b,$$

where $b_i = f(x_i)$ is the function value at the corresponding point.

Most of quantum algorithms for linear systems assumes that it is possible to efficiently prepare any desired state b [1, 3, 5]. However, there is still no general and established framework for this task and we expect that our approach facilitates the vectors encoding.

C.3. Expectation value calculation

Calculation of the expectation value is one of the main routine in the computational finance, and many quantum algorithms are devoted to this problem [7, 12]. At the same time, the quantum state preparation is one of the main subroutine in these approaches, so our method is applied here as well.

For example, in the risk analysis, the Value at Risk needs to be calculated [6]. That is to find the smallest X such that:

$$P[x \leqslant X] \geqslant (1-\alpha)$$

for fixed α. This can be reformulated to calculation of the expectation value of the Heaviside step function $\theta(x-X)$ with various values of X. Suppose we have distribution function $p(x)$ defined on the interval (a,b), then:

$$P[x \leqslant X] = \int_a^X p(x)\mathrm{d}x = \int_a^b p(x)\theta(x-X)\mathrm{d}x = \mathbf{E}(\theta(x-X)).$$

Thus, the task is reduced to finding the expectation value. If one wants to calculate it on a quantum computer one can use the amplitude estimation algorithm [75] which provides quadratically better error scaling than classical Monte Carlo. But first of all, one needs to prepare the distribution function $p(x)$ discretized on the interval (a,b):

$$|p\rangle = \sum_i \sqrt{p(x_i)}|i\rangle.$$

ORCID iDs

Ar A Melnikov https://orcid.org/0000-0002-6975-4009
A A Termanova https://orcid.org/0000-0002-7842-9574
S V Dolgov https://orcid.org/0000-0002-1647-4214
F Neukart https://orcid.org/0000-0002-2562-1618
M R Perelshtein https://orcid.org/0000-0001-7912-1750

References

[1] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 15
[2] Perelshtein M R, Pakhomchik A I, Melnikov A A, Novikov A A, Glatz A, Paraoanu G S, Vinokur V M and Lesovik G B 2022 Solving large-scale linear systems of equations by a quantum hybrid algorithm *Ann. Phys., Lpz.* **534** 2200082
[3] Childs A M, Kothari R and Somma R D 2017 Quantum algorithm for systems of linear equations with exponentially improved dependence on precision *SIAM J. Comput.* **46** 1920–50

[4] Bravo-Prieto C, LaRose R, Cerezo M, Subasi Y, Cincio L and Coles P J 2019 Variational quantum linear solver (arXiv:1909.05820)
[5] Childs A M, Liu J-P and Ostrander A 2021 High-precision quantum algorithms for partial differential equations *Quantum* **5** 574
[6] Woerner S and Egger D J 2019 Quantum risk analysis *npj Quantum Inf.* **5** 15
[7] Markov V, Stefanski C, Rao A and Gonciulea C 2022 A generalized quantum inner product and applications to financial engineering (arXiv:2201.09845)
[8] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195–202
[9] Broughton M *et al* 2020 Tensorflow quantum: a software framework for quantum machine learning (arXiv:2003.02989)
[10] Liu Y, Arunachalam S and Temme K 2021 A rigorous and robust quantum speed-up in supervised machine learning *Nat. Phys.* **17** 1013–7
[11] Cerezo M *et al* 2021 Variational quantum algorithms *Nat. Rev. Phys.* **3** 625–44
[12] Herman D, Googin C, Liu X, Galda A, Safro I, Sun Y, Pistoia M and Alexeev Y 2022 A survey of quantum computing for finance (https://doi.org/10.48550/arXiv.2201.02773)
[13] Duan B, Yuan J, Chao-Hua Y, Huang J and Hsieh C-Y 2020 A survey on HHL algorithm: from theory to application in quantum machine learning *Phys. Lett.* A **384** 126595
[14] Plesch M and Brukner Časlav 2011 Quantum-state preparation with universal gate decompositions *Phys. Rev.* A **83** 032302
[15] Araujo I, Park D, Petruccione F and Adenilton da S 03 2021 A divide-and-conquer algorithm for quantum state preparation *Sci. Rep.* **11** 6329
[16] Bharti K *et al* 2022 Noisy intermediate-scale quantum algorithms *Rev. Mod. Phys.* **94** 015004
[17] Marin-Sanchez G, Gonzalez-Conde J and Sanz M 2021 Quantum algorithms for approximate function loading (https://doi.org/10.48550/arXiv.2111.07933)
[18] Araujo I F, Blank C and da Silva A J 2021 Entanglement as a complexity measure for quantum state preparation (https://doi.org/10.48550/arXiv.2111.03132)
[19] Rattew A G, Sun Y, Minssen P and Pistoia M 2021 The efficient preparation of normal distributions in quantum registers *Quantum* **5** 609
[20] Ballani J and Grasedyck L 2014 Tree adaptive approximation in the hierarchical tensor format *SIAM J. Sci. Comput.* **36** A1415–31
[21] Román O 2014 A practical introduction to tensor networks: matrix product states and projected entangled pair states *Ann. Phys., NY* **349** 117–58
[22] Ulrich S 2011 The density-matrix renormalization group in the age of matrix product states *Ann. Phys., NY* **326** 96–192
[23] Robert N C Pfeifer J H and Verstraete F 2014 Faster identification of optimal contraction sequences for tensor networks *Phys. Rev.* E **90** 033315
[24] Perez-Garcia D, Verstraete F, Wolf M M and Cirac J I 2006 Matrix product state representations (arXiv:quant-ph/0608197)
[25] Naumov A, Melnikov A, Abronin V, Oxanichenko F, Izmailov K, Pflitsch M, Melnikov A and Perelshtein M 2023 Tetra-AML: automatic machine learning via tensor networks (arXiv:2303.1621)
[26] Morozov D, Melnikov A, Shete V and Perelshtein M 2023 Protein-protein docking using a tensor train black-box optimization method (arXiv:2302.03410)
[27] Belokonev N, Melnikov A, Podapaka M, Pinto K, Pflitsch M and Perelshtein M 2023 Optimization of chemical mixers design via tensor trains and quantum computing (arXiv:2304.12307)
[28] Chorin A J 1968 Numerical solution of the Navier-Stokes equations *Math. Comput.* **22** 745–62
[29] Kornev E, Dolgov S, Pinto K, Pflitsch M, Perelshtein M and Melnikov A 2023 Numerical solution of the incompressible Navier-Stokes equations for chemical mixers via quantum-inspired tensor train finite element method (arXiv:2305.10784)
[30] Khoromskaia V, Khoromskij B N and Schneider R 2013 Tensor-structured factorized calculation of two-electron integrals in a general basis *SIAM J. Sci. Comput.* **35** A987–1010
[31] Hangleiter D and Eisert J 2022 Computational advantage of quantum random sampling (arXiv:2206.04079)
[32] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 1–6
[33] Schön C, Solano E, Verstraete F, Cirac J I and Wolf M M 2005 Sequential generation of entangled multiqubit states *Phys. Rev. Lett.* **95** 110503
[34] Schön C, Hammerer K, Wolf M M, Cirac J I and Solano E 2007 Sequential generation of matrix-product states in cavity QED *Phys. Rev.* A **75** 032311
[35] Lubasch M, Joo J, Moinier P, Kiffner M and Jaksch D 2020 Variational quantum algorithms for nonlinear problems *Phys. Rev.* A **101** 1
[36] Ran S-J 2020 Encoding of matrix product states into quantum circuits of one- and two-qubit gates *Phys. Rev.* A **101** 032310
[37] Zhou P-F, Hong R and Ran S-J 2021 Automatically differentiable quantum circuit for many-qubit state preparation *Phys. Rev.* A **104** 042601
[38] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2018 Automatic differentiation in machine learning: a survey *J. Mach. Learn. Res.* **18** 1–43
[39] Lubich C, Oseledets I V and Vandereycken B 2015 Time integration of tensor trains *SIAM J. Numer. Anal.* **53** 917–41
[40] Kressner D, Steinlechner M and Vandereycken B 2016 Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure *SIAM J. Sci. Comput.* **38** A2018–44
[41] Holmes A and Matsuura A Y 2020 Efficient quantum circuits for accurate state preparation of smooth, differentiable functions *2020 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) pp 169–79
[42] Perelshtein M R *et al* 2023 NISQ-compatible approximate quantum algorithm for unconstrained and constrained discrete optimization (arXiv:2305.14197)
[43] Stewart G W 1993 On the early history of the singular value decomposition *SIAM Rev.* **35** 551–66
[44] Oseledets I and Tyrtyshnikov E 2010 TT-cross approximation for multidimensional arrays *Linear Algebr. Appl.* **432** 70–88
[45] Oseledets I 2010 Constructive representation of functions in low-rank tensor formats *Constr. Approx.* **37** 09
[46] Gray J and Kourtis S 2021 Hyper-optimized tensor network contraction *Quantum* **5** 410
[47] Liu Y *et al* 2021 Closing the quantum supremacy gap *Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis* (https://doi.org/10.1145/3458817.3487399)
[48] Huber B, Schneider R and Wolf S 2017 A randomized tensor train singular value decomposition *Compressed Sensing and Its Applications* (Berlin: Springer) pp 261–90
[49] Evenbly G 2022 A practical guide to the numerical implementation of tensor networks I: contractions, decompositions and gauge freedom (https://doi.org/10.48550/arXiv.2202.02138)

[50] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 3
[51] Luchnikov I A, Krechetov M E and Filippov S N 2021 Riemannian geometry and automatic differentiation for optimization problems of quantum physics and quantum technologies *New J. Phys.* **23** 073006
[52] Liao H-J, Liu J-G, Wang L and Xiang T 2019 Differentiable programming tensor networks *Phys. Rev.* X **9** 031041
[53] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505–10
[54] Markov I L and Shi Y 2008 Simulating quantum computation by contracting tensor networks *SIAM J. Comput.* **38** 963–81
[55] Glasser I, Sweke R, Pancotti N, Eisert J and Ignacio Cirac J 2019 Expressive power of tensor-network factorizations for probabilistic modeling *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc.) (https://doi.org/10.5555/3454287.3454421)
[56] Ahmadi-Asl S, Asante-Mensah M G, Cichocki A, Phan A-H, Oseledets I and Wang J 2022 Cross tensor approximation for image and video completion (arXiv:2207.06072)
[57] Stoudenmire E and Schwab D J 2016 Supervised learning with tensor networks *Advances in Neural Information Processing Systems* vol 29 (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper_files/paper/2016/file/5314b9674c86e3f9d1ba25ef9bb32895-Paper.pdf)
[58] Huggins W, Patil P, Mitchell B, Birgitta Whaley K and Miles Stoudenmire E 2019 Towards quantum machine learning with tensor networks *Quantum Sci. Technol.* **4** 024001
[59] Stamatopoulos N, Egger D J, Sun Y, Zoufal C, Iten R, Shen N and Woerner S 2020 Option pricing using quantum computers *Quantum* **4** 291
[60] Lund A P, Bremner M J and Ralph T C 2017 Quantum sampling problems, bosonsampling and quantum supremacy *npj Quantum Inf.* **3** 1–8
[61] Dolgov S, Anaya-Izquierdo K, Fox C and Scheichl R 2020 Approximation and sampling of multivariate probability distributions in the tensor train decomposition *Stat. Comput.* **30** 603–25
[62] Sagingalieva A, Kurkin A, Melnikov A, Kuhmistrov D, Perelshtein M, Melnikov A, Skolik A and David V D 2022 Hyperparameter optimization of hybrid quantum neural networks for car classification (arXiv:2205.04878)
[63] Melnikov A, Kordzanganeh M, Alodjants A and Lee R-K 2023 Quantum machine learning: from physics to software engineering *Adv. Phys.* X **8** 2165452
[64] Giovannetti V, Lloyd S and Maccone L 2008 Quantum random access memory *Phys. Rev. Lett.* **100** 16
[65] Hou H and Andrews H 1978 Cubic splines for image interpolation and digital filtering *IEEE Trans. Acoust. Speech Signal Process.* **26** 508–17
[66] Danilin S and Weides M 2021 Quantum sensing with superconducting circuits (arXiv:2103.11022)
[67] Gusarov N N, Perelshtein M R, Hakonen P J and Paraoanu G S 2023 Optimized emulation of quantum magnetometry via superconducting qubits *Phys. Rev.* A 107 052609
[68] Dolgov S V, Khoromskij B N and Oseledets I V 2012 Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation *SIAM J. Sci. Comput.* **34** A3016–38
[69] Wang X and Sloan I H 2011 Quasi-Monte Carlo methods in financial engineering: an equivalence principle and dimension reduction *Oper. Res.* **59** 80–95
[70] Cichocki A, Lee N, Oseledets I, Phan A-H, Zhao Q and Mandic D P 2016 Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions *Found. Trends Mach. Learn.* **9** 249–429
[71] Cramer M, Plenio M B, Flammia S T, Somma R, Gross D, Bartlett S D, Landon-Cardinal O, Poulin D and Liu Y-K 2010 Efficient quantum state tomography *Nat. Commun.* **1** 1–7
[72] Perelshtein M, Sagingalieva A, Pinto K, Shete V, Pakhomchik A, Melnikov A, Neukart F, Gesek G, Melnikov A and Vinokur V 2022 Practical application-specific advantage through hybrid quantum computing (https://doi.org/10.48550/arXiv.2205.04858)
[73] Melnikov A A, Termanova A A, Dolgov S V, Neukart F and Perelshtein M R `QPrep:` Library for quantum state preparation using tensor networks (available at: https://github.com/Terra-Quantum-AG/quantum_state_preparation)
[74] Rohrbach P B, Dolgov S, Grasedyck L and Scheichl R 2020 Rank bounds for approximating gaussian densities in the tensor-train format (https://doi.org/10.48550/arXiv.2001.08187)
[75] Brassard G, Hoyer P, Mosca M and Tapp A 2002 Quantum amplitude amplification and estimation *Contemp Math.* **305** 53–74