QGOpt: Riemannian optimization for quantum technologies

Ilia A. Luchnikov[1,2,3*]**, Alexander Ryzhov**[2]**, Sergey N. Filippov**[1,4,5] **and Henni Ouerdane**[2]

1 Moscow Institute of Physics and Technology, Institutskii Pereulok 9,
Dolgoprudny, Moscow Region 141700, Russia
2 Center for Energy Science and Technology, Skolkovo Institute of Science and Technology,
Moscow 121205, Russia
3 Russian Quantum Center, Skolkovo, Moscow 143025, Russia
4 Steklov Mathematical Institute of Russian Academy of Sciences,
Gubkina Street 8, Moscow 119991, Russia
5 Valiev Institute of Physics and Technology of Russian Academy of Sciences,
Nakhimovskii Prospect 34, Moscow 117218, Russia

⋆ Ilia.Luchnikov@skoltech.ru

Abstract

Many theoretical problems in quantum technology can be formulated and addressed as constrained optimization problems. The most common quantum mechanical constraints such as, e.g., orthogonality of isometric and unitary matrices, CPTP property of quantum channels, and conditions on density matrices, can be seen as quotient or embedded Riemannian manifolds. This allows to use Riemannian optimization techniques for solving quantum-mechanical constrained optimization problems. In the present work, we introduce QGOpt, the library for constrained optimization in quantum technology. QGOpt relies on the underlying Riemannian structure of quantum-mechanical constraints and permits application of standard gradient based optimization methods while preserving quantum mechanical constraints. Moreover, QGOpt is written on top of TensorFlow, which enables automatic differentiation to calculate necessary gradients for optimization. We show two application examples: quantum gate decomposition and quantum tomography.

Contents

1 Introduction

Many quantum-mechanical problems can be solved using optimization methods as illustrated by the following examples. The ground state of a quantum system with Hamiltonian H can be found using the variational method, which is akin to an optimization problem [1]:

$$|\Omega\rangle = \operatorname*{argmin}_{|\psi\rangle} \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}, \tag{1}$$

where $|\psi\rangle$ is a non-normalized trial state, $|\Omega\rangle$ is the non-normalized ground state. This formulation of a ground state search problem was successfully used for the study of many-body quantum systems [2,3]. In particular, the ground state of a correlated spin system can be found in the following forms: matrix product states [4–6], projected entangled pair states [7,8] or neural networks [9–11]. To perform variational energy optimization one can utilize optimization algorithms such as the density matrix renormalization group [12,13], the time evolving block decimation [14–16] for tensor network architectures, the quantum natural gradient [17], and adaptive first-order optimization methods like the Adam optimizer [18] for neural-networks-based quantum parametrization.

Problems of reconstruction of quantum states, quantum channels and quantum processes from measured data can also be formulated as optimization problems. For example, the state of a many-body quantum system can be reconstructed with neural networks by maximization of the logarithmic likelihood function on a set of measurement outcomes [19–22]. The Choi matrix of an unknown quantum channel can be reconstructed in a tensor network form via the minimization of the Kullback-Leibler divergence [23]. Non-Markovian quantum dynamics can be reconstructed from measured data in different ways [24,25] by use of optimization algorithms.

Some quantum mechanics problems require keeping certain constraints while minimizing or maximizing an objective function. For example, quantum phase transitions can be described using an entanglement renormalization technique, which requires an optimization over matrices with orthogonality constraints, i.e. isometric matrices. To solve this problem, Vidal and Evenbly suggested an algorithm [26–28] that does not have analogs in standard optimization theory. Another example of a constrained optimization problem emerging in quantum mechanics is quantum channel tomography. It requires preservation of natural "quantum" constraints,

i.e. the completely positive and trace preserving (CPTP) property of quantum channels [29]. Constraints preservation here can be achieved by using a particular parametrization or by adding regularizers that ensure that the constraints are satisfied.

Adding regularizers into a loss function merely provides approximate preservation of constraints, and a naive parametrization may lead to over-parametrization and result in the optimization slowing down. One therefore needs a universal approach to quantum technology optimization. As many natural "quantum" constraints can be seen as Riemannian manifolds, Riemannian optimization can become a candidate well-suited for the role of universal framework for constrained optimization in quantum mechanics. In the present work, we introduce QGOpt (Quantum Geometric Optimization) [30], a library for Riemannian optimization in quantum mechanics and quantum technologies. It allows one to perform an optimization with all typical constraints of quantum mechanics.

This article is organized as follows. In Sec. 2, we give an overview of Riemannian optimization. We then turn to Riemannian manifolds in quantum mechanics in Sec. 3. In Sec. 4, we present the QGOpt application programming interface (API), and we illustrate its use in Sec. 5, with two examples: quantum gate decomposition and quantum channel tomography. In Sec. 6, we also show that QGOpt can handle optimization over an arbitrary Cartesian product of manifolds.

2 Overview of the Riemannian optimization

While optimizing an objective function defined on the Euclidean space, one performs a sequence of elementary operations like points and vectors transportation. We call these elementary operations optimization primitives. For example, one iteration of the simplest gradient descent method involves an update of the current estimation of the optimal point x_t as follows: $x_{t+1} = x_t + v_t$, where $v_t = -\eta \nabla f(x_t)$ is a vector tending to improve the current estimation, t is the number of previous iterations, and η is the step-size. This update can be seen as a transportation of a point x_t along a vector v_t. More sophisticated algorithms may require keeping additional information about the optimization landscape in terms of vectors $\{m_t^{(0)}, \ldots, m_t^{(N)}\}$ associated with the current point x_t. These vectors should be transported together with x_t to a new point and then updated according to a particular algorithm. However, as transportation of vectors in a Euclidean space is the trivial identity transformation, it may be safely skipped. Optimization on curved spaces requires a generalization of optimization primitives in a certain way. As an example of optimization algorithms we consider a gradient descent with momentum [31] and its Riemannian generalization [32, 33]. We keep our overview simple, and for an in-depth introduction to the topic, we recommend references [34, 35].

Let us assume that we aim to minimize the value of a function $f : \mathbb{R}^n \to \mathbb{R}$, and that we have access to its gradient $\nabla f(x)$. In the Euclidean space $\mathbb{R}^n$, a gradient descent with momentum consists of the following steps wrapped into a loop:

1. Calculation of the momentum vector $m_{t+1} = \beta m_t + (1 - \beta)\nabla f(x_t)$,

2. Taking a step along the direction of a momentum vector $x_{t+1} = x_t - \eta m_{t+1}$,

where the initial momentum vector m_0 is the null vector, β is a hyperparameter whose value is usually taken around $\beta \approx 0.9$, and η is the size of the optimization step.

Let us assume now that a function f is defined on a Riemannian manifold $\mathcal{M}$ that is embedded in the Euclidean space: $f : \mathcal{M} \to \mathbb{R}$. Then we can no longer apply the standard scheme of gradient descent with momentum because it clearly takes x_t out of the manifold $\mathcal{M}$. This scheme can be generalized step by step. First, we have to generalize the notion of

gradient. The standard Euclidean gradient is not a tangent vector to a manifold and it does not take into account the metric of a manifold. One may introduce the Riemannian gradient that can be constructed based on the standard gradient $\nabla f(x)$. The Riemannian gradient lies in the space tangent to a point x and properly takes the metric of a tangent space into account. Although an optimization algorithm takes a step along a vector tangent to a manifold, it still takes a point out of the manifold. In order to fix this issue, one can replace a straight line step with a proper curved line step. In the Riemannian geometry the generalization of the straight line step is given by the notion of exponential map that reads

$$x_{\text{out}} = \text{Exp}_{x_{\text{in}}}(v) = \gamma(1), \tag{2}$$

where $\gamma(t)$ is a geodesic [36] such that $\gamma(0) = x_{\text{in}}$ and $\frac{d\gamma(t)}{dt}\big|_{t=0} = v$, x_{in} is an initial point on a manifold, x_{out} is a final point. However, in practice the calculation of a geodesic is often computationally too inefficient. In these cases, one can use a retraction instead of an exponential map, which is a first-order approximation of a geodesic [36]:

$$\tilde{x}_{\text{out}} = R_{x_{\text{in}}}(v), \tag{3}$$

where $\tilde{x}_{\text{out}}$ also lies in a manifold and $\|\tilde{x}_{\text{out}} - x_{\text{out}}\| = O(\|v\|^2)$. A retraction is not unique so it can be chosen to be computationally efficient.

The gradient descent with momentum also requires to transport the momentum vector at each iteration from a previous point to a new point. The Euclidean version of the gradient descent with momentum does not have an explicit step with transportation of the momentum vector because in the Euclidean space transportation of a vector is trivial. However, this step is necessary in the Riemannian case, where the trivial Euclidean vector transportation takes a vector out of a tangent space. A vector transport $\tau_{x,w}(v)$ is the result of transportation of a vector v along a vector w which takes into account that a tangent space varies from one manifold's point to another in the Riemannian case. The overall Riemannian generalization of the gradient descent with momentum can be summarized as follows:

1. Calculation of the momentum vector $\tilde{m}_{t+1} = \beta m_t + (1-\beta)\nabla_R f(x_t)$,

2. Taking a step along a new direction of the momentum $x_{t+1} = R_{x_t}(-\eta \tilde{m}_{t+1})$,

3. Transport of the momentum vector to a new point x_{t+1}: $m_{t+1} = \tau_{x_t, -\eta \tilde{m}_{t+1}}(\tilde{m}_{t+1})$.

Note that other first-order optimization methods can be generalized in a similar fashion.

3 Riemannian manifolds in quantum mechanics

Many objects of quantum mechanics can be seen as elements of smooth manifolds. However, their mathematical description, suitable for numerical algorithms, may involve some abstract constructions that should be clarified. In this section we consider an illustrative example of the set of Choi matrices and describe this set as a smooth quotient manifold. We restrict our consideration to a plain description of all necessary mathematical concepts. At the end of the section, we also list all the manifolds implemented in the QGOpt library and describe their possible use.

The evolution of any quantum system that interacts with its environment can be described by a quantum channel. Here, we consider quantum channels defined as the following CPTP linear map: $\Phi : \mathbb{C}^{n \times n} \to \mathbb{C}^{n \times n}$. Any quantum channel can be represented through its Choi matrix [29]. A Choi matrix is a positive semi-definite operator $C \in \mathbb{C}^{n^2 \times n^2}$ that has a constraint $\text{Tr}_p(C) = \mathbb{1}$, where Tr_p is a partial trace over the second subsystem and $\mathbb{1}$ is the identity matrix.
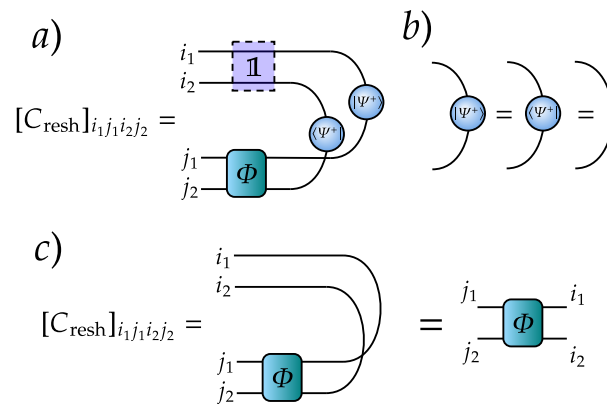
Figure 1: a) Diagrammatic representation of the Choi matrix. The block denoted by $\mathbb{1}$ represents the identity map in the definition of the Choi matrix. b) One can note that the state of a two-component quantum system $|\Psi^+\rangle$ can be seen as the identity matrix. c) Finally, we note that the Choi matrix is a quantum channel itself.

To make the notion of the partial trace less abstract, let us consider a piece of the TensorFlow code, which computes a partial trace of a Choi matrix. First, we apply a reshape operation to a Choi matrix that changes the shape of a matrix as follows

```
C_resh = tf.reshape(C, (n, n, n, n))
```

The tensor $C_{\text{resh}} \in \mathbb{C}^{n \times n \times n \times n}$ is an alternative representation of the Choi matrix. Further in the text, we distinguish two equivalent representations of a Choi matrix: C and C_{resh}. The partial trace of a Choi matrix can be calculated using C_{resh} as follows $[\text{Tr}_p(C)]_{i_1 i_2} = \sum_j [C_{\text{resh}}]_{i_1 j i_2 j}$. Practically it can be done by running

```
tf.einsum('ikjk->ij', C_resh)
```

which means that we take a trace over the first and third indices (with numeration of indices starting from 0).

The Choi–Jamiołkowski isomorphism [37] establishes a one-to-one correspondence between quantum channels and Choi matrices. One can calculate the Choi matrix of a known quantum channel as follows

$$C = (\mathbb{1} \otimes \Phi)[|\Psi^+\rangle \langle \Psi^+|], \tag{4}$$

where $|\Psi^+\rangle = \sum_{i=1}^n |i\rangle \otimes |i\rangle$ and $\{|i\rangle\}_{i=1}^n$ is an orthonormal basis in $\mathbb{C}^n$. In order to show that the Choi matrix is a quantum channel itself, we consider the representation of Eq. (4) in terms of tensor diagrams [38, 39]. The reshaped version of a Choi matrix $[C_{\text{resh}}]_{i_1 j_1 i_2 j_2}$ is shown in Fig. 1. The tensor diagrams in Fig. 1 show that $|\Psi^+\rangle$ and $\mathbb{1}$ in the definition of the Choi matrix lead only to relabeling of multi-indices.

The set of all Choi matrices of size $n^2 \times n^2$ (the corresponding quantum channel acts on density matrices of size $n \times n$) C_n is the following subset of $\mathbb{C}^{n^2 \times n^2}$

$$C_n = \left\{ C \in \mathbb{C}^{n^2 \times n^2} \middle| C \geq 0, \ \text{Tr}_p(C) = \mathbb{1} \right\}, \tag{5}$$

where $C \geq 0$, and $\text{Tr}_p(C) = \mathbb{1}$ corresponds to the CPTP property of the corresponding quantum channel. This subset can be described as a Riemannian manifold that admits different Riemannian optimization algorithms. In order to describe C_n as a Riemannian manifold, we may parametrize the Choi matrix with an auxiliary matrix $A \in \mathbb{C}^{n^2 \times n^2}$:

$$C = AA^\dagger. \tag{6}$$

Figure 2: a) Decomposition of a Choi matrix into A and $A^\dagger$. b) Diagrammatic representation of the isometric property of A.
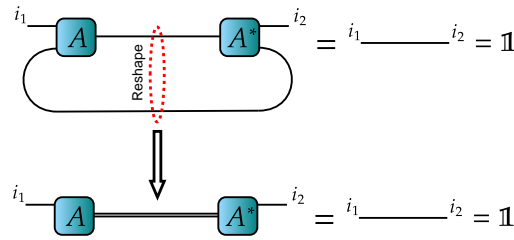

Figure 3: Diagrammatic representation of the reshape operation turning the tensor A_resh into an isometric matrix.

The matrix C is positive semi-definite by construction. We also distinguish $A \in \mathbb{C}^{n^2 \times n^2}$ and its reshaped version $A_\text{resh} \in \mathbb{C}^{n \times n \times n^2}$ that are connected by the reshape operation. The condition on a partial trace of a Choi matrix transforms to the following equality:

$$[\text{Tr}_p(C)]_{i_1 i_2} = [\text{Tr}_p(A^\dagger A)]_{i_1 i_2} = \sum_{kj} [A_\text{resh}]^*_{k i_1 j} [A_\text{resh}]_{k i_2 j} = \delta_{i_1 i_2}, \tag{7}$$

and its diagrammatic form is given in Fig. 2. One can see that if in Eq. (7) we recast the two indices k and j into one index q, we then end up with the following relation:

$$\sum_q [A_\text{resh}]^*_{q i_1} [A_\text{resh}]_{q i_2} = \delta_{i_1 i_2}, \tag{8}$$

which means that $[A_\text{resh}]_{qi}$ is an isometric matrix and the corresponding tensor $[A_\text{resh}]_{kij}$ is a reshaped isometric matrix. The corresponding diagrammatic representation of Eq. (8) is given in Fig. 3. We call such a tensor obtained by reshaping an isometric matrix an isometric tensor. The set of all complex isometric matrices of fixed size forms a Riemannian manifold called complex Stiefel manifold [40] that we denote as St. Equations (7) and (8), and the diagram in Fig. 3 show that the set of tensors A_resh can be seen as a complex Stiefel manifold.

At first glance, it looks like we have shown that the set of Choi matrices can be seen as a Stiefel manifold, but there is a problem that invalidates this statement: the matrices A and AQ, where Q is an arbitrary unitary matrix, correspond to the same Choi matrix; in other words, we have an equivalence relation $AQ \sim A$. Indeed,

$$C = AQQ^\dagger A^\dagger = AA^\dagger. \tag{9}$$

A diagrammatic version of Eq. (9) is depicted in Fig. 4. It shows that for any A there is a family of equivalent matrices $[A] = \{AQ | Q \in \mathbb{C}^{n^2 \times n^2}, Q^\dagger Q = QQ^\dagger = \mathbb{1}\}$, which is the equivalence class of A and leads to the same Choi matrix. One can eliminate this symmetry by turning to a quotient manifold $\text{St}/Q = \{[A] | A \in \text{St}\}$, which consists of equivalence classes. This rather abstract construction can be imagined as a projection of a manifold along surfaces representing equivalence classes (see Fig. 5). Having a map $\pi(A) = [A]$ and a horizontal lift [34], that connects
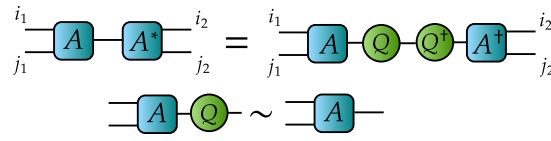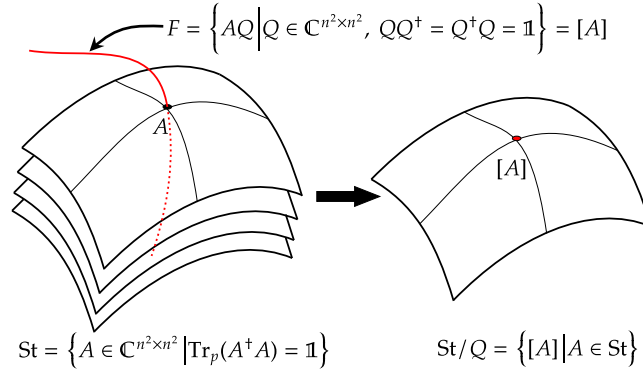
Figure 4: Diagrammatic representation of Eq. (9).

Figure 5: Graphical representation of the transition from the manifold St of all matrices A, to the quotient manifold St/Q that eliminates undesirable symmetry of the parametrization. The red curve represents a particular equivalence class F that is also called a fiber.

tangent spaces of St/Q and tangent spaces of St, one can describe the abstract manifold St/Q through St. The quotient manifold St/Q can be further identified with the set of Choi matrices C_n. It allows one to perform a Riemannian optimization on C_n, by using the parametrization $C = AA^\dagger$. Mathematical details of this construction are given in Appendix A.

The example of the quotient manifold representing the Choi matrices through their parametrization shows all the necessary steps that emerge while building the mathematical description of quantum mechanical manifolds. The set of all manifolds implemented in QGOpt library is listed below.

- The complex Stiefel manifold $\mathrm{St}_{n,p} = \left\{ V \in \mathbb{C}^{n \times p} | V^\dagger V = \mathbb{1} \right\}$ is a set of all isometric matrices of fixed size. A particular case of this manifold is a set of all unitary matrices of fixed size; therefore, this manifold can be used for different tasks related to quantum control. Some architectures of tensor networks may include isometric matrices as building blocks [41, 42]; thus, one can use this manifold to optimize such tensor networks.

- The manifold of density matrices of fixed rank $\varrho_{n,r} = \left\{ \varrho \in \mathbb{C}^{n \times n} \middle| \varrho = \varrho^\dagger, \ \mathrm{Tr}(\varrho) = 1, \ \varrho \geq 0, \ \mathrm{rank}(\varrho) = r \right\}$ is a set of all fixed-rank Hermitian positive semi-definite matrices with unit trace. Since density matrices represent states of quantum systems, one can use this manifold to perform state tomography and optimization of initial quantum states in different quantum circuits. This manifold is implemented through a parametrization with a quotient structure on top of it.

- The manifold of Choi matrices of fixed rank $C_{n,r} = \left\{ C \in \mathbb{C}^{n^2 \times n^2} \middle| C = C^\dagger, \ \mathrm{Tr}_p(C) = \mathbb{1}, \ C \geq 0, \ \mathrm{rank}(C) = r \right\}$ is a set of all fixed-rank Hermitian positive semi-definite matrices with auxiliary linear constraint (equality of the partial trace to the identity matrix). Choi matrices are used as representations of quantum channels; hence, one may use this manifold to perform quantum channel tomography and optimization of quantum channels in different quantum circuits. This

manifold is implemented through a parametrization with a quotient structure on top of it.

- The manifold of Hermitian matrices $H_n = \left\{ H \in \mathbb{C}^{n \times n} \big| H = H^\dagger \right\}$ is a linear subspace of a space $\mathbb{C}^{n \times n}$. Since Hermitian matrices represent measurable physical operators in the quantum theory, one can use this manifold to perform a search of optimal measurable physical operators in different problems.

- The manifold of Hermitian positive definite matrices $\mathbb{S}_{++}^n = \left\{ S \in \mathbb{C}^{n \times n} \big| S = S^\dagger, \, S \succ 0 \right\}$ is a set of all positive definite matrices of fixed size. One can use it to search the optimal non-normalized quantum state in different tasks.

- The manifold of positive operator-valued measures (POVMs) with full rank elements $\text{POVM}_{m,n} = \left\{ \{E_i\}_{i=1}^m \in \mathbb{C}^{m \times n \times n} \big| E_i = E_i^\dagger, \, E_i \geq 0, \, \sum_{i=1}^m E_i = \mathbb{1}, \, \text{rank}(E_i) = n \right\}$ can be considered as a tensor with Hermitian positive semi-definite full-rank slices that sum into the identity matrix. Since POVMs describe generalized measurements in quantum theory, one can use this manifold to perform a search of optimal measurements that give the largest information gain. This manifold is implemented through a parametrization with a quotient structure on top of it.

Mathematical details of the implementation of manifolds are given in Appendix A.

4 QGOpt API

4.1 Manifolds API

In this section we discuss the API of the version 1.0.0 of the QGOpt library. The central class of the QGOpt library is the manifold base class. All particular manifold types are inherited from the manifold base class. All manifold subclasses admit working with the direct product of several manifolds. Optimization primitives of each particular manifold are implemented as methods of the corresponding class describing a manifold. This list of methods allows one not to pay particular attention to the details of the underlying Riemannian geometry.

Let us consider basic illustrative examples. First, one needs to import all necessary libraries and create an example of a manifold. As an example we consider the complex Stiefel manifold.

```
import QGOpt as qgo
import tensorflow as tf

# example of complex Stiefel manifold
m = qgo.manifolds.StiefelManifold()
```

Here, m is an example of the complex Stiefel manifold that contains all the necessary information on the manifold's geometry. Some manifolds allow one to specify a type of metric and retraction as well. Using this example of a manifold one can sample a random point from a manifold:

```
u = m.random((4, 3, 2))
```

Here, we sample a random tensor u, that is a complex valued TensorFlow tensor of size $4 \times 3 \times 2$. This tensor represents a point from the direct product of four complex Stiefel manifolds. The first index of this tensor enumerates a manifold and the last two indices are matrix indices. Therefore, the tensor u can be seen as a set of four isometric matrices. One can generate a random tangent vector drawn from u.

```
v = m.random_tangent(u)
```

Here, v is a complex-valued TensorFlow tensor of the same size and type as u, and represents the random tangent vector drawn from u. Now let us assume that we have a random vector w which is of the same size and type, but is not tangent to u. One can make the orthogonal projection of this vector on the tangent space of u:

```
w = m.proj(u, w)
```

The updated vector w is an element of the tangent space of u now. The projection method of quotient manifold performs the projection on the horizontal space. To get the scalar product of two tangent vectors one can use the following line of code:

```
wv_inner = m.inner(u, w, v)
```

Here we pass u to the inner product method to specify the tangent space where we compute the inner product, because in Riemannian geometry the metric and inner product are point-dependent in general.

To implement first-order Riemannian optimization methods on a manifold one needs to be able to move points and vectors along the manifold. There are retraction and vector transport methods for this purpose. As an example let us move a point u along a tangent vector v via the retraction map:

```
u_tilde = m.retraction(u, v)
```

The new point $\tilde{u}$ is the result of transportation of u along vector v. To perform transportation of a vector along some other vector one can run:

```
v_tilde = m.vector_transport(u, v, w)
```

Here we start from point u and transport a tangent vector v along a tangent vector w, and obtain $\tilde{v}$ that is the result of the vector transportation.

The last important method converts the Euclidean gradient of a function to the Riemannian gradient. The Riemannian gradient replaces the Euclidean gradient to take into account the metric of a manifold and the tangent space in a given point. To calculate the Riemannian gradient one can use:

```
r = m.egrad_to_rgrad(u, e)
```

where we denote the Euclidean gradient as e and the Riemannian gradient as r.

The numerical complexity of each optimization primitive varies from one manifold to another. The complexity of all primitives is summarized in Appendix B.

4.2 Optimizers

The Riemannian optimizers implemented in QGOpt are inherited from TensorFlow optimizers and hence have the same API. The main difference is that one should also pass an example of manifold while defining an optimizer, which guides the optimizer and preserves the manifold's constraints. Two optimizers are implemented, that are among the most popular in machine learning: Riemannian versions of Adam [18] and SGD [43].

If m is a manifold element and lr is a learning rate (optimization step size), then the Adam and SGD optimizers can be initialized as:

```
1 # Riemannian ADAM optimizer
2 opt = qgo.optimizers.RAdam(m, lr)
3 # Riemannian SGD optimizer
4 opt = qgo.optimizers.RSGD(m, lr)
```

Note that some other attributes like the momentum value of the SGD optimizer or the AMSGrad modification of the Adam optimizer can also be specified.

4.3 Auxiliary functions

It is important to keep in mind that TensorFlow optimizers work well only with real variables. Therefore, one cannot use complex variables to represent a point on a manifold because they are being tuned while optimizing. The simplest way of representing a point from a complex manifold through real tensors is by introducing an additional index that enumerates real and imaginary parts of a tensor. For example a complex-valued tensor of shape (a, b, c) can be represented as a real-valued tensor of shape $(a, b, c, 2)$. During calculations, we need to convert tensors from their real representation to their complex representation and back.

Let us initialize a complex-valued tensor as a point from a manifold using the method "random". In order to make this tensor a variable suitable for an optimizer, one needs to convert it to the real representation. Then, while building a computational graph, one may need to have a complex form of a tensor again:

```
# a random real tensor, last index enumerates
# real and imaginary parts
w = tf.random.normal((4, 3, 2),
                     dtype=tf.float64)
# corresponding complex tensor of shape (4, 3)
wc = qgo.manifolds.real_to_complex(w)
# corresponding real tensor (wr = w)
wr = qgo.manifolds.complex_to_real(wc)
```

5 Examples of application of QGOpt

5.1 Quantum gate decomposition

In this subsection we consider an illustrative example of quantum gate decomposition. It is known that any two qubit-quantum gate U can be decomposed [44]:

$$U = [\tilde{u}_{11} \otimes \tilde{u}_{12}]U_{\text{CNOT}}[\tilde{u}_{21} \otimes \tilde{u}_{22}]U_{\text{CNOT}}[\tilde{u}_{31} \otimes \tilde{u}_{32}]U_{\text{CNOT}}[\tilde{u}_{41} \otimes \tilde{u}_{42}], \qquad (10)$$

where U_{CNOT} is the CNOT gate and $\{\tilde{u}_{ij}\}_{i,j=1}^{4,2}$ is a set of unknown one qubit-gates. Since a set $\{\tilde{u}_{ij}\}_{i,j=1}^{4,2}$ can be seen as the direct product of 8 complex Stiefel manifolds, one can use Riemannian optimization methods to find all $\tilde{u}_{ij}$. First, we initialize randomly a trial set $\{u_{ij}\}_{i,j=1}^{4,2}$ that will be tuned by Riemannian optimization methods. For simplicity, we denote the decomposition introduced above as

$$D(u_{ij}) = [u_{11} \otimes u_{12}]U_{\text{CNOT}}[u_{21} \otimes u_{22}]U_{\text{CNOT}}[u_{31} \otimes u_{32}]U_{\text{CNOT}}[u_{41} \otimes u_{42}]. \qquad (11)$$

The optimal set of one-qubit gates can be expressed as:

$$\underset{\{u_{ij}\}_{i,j=1}^{4,2}}{\arg\min} \|U - D(u_{ij})\|_F, \qquad (12)$$

where each u_{ij} obeys the unitarity constraint and $\|\cdot\|_F$ is the Frobenius norm.

Before considering the main part of the code that solves the above problem, we need to introduce a function that calculates the Kronecker product of two matrices:

```
def kron(A, B):
    AB = tf.tensordot(A, B, axes=0)
    AB = tf.transpose(AB, (0, 2, 1, 3))
    AB = tf.reshape(AB, (A.shape[0]*B.shape[0],
                         A.shape[1]*B.shape[1]))
    return AB
```

Now, we define an example of the complex Stiefel manifold:

```
m = qgo.manifolds.StiefelManifold()
```

We use a randomly generated target gate that we want to decompose,

```
U = m.random((4, 4), dtype=tf.complex128)
```

We initialize the initial set $\{u_{ij}\}_{i,j=1}^{4,2}$ randomly as a rank-4 tensor,

```
u = m.random((4, 2, 2, 2), dtype=tf.complex128)
```

The first two indices of this tensor enumerate a particular one-qubit gate, the last two indices are matrix indices of a gate. We turn this tensor into its real representation in order to make it suitable for an optimizer and wrap it up into the TensorFlow variable:

```
u = qgo.manifolds.complex_to_real(u)
u = tf.Variable(u)
```

We initialize the CNOT gate U_{CNOT} as follows:

```
1 cnot = tf.constant([[1, 0, 0, 0],
2                      [0, 1, 0, 0],
3                      [0, 0, 0, 1],
4                      [0, 0, 1, 0]],
5                      dtype=tf.complex128)
```

As the next step, we initialize the Riemannian Adam optimizer:

```
lr = 0.2  # optimization step size
opt = qgo.optimizers.RAdam(m, lr)
```

and run the forward pass of computations:

```
1 with tf.GradientTape() as tape:
2     # turning u back into its
3     # complex representation
4     uc = qgo.manifolds.real_to_complex(u)
5     # decomposition
6     D = kron(uc[0, 0], uc[0, 1])
7     D = cnot @ D
8     D = kron(uc[1, 0], uc[1, 1]) @ D
9     D = cnot @ D
10    D = kron(uc[2, 0], uc[2, 1]) @ D
11    D = cnot @ D
12    D = kron(uc[3, 0], uc[3, 1]) @ D
13    # loss function
14    L = tf.linalg.norm(D - U) ** 2
15    # is equivalent to casting to a real dtype
16    L = tf.math.real(L)
```

The final step is to minimize the loss function $L = \|D(u_{ij}) - U\|_F^2$ calculated during the previous step. We calculate the gradient of L, using automatic differentiation, with respect to the set $\{u_{ij}\}_{i,j=1}^{4,2}$:

```
grad = tape.gradient(L, u)
```

and pass the gradient to the optimizer:

```
opt.apply_gradients(zip([grad], [u]))
```

The Adam optimizer performs one optimization step keeping the orthogonality constraints. We repeat the forward pass, gradient calculation and optimization steps several times, wrapping them into a for loop until convergence and end up with a proper decomposition of the gate U. The optimization result is given in Fig. 6. One can see that at the end of the optimization process, the error is completely negligible. This section in the form of a tutorial is available in the QGOpt online [45].
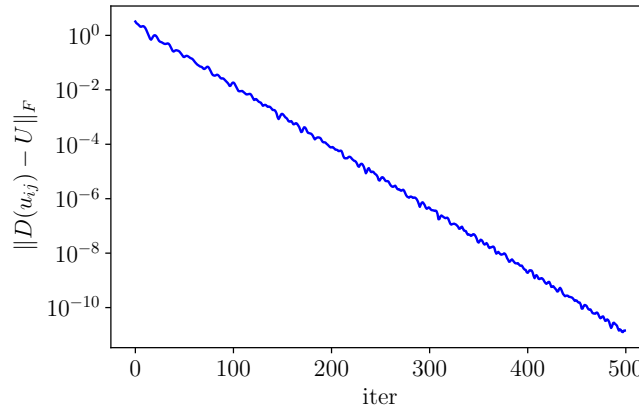
Figure 6: Frobenius distance between a gate and its decomposition. One can see that the distance rapidly decreases with the number of iteration towards nearly zero within machine precision.

5.2 Quantum tomography

Another typical problem that can be addressed by Riemannian optimization is the quantum tomography of states [46, 47] and channels [48, 49]. Here, we consider an example of quantum tomography of channels because it involves a more complicated structure than quantum tomography of states.

Let $\mathcal{H} = \bigotimes_{i=1}^{n} \mathbb{C}^2$ be the Hilbert space of a system consisting of n qubits. Let us assume that one has a set of input states $\{\rho_i\}_{i=1}^{N}$, where N is a total number of states, and each ρ_i is a density matrix on $\mathcal{H}$. One passes initial states through an unknown quantum channel Φ_{true} and observes a set of measurement outcomes $\left\{ M_{k_i^1}^{\text{tetra}} \otimes \cdots \otimes M_{k_i^n}^{\text{tetra}} \right\}_{i=1}^{N}$, where M_k^{tetra} is an element of a tetrahedral POVM [50]:

$$M_k^{\text{tetra}} = \frac{1}{4} \left(\mathbb{1} + s_k^T \boldsymbol{\sigma} \right), \ k \in (0, 1, 2, 3), \tag{13}$$

$$\boldsymbol{\sigma} = \left(\sigma_x, \sigma_y, \sigma_z \right), s_0 = (0, 0, 1), s_1 = \left(\frac{2\sqrt{2}}{3}, 0, -\frac{1}{3} \right),$$

$$s_2 = \left(-\frac{\sqrt{2}}{3}, \sqrt{\frac{2}{3}}, -\frac{1}{3} \right), s_3 = \left(-\frac{\sqrt{2}}{3}, -\sqrt{\frac{2}{3}}, -\frac{1}{3} \right).$$

One can estimate an unknown channel by maximizing the logarithmic likelihood of measurement outcomes:

$$\underset{\Phi \text{ is CPTP}}{\text{argmax}} \sum_{i=1}^{N} \log \text{Tr} \left(M_{k_i^1}^{\text{tetra}} \otimes \cdots \otimes M_{k_i^n}^{\text{tetra}} \Phi(\rho_i) \right). \tag{14}$$

For simplicity, we assume that the many-body tetrahedral POVM M is already predefined and has the shape $(2^{2n}, 2^n, 2^n)$, where the first index enumerates the POVM element. We also assume that we have a data set that consists of a set of initial density matrices of shape $(N, 2^n, 2^n)$ and a set of POVM elements of the same shape that came true after measurements. In our experiments, the unknown channel has Kraus rank 2 and is generated randomly, the initial density matrices are pure and also generated randomly.

Let us proceed with the practical implementation. First, we define an example of the quotient manifold equivalent to the manifold of Choi matrices:

```
m = qgo.manifolds.ChoiMatrix()
```

Elements of this manifold are connected with Choi matrices via the relation (6). We randomly initialize a point from the quotient manifold,

```
1  # random initial parametrization
2  A = m.random((2**(2*n), 2**(2*n)),
3  dtype=tf.complex128)
4  # variable should be real
5  # to make an optimizer work correctly
6  A = qgo.manifolds.complex_to_real(A)
7  # variable
8  A = tf.Variable(A)
```

Then we initialize the Riemannian Adam optimizer:

```
lr = 0.07
opt = qgo.optimizers.RAdam(m, lr)
```

and calculate the logarithmic likelihood function:

```
1  with tf.GradientTape() as tape:
2      # Ac is a complex representation of A
3      # shape=(2**2n, 2**2n)
4      Ac = qgo.manifolds.real_to_complex(A)
5
6      # reshape parametrization
7      # (2**2n, 2**2n) --> (2**n, 2**n, 2**2n)
8      Ac = tf.reshape(Ac, (2**n, 2**n, 2**(2*n)))
9
10     # Choi tensor (reshaped Choi matrix)
11     choi = tf.tensordot(Ac,
12                         tf.math.conj(Ac),
13                         [[2], [2]])
14
15     # turning Choi tensor to the
16     # corresponding quantum channel
17     phi = tf.transpose(choi, (1, 3, 0, 2))
18     phi = tf.reshape(phi, (2**(2*n), 2**(2*n)))
19
20     # reshape initial density
21     # matrices to vectors
22     rho_resh = tf.reshape(rho_in, (N, 2**(2*n)))
23
24     # passing density matrices
25     # through a quantum channel
26     rho_out = tf.tensordot(phi,
27                            rho_resh,
28                            [[1], [1]])
29     rho_out = tf.transpose(rho_out)
30     rho_out = tf.reshape(rho_out,
31                          (N, 2**n, 2**n))
32
33     # probabilities of measurement outcomes
34     # (povms is a set of POVM elements
35     # came true of shape (N, 2**n, 2**n))
36     p = tf.linalg.trace(povms @ rho_out)
37
38     # negative log likelihood (to be minimized)
39     L = -tf.reduce_mean(tf.math.log(p))
```

The complexity of the code above can be reduced by choosing the optimal order of tensor contraction; however, it becomes more complicated in this case, and is not suitable for the tutorial. Finally, we calculate the logarithmic likelihood gradient with respect to the parametrization of the Choi matrix:
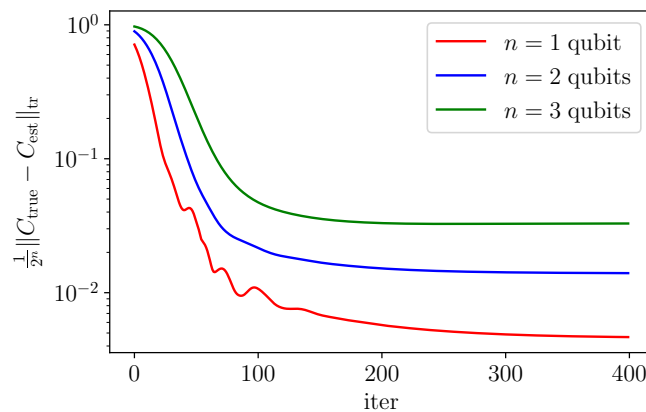
Figure 7: Dependence between Jamiołkowski process distance and number of iteration. Number of measurement outcomes $N = 600000$ for all experiments.

```
grad = tape.gradient(L, A)
```

and apply the optimizer to make an optimization step that does not violate the CPTP constraints:

```
opt.apply_gradients(zip([grad], [A]))
```

We repeat the calculation of the logarithmic likelihood function, gradient calculation and optimization steps several times, wrapping them into a for loop, until convergence is reached. To evaluate the quality of an unknown quantum channel estimation, we calculate the Jamiołkowski process distance [51]:

$$J(\Phi_{\text{true}}, \Phi_{\text{est}}) = \frac{1}{2^n} \|C_{\text{true}} - C_{\text{est}}\|_{\text{tr}}, \tag{15}$$

where $\Phi_{\text{true}}(\Phi_{\text{est}})$ is the true (estimated) quantum channel, $C_{\text{true}}(C_{\text{est}})$ is the corresponding Choi matrix, $\|\cdot\|_{\text{tr}}$ is the trace norm and $0 \leq J(\Phi_{\text{true}}, \Phi_{\text{est}}) \leq 1$. One can see in Fig. 7 that the Jamiołkowski process distance converges to some small value with the number of iterations and we end up with a reasonable estimation of an unknown quantum channel. This section is available in the QGOpt online documentation in the form of a tutorial [45].

6 Optimization over an arbitrary Cartesian product of manifolds

In general, it is possible to perform optimization over the Cartesian product of different manifolds. The QGOpt library allows finding

$$\underset{A \in \mathcal{M}}{\text{argmin}} f(A), \tag{16}$$

where $\mathcal{M}$ is an arbitrary Cartesian product of manifolds, implemented in the QGOpt library, f is a function that can be evaluated within the TensorFlow framework.

As an example, let us consider an optimization over the manifold

$$\mathcal{M} = \varrho_{n_1, n_1} \times \varrho_{n_2, n_2} \times \varrho_{n_2, n_2} \times C_{n,r} \times C_{n,r}, \tag{17}$$

where $\times$ denotes the Cartesian product. In other words, one has one manifold of full-rank density matrices of size $n_1 \times n_1$, two manifolds of full-rank density matrices of size $n_2 \times n_2$ and two manifolds of Choi matrices of size $n^2 \times n^2$ and rank r. Let us define the building blocks of $\mathcal{M}$,

```
m_choi = qgo.manifolds.ChoiMatrix()
m_dens = qgo.manifolds.DensityMatrix()
```

The next step is to define variables representing points on manifolds. First, we define a variable representing a point in ϱ_{n_1,n_1}

```
1  # random initialization
2  A_rho_1 = m_dens.random((n1, n1),
3                          dtype=tf.complex128)
4  # variable should be real
5  # to make an optimizer work correctly
6  A_rho_1 = qgo.manifolds.complex_to_real(A_rho_1)
7  # variable
8  A_rho_1 = tf.Variable(A_rho_1)
```

Then we define a variable representing a point from $\varrho_{n_2,n_2} \times \varrho_{n_2,n_2}$

```
1  # random initialization
2  A_rho_2 = m_dens.random((2, n2, n2),
3                          dtype=tf.complex128)
4  # variable should be real
5  # to make an optimizer work correctly
6  A_rho_2 = qgo.manifolds.complex_to_real(A_rho_2)
7  # variable
8  A_rho_2 = tf.Variable(A_rho_2)
```

where we take advantage of the fact that both matrices are of the same size, and we can represent them as one tensor. Let us group these two variables into one list

```
A_rho = [A_rho_1, A_rho_2]
```

which is passed to the Riemannian optimizer on the manifold of density matrices. One also needs to define a variable representing a point from $C_{n,r}$.

```
1  # random initialization
2  A_choi = m_choi.random((n**2, r))
3  # variable should be real
4  # to make an optimizer work correctly
5  A_choi = qgo.manifolds.complex_to_real(A_choi)
6  # variable
7  A_choi = tf.Variable(A_choi)
```

Now one needs to define optimizers

```
1  # learning rate
2  lr = 0.01
3  # optimizer over density matrices
4  opt_dens = qgo.optimizers.RAdam(m_dens, lr)
5  # optimizer over choi matrices
6  opt_choi = qgo.optimizers.RAdam(m_choi, lr)
```

and perform an optimization step

```
1  with tf.GradientTape() as tape:
2      L = f(A_rho_1, A_rho_2, A_choi)
3  # gradient over all variables
4  grad_total = tape.gradient(L, A_rho + [A_choi])
5  # gradient over variables representing
6  # density matrices
7  grad_rho = grad_total[:2]
8  # gradient over the variable representing
9  # Choi matrix
10 grad_choi = grad_total[-1]
11 # optimization step
12 opt_dens.apply_gradients(zip(grad_rho, A_rho))
13 opt_choi.apply_gradients(zip([grad_choi],
14                              [A_choi]))
```

where we assume that the function f is predefined. In order to iterate optimization steps until convergence, one can wrap the code above into a loop. This general scheme can be used for optimization over an arbitrary set of different manifolds.

7 Discussion and concluding remarks

The range of application of the QGOpt library to different problems of quantum technologies is not limited to quantum gate decomposition and quantum tomography. The six manifolds implemented in QGOpt give rise to different interesting scenarios of constrained optimization usage in quantum technologies. For example, the complex Stiefel manifold can be used to address different control problems [52–54] where one needs to find an optimal set of unitary gates driving a quantum system to a desirable quantum state. It is also possible to use a complex Stiefel manifold to perform entanglement renormalization [41, 42], machine learning by unitary tensor networks [55] or non-Markovian quantum dynamics identification [24]. In addition, quotient manifolds for quantum tomography, or for density matrices and Choi matrices can be used to maintain natural quantum constraints in different tensor network architectures. Quotient manifold of POVMs can be used for searching for an optimal generalized measurement scheme with maximum information gain. Finally, all these manifolds can be combined in one optimization task, which allows to address multi-component problems.

Although as of yet the QGOpt library includes only first-order optimization methods, we plan to extend the list of optimizers by including quasi-Newton methods such as the Riemannian BFGS [56], and the recently developed quantum natural gradient descent [17] generalized to the case of embedded and quotient manifolds.

To conclude, we have introduced the QGOpt library aimed at solving constrained optimization problems with natural quantum constraints. We have introduced and discussed abstract concepts such as quotient manifolds under the hood of QGOpt. We have gone through the QGOpt API and covered its most important features. We also sorted out examples of codes solving illustrative quantum technology problems.

Acknowledgements

A Underlying geometry of manifolds implemented in the QGOpt library

In this appendix, we consider some mathematical aspects of the implementation of manifolds in the QGOpt library. First, we discuss how one can identify complex matrices, which are elements of all manifolds implemented in the QGOpt library, with real matrices. Any complex

matrix A can be represented as follows

$$\tilde{A} = \begin{bmatrix} \mathrm{Re}(A) & \mathrm{Im}(A) \\ -\mathrm{Im}(A) & \mathrm{Re}(A) \end{bmatrix}. \tag{18}$$

The following correspondences between operations with complex matrices and operations with their real representations

$$A^\dagger \to \tilde{A}^T, \; AB \to \tilde{A}\tilde{B}, \; A + B \to \tilde{A} + \tilde{B}, \; 2\mathrm{Re}(\mathrm{Tr}(A)) = \mathrm{Tr}(\tilde{A}), \tag{19}$$

allow us to work with certain sets of complex matrices as with Riemannian manifolds of real matrices [57]. The QGOpt library contains six manifolds: three implemented as embedded manifolds and three implemented as quotient manifolds.

Table 1 summarizes the geometry lying under the hood of a high-level description of the embedded manifolds in the QGOpt library.

We also summarize the geometry of the manifolds that are implemented as quotient manifolds. In our summary, we follow the book by Nicolas Boumal [34], which provides a very instructive presentation of the optimization on quotient manifolds.

Having an optimization problem on a quotient manifold, one works with two sets $\overline{\mathcal{M}}$ and $\mathcal{M}$ that are connected as

$$\mathcal{M} = \overline{\mathcal{M}}/\sim = \{[x]|x \in \overline{\mathcal{M}}\}, \tag{20}$$

where $[x] = \{y|y \in \overline{\mathcal{M}}, \; y \sim x\}$ is the equivalence class of x, $\overline{\mathcal{M}}$ is some Riemannian manifold and $\mathcal{M}$ is its quotient. We call a map π a canonical projection if it maps any x from $\overline{\mathcal{M}}$ to its equivalence class:

$$\pi(x) = [x]. \tag{21}$$

For $\mathcal{M}$ to be a manifold, one requires π to be smooth and its differential $D\pi(x): T_x\overline{\mathcal{M}} \to T_{[x]}\mathcal{M}$ must have a constant rank $r = \dim(\mathcal{M})$ for all $x \in \overline{\mathcal{M}}$. We call V_x a vertical space at $x \in \overline{\mathcal{M}}$ if it is the kernel of $D\pi(x)$, i.e.

$$V_x = \ker(D\pi(x)), \tag{22}$$

then one can decompose the tangent space $T_x\overline{\mathcal{M}}$ at a point x as

$$T_x\overline{\mathcal{M}} = H_x \oplus V_x, \tag{23}$$

where H_x is the orthogonal complement of V_x also called the horizontal space. The restricted linear map $D\pi(x)|_{H_x}: H_x \to T_{[x]}\mathcal{M}$ is bijective by construction and can be used to represent a vector from $T_{[x]}\mathcal{M}$ as a vector from H_x. This representation is called horizontal lift and reads

$$v = (D\pi(x)|_{H_x})^{-1}[\xi] = \mathrm{lift}_x(\xi), \tag{24}$$

where ξ is a vector from $T_{[x]}\mathcal{M}$ and v is its representation from H_x.

Having introduced all the objects above, one can try to construct all primitives for optimization algorithms on a quotient manifold through the same primitives on a total manifold (see table 2).

These primitives are correct if they do not depend on a choice of a particular point from an equivalence class, i.e. for all $[x] \in \mathcal{M}$ and ξ, $\zeta \in T_{[x]}\mathcal{M}$ if $x \sim y$ the following statements are true

$$\langle \mathrm{lift}_x(\xi), \mathrm{lift}_x(\zeta) \rangle_x = \langle \mathrm{lift}_y(\xi), \mathrm{lift}_y(\zeta) \rangle_y, \tag{25}$$

$$\pi\left(\overline{R}_x(\mathrm{lift}_x(\xi))\right) = \pi\left(\overline{R}_y(\mathrm{lift}_y(\xi))\right), \tag{26}$$

Table 1: Summary of the geometry of embedded manifolds implemented in the QGOpt library.

Manifold	Description	Inner product	Riemannian gradient	Retraction	Vector transport
Complex Stiefel manifold $\text{St}_{n,p}$	Embedded manifold of complex isometric matrices.	Two types of inner product [40, 41], induced by embedding, are available in the QGOpt library: Eucledian inner product $\langle v, w \rangle_u = \text{Re}(\text{Tr}(v^\dagger w))$, and canonical inner product $\langle v, w \rangle_u = \text{Re}\left(\text{Tr}\left(v^\dagger\left(I - \frac{1}{2}uu^\dagger\right)w\right)\right)$. These two types of inner product induce the same orthogonal projection [40].	The Riemannian gradient for the Euclidean inner product takes the following form [40] $\nabla_R f(u) = \frac{1}{2}u(u^\dagger\nabla f(u) - \nabla f(u)^\dagger u) + (I - uu^\dagger)\nabla f(u)$, the Riemannian gradient for the canonical inner product takes the following form [40] $\nabla_R f(u) = \nabla f(u) - u\nabla f(u)^\dagger u$.	Three types of retraction are available in the QGOpt library: SVD decomposition based retraction [35], QR decomposition based retraction [35] and Cayley retraction [33,35].	Vector transport is induced by a retraction. It is implemented as the orthogonal projection of a vector on the tangent space of a point obtained via a retraction [33,35].
Manifold of Hermitian positive definite matrices $\mathbb{S}_{++}^n$	Embedded manifold of complex Hermitian, positive definite matrices. Two different inner products are introduced in a way that the manifold is complete. Inner products are not extended on the ambient space, which does not allow the orthogonal projection on the tangent space of a point. However, for this manifold we do not use the orthogonal projection at all.	Two types of inner product are available in the QGOpt library: Log–Cholesky inner product [41, 58] and Log–Euclidean inner product [41]. Both inner products keep the manifold complete.	The Riemannian gradient for both inner products that are used in the QGOpt library is derived in [41].	Instead of a retraction, one uses the exponential map for both inner products in the QGOpt library. Closed form of the exponential map for the Log–Euclidean inner product can be found in [41], for the Log–Cholesky inner product in [41,58].	Instead of a vector transport, one uses the parallel transport for both inner products in the QGOpt library. The closed form of the parallel transport for the Log–Euclidean metric can be found in [41], for the Log–Cholesky metric in [41,58].
Manifold of Hermitian matrices H_n.	Embedded manifold of Hermitian matrices that also is a linear subspace of the ambient space.	Only the Euclidean inner product is available in the QGOpt library; it reads $\langle v, w \rangle_u = \text{Re}(\text{Tr}(v^\dagger w))$. The inner product is induced by the embedding.	The Riemannian gradient for the Euclidean inner product takes the following form $\nabla_R f(u) = \frac{1}{2}\left(\nabla f(u) + \nabla f(u)^\dagger\right)$.	Instead of a retraction, one uses the exponential map, which is a trivial transportation along a straight line.	Instead of a vector transport, one uses the parallel transport, which is the identity transformation.

Table 2: Optimization primitives of $\mathcal{M}$ expressed through optimization primitives of $\overline{\mathcal{M}}$.

Inner product	$\langle \xi, \zeta \rangle_{[x]} = \langle \mathrm{lift}_x(\xi), \mathrm{lift}_x(\zeta) \rangle_x$, where $\langle \cdot, \cdot \rangle_x$ is an inner product in $T_x \overline{\mathcal{M}}$ and $\xi, \zeta \in T_{[x]} \mathcal{M}$
Retraction	$R_{[x]}(\xi) = \pi \left(\overline{R}_x(\mathrm{lift}_x(\xi)) \right)$, where $\overline{R}$ is an retraction on $\overline{\mathcal{M}}$
Vector transport	$\tau_{[x],\xi}(\zeta) = \mathrm{lift}^{-1}_{\overline{R}_x(\mathrm{lift}_x(\xi))} \left(P_{H_{\overline{R}_x(\mathrm{lift}_x(\xi))}}(\mathrm{lift}_x(\zeta)) \right)$, where P_S is the orthogonal projection operator on a subspace S
Function	$\overline{f} = f \circ \pi$, where $f : \mathcal{M} \to \mathbb{R}$ and $\overline{f} : \overline{\mathcal{M}} \to \mathbb{R}$
Riemannian gradient	$\nabla_R f([x]) = \mathrm{lift}^{-1}_x \left(\nabla_R \overline{f}(x) \right)$

$$\mathrm{lift}^{-1}_{\overline{R}_x(\mathrm{lift}_x(\xi))} \left(P_{H_{\overline{R}_x(\mathrm{lift}_x(\xi))}}(\mathrm{lift}_x(\zeta)) \right) = \mathrm{lift}^{-1}_{\overline{R}_y(\mathrm{lift}_y(\xi))} \left(P_{H_{\overline{R}_y(\mathrm{lift}_y(\xi))}}(\mathrm{lift}_y(\zeta)) \right), \tag{27}$$

where $\sim$ denotes equivalence relation between elements of $\overline{\mathcal{M}}$. It is also worth noting that in practice there is no need to go back from $\overline{\mathcal{M}}$ to $\mathcal{M}$ after application of each primitive. Instead, one can work only with objects from $\overline{\mathcal{M}}$, which makes optimization algorithms on $\mathcal{M}$ almost identical to algorithms on $\overline{\mathcal{M}}$.

Manifolds $\varrho_{n,r}$, $C_{n,r}$ and $\mathrm{POVM}_{m,n}$ in the QGOpt library are implemented using the above idea. The quotient geometry of the real version of the manifold $\varrho_{n,r}$ is described in [59] and also is implemented in the Manopt library [60]. The alternative approach to optimization on a $\mathrm{POVM}_{m,n}$ is also considered in [61] and implemented in Manopt. To the best of the authors' knowledge, the manifold $C_{n,r}$ has not been considered from the Riemannian optimization point of view.

Let us consider total manifolds that are used to build quotient manifolds implemented in the QGOpt library. They are

$$\overline{\varrho}_{n,r} = \left\{ A \in \mathbb{C}^{n \times r}_* \Big| \mathrm{Tr}(AA^\dagger) = 1 \right\}, \tag{28}$$

$$\overline{C}_{n,r} = \left\{ A \in \mathbb{C}^{n^2 \times r}_* \Big| \mathrm{Tr}_p \left(AA^\dagger \right) = \mathbb{1} \right\}, \tag{29}$$

$$\overline{\mathrm{POVM}}_{m,n} = \left\{ \{A_i\}_{i=1}^m \Big| \sum_{i=1}^m A_i A_i^\dagger = \mathbb{1}, \ A_i \in \mathbb{C}^{n \times n}_* \right\}, \tag{30}$$

where $\mathbb{C}^{p \times q}_*$ is the set of complex full-rank matrices of size $p \times q$. One can note that the manifold $\overline{\varrho}_{n,r}$ is a sphere with the additional condition on the rank of A, manifolds $\overline{C}_{n,r}$ and $\overline{\mathrm{POVM}}_{m,n}$ are complex Stiefel manifolds with the additional condition on the ranks of A and A_i. Any element of the total manifolds above corresponds to either a density matrix, a Choi matrix, or a POVM. Indeed

$$\varrho = AA^\dagger, \ \text{if } A \in \overline{\varrho}_{n,r}, \tag{31}$$

$$C = AA^\dagger, \ \text{if } A \in \overline{C}_{n,r}, \tag{32}$$

$$E_i = A_i A_i^\dagger, \ \text{if } A \in \overline{\mathrm{POVM}}_{m,n}, \tag{33}$$

where ϱ is some density matrix, C is some Choi matrix and E_i is an element of some POVM. However, there is an ambiguity:

$$\varrho = AA^\dagger = AQQ^\dagger A^\dagger, \tag{34}$$

$$C = AA^\dagger = AQQ^\dagger A^\dagger, \tag{35}$$

$$E_i = A_i A_i^\dagger = A_i Q_i Q_i^\dagger A_i^\dagger, \tag{36}$$

where Q is unitary and $\{Q_i\}_{i=1}^m$ is a set of unitary matrices of the appropriate size. In order to lift the ambiguity we introduce equivalence classes

$$[A] = \{AQ | Q \in \mathbb{C}^{r \times r}, \, QQ^\dagger = \mathbb{1}\}, \text{ for } \overline{\varrho}_{n,r} \text{ and } \overline{C}_{n,r}, \tag{37}$$

$$[\{A\}_{i=1}^m] = \{\{A_i Q_i\}_{i=1}^m | Q_i \in \mathbb{C}^{n \times n}, \, Q_i Q_i^\dagger = \mathbb{1}\}, \text{ for } \overline{\text{POVM}}_{m,n}, \tag{38}$$

and the corresponding quotient manifolds

$$\overline{\varrho}_{n,r} / \sim = \left\{ [A] \big| A \in \overline{\varrho}_{n,r} \right\}, \tag{39}$$

$$\overline{C}_{n,r} / \sim = \left\{ [A] \big| A \in \overline{C}_{n.r} \right\}, \tag{40}$$

$$\overline{\text{POVM}}_{m,n} / \sim = \left\{ [\{A_i\}_{i=1}^m] \big| \{A_i\}_{i=1}^m \in \overline{\text{POVM}}_{m,n} \right\}. \tag{41}$$

To identify quotient manifolds with those that are introduced in the main text, we introduce the maps

$$\phi_\varrho : \overline{\varrho}_{n,r} / \sim \to \varrho_{n,r} : [A] \mapsto AA^\dagger, \tag{42}$$

$$\phi_C : \overline{C}_{n,r} / \sim \to C_{n,r} : [A] \mapsto AA^\dagger, \tag{43}$$

$$\phi_{\text{POVM}} : \overline{\text{POVM}}_{n,m} / \sim \to \text{POVM}_{n,m} : [\{A\}_{i=1}^m] \mapsto \{A_i A_i^\dagger\}_{i=1}^m. \tag{44}$$

These three maps are bijections, which follows from Proposition 2.1 in [62] that is proved for real matrices, but generalization to the complex case is straightforward. They, as well as their inverses, are also differentiable which implies that these three maps are diffeomorphisms. It is enough to identify quotient manifolds with those introduced in the main text and turn to the optimization on quotient manifolds.

Now, to perform optimization on $\varrho_{n,r}$, $C_{n,r}$ and $\text{POVM}_{m,n}$ it is sufficient to introduce appropriate primitives for $\overline{\varrho}_{n,r}$, $\overline{C}_{n,r}$ and $\overline{\text{POVM}}_{m,n}$ that additionally satisfy Eqs. (25-27), and the projection on the horizontal space. The total manifolds equipped with the following inner products, induced by inner products of ambient spaces

$$\langle v, w \rangle_A = \text{Re}\left(\text{Tr}(vw^\dagger) \right), \text{ where } w, \, v \in T_A \overline{\varrho}_{n,r}, \tag{45}$$

$$\langle v, w \rangle_A = \text{Re}\left(\text{Tr}(vw^\dagger) \right), \text{ where } w, \, v \in T_A \overline{C}_{n,r}, \tag{46}$$

$$\langle \{v_i\}_{i=1}^m, \{w_i\}_{i=1}^m \rangle_A = \text{Re}\left(\sum_i \text{Tr}(v_i w_i^\dagger) \right), \tag{47}$$

where $\{w_i\}_{i=1}^m, \, \{v_i\}_{i=1}^m \in T_{\{A_i\}_{i=1}^m} \overline{\text{POVM}}_{m,n}$,

satisfy the condition (25) as shown in [34]. The projections on the horizontal space for the total manifolds are

$$P_{H_A}(v) = P_{T_A \overline{\varrho}_{n,r}}(v) - P_{V_A}(v), \text{ for } \overline{\varrho}_{n,r}, \tag{48}$$

$$P_{H_A}(v) = P_{T_A \overline{C}_{n,r}}(v) - P_{V_A}(v), \text{ for } \overline{C}_{n,r}, \tag{49}$$

$$P_{H_{\{A_i\}_{i=1}^m}}(\{v_i\}_{i=1}^m) = P_{T_{\{A_i\}_{i=1}^m} \overline{\text{POVM}}_{n,m}}(\{v_i\}_{i=1}^m) - P_{V_{\{A_i\}_{i=1}^m}}(\{v_i\}_{i=1}^m), \tag{50}$$

for $\overline{\text{POVM}}_{n,m}$,

Table 3: Complexity of optimization primitives for all manifolds implemented in the QGOpt library.

Manifold	Retraction	Vector transport	Riemannian gradient	Inner product	Projection
$C_{n,r}$	$O(n^3 r)$	$O(\max(n^2 r^2, n^3 r))$	$O(n^3 r)$	$O(n^2 r)$	$O(\max(n^2 r^2, n^3 r))$
$\text{St}_{n,p}$	For QR and SVD retractions $O(np^2)$, for Cayley retraction $O(n^3)$	$O(np^2)$	$O(np^2)$	For Euclidean metric $O(np)$, for canonical metric $O(np^2)$	$O(np^2)$
$\text{POVM}_{m,n}$	$O(mn^3)$	$O(mn^3)$	$O(mn^3)$	$O(mn^2)$	$O(mn^3)$
$\varrho_{n,r}$	$O(nr)$	$O(nr^2)$	$O(nr)$	$O(nr)$	$O(nr^2)$
H_n	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
$\mathbb{S}_{++}^n$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^2)$

where the projections on tangent spaces are known for the total manifolds that are the sphere and the complex Stiefel manifolds; and the projections on the vertical spaces can be found by solving the Sylvester equation [63]. One can introduce several different retractions for total manifolds that, however, may not satisfy the condition Eq. (26). Since manifolds $\overline{C}_{n,r}$ and $\overline{\text{POVM}}_{m,n}$ are complex Stiefel manifolds we can use SVD-based retraction for them. One can show that SVD-based retraction satisfies the condition Eq. (9) (see [34]). For the manifold $\overline{\varrho}_{n,r}$ one can use retraction on a sphere (see Example 4.1.1 in [35]). This retraction also satisfies the condition Eq. (26). Vector transports (see Table 2) induced by retractions above also satisfies the condition Eq. (27). The Riemannian gradients for $\overline{\varrho}_{n,r}$, $\overline{C}_{n,r}$ and $\overline{\text{POVM}}_{m,n}$ are known and can be used without modifications for optimization on quotient versions of these manifolds.

We thus have all optimization primitives for total manifolds, quotient manifolds, and equivalence between quotient manifolds and manifolds from the main text, which allows us to perform optimization on $\varrho_{n,r}$, $C_{n,r}$ and $\text{POVM}_{n,r}$.

B   Complexity of algorithms and comparison with other libraries

In this appendix, we discuss questions of scalability of optimization algorithms presented in the QGOpt library and compare the QGOpt library with other frameworks. To address the scalability of optimization algorithms, one needs to estimate the asymptotic complexity of primitives used in those algorithms. Table 3 shows the complexity of optimization primitives for all manifolds.

Let us compare the complexity of algorithms from the QGOpt library with some state of the art algorithms in quantum technologies. For example, let us consider quantum channel tomography, which can be implemented via optimization on $C_{n,r}$. Under the assumption that a particular algorithm uses all the optimization primitives, one step of an optimization algorithm scales like $O(n^3 r)$, where n is the dimension of a Hilbert space and r is Kraus rank (see Table 3). In general, the Kraus rank r is equal to n^2, which means that the maximal complexity is $O(n^5)$; however, if we have prior information that r is small, then one can significantly reduce the complexity of an algorithm. One can compare the one-step complexity of Riemannian-

optimization-based algorithms for quantum channel tomography with the one-step complexity of an algorithm suggested in [48] that is based on the orthogonal projection on the set of CPTP maps. In turn, the orthogonal projection on the set of CPTP maps is implemented through repeated averaged projections on CP and TP sets of maps. The projection on the CP set has complexity $O(n^6)$ that is larger than the complexity of Riemannian-optimization-based algorithms.

Let us also compare the QGOpt library with other libraries for Riemannian optimization. Table 4 shows a list of some related libraries. One can see that QGOpt suits best quantum technologies problems in terms of the number of quantum manifolds.

Table 4: Comparison of the QGOpt library with other libraries for Riemannian optimization in terms of the supported "quantum" manifolds.

Library	Language	Specific "quantum" manifolds
QGOpt	Python	Manifolds of density matrices, POVMs, Choi matrices, complex Stiefel manifold
Manopt	Matlab, Python, Julia	POVMs, complex Stiefel manifold
Geoopt	Python	None
mctorch	Python	None

References

[1] J. Toulouse, R. Assaraf and C. J. Umrigar, *Introduction to the variational and diffusion Monte Carlo methods*, in Advances in Quantum Chemistry, Elsevier, Amsterdam, Netherlands, ISBN 9780128030608 (2016), doi:10.1016/bs.aiq.2015.07.003.

[2] D. Ceperley, G. V. Chester and M. H. Kalos, *Monte Carlo simulation of a many-fermion study*, Phys. Rev. B **16**, 3081 (1977), doi:10.1103/PhysRevB.16.3081.

[3] D. Bressanini, G. Morosi and M. Mella, *Robust wave function optimization procedures in quantum Monte Carlo methods*, J. Chem. Phys. **116**, 5345 (2002), doi:10.1063/1.1455618.

[4] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Ann. Phys. **326**, 96 (2011), doi:10.1016/j.aop.2010.09.012.

[5] D. Perez-Garcia, F. Verstraete, M. M. Wolf and J. I. Cirac, *Matrix product state representations* (2006), arXiv:quant-ph/0608197.

[6] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Ann. Phys. **349**, 117 (2014), doi:10.1016/j.aop.2014.06.013.

[7] F. Verstraete, V. Murg and J. I. Cirac, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, Adv. Phys. **57**, 143 (2008), doi:10.1080/14789940801912366.

[8] F. Verstraete and J. I. Cirac, *Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions* (2004), arXiv:cond-mat/0407066.

[9] G. Carleo and M. Troyer, *Solving the quantum many-body problem with artificial neural networks*, Science **355**, 602 (2017), doi:10.1126/science.aag2302.

[10] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko and J. Carrasquilla, *Recurrent neural network wave functions*, Phys. Rev. Research **2**, 023358 (2020), doi:10.1103/PhysRevResearch.2.023358.

[11] K. Choo, T. Neupert and G. Carleo, *Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states*, Phys. Rev. B **100**, 125124 (2019), doi:10.1103/PhysRevB.100.125124.

[12] S. R. White, *Density matrix formulation for quantum renormalization groups*, Phys. Rev. Lett. **69**, 2863 (1992), doi:10.1103/PhysRevLett.69.2863.

[13] U. Schollwöck, *The density-matrix renormalization group*, Rev. Mod. Phys. **77**, 259 (2005), doi:10.1103/RevModPhys.77.259.

[14] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations*, Phys. Rev. Lett. **91**, 147902 (2003), doi:10.1103/PhysRevLett.91.147902.

[15] G. Vidal, *Efficient simulation of one-dimensional quantum many-body systems*, Phys. Rev. Lett. **93**, 040502 (2004), doi:10.1103/PhysRevLett.93.040502.

[16] R. Orús and G. Vidal, *Infinite time-evolving block decimation algorithm beyond unitary evolution*, Phys. Rev. B **78**, 155117 (2008), doi:10.1103/PhysRevB.78.155117.

[17] J. Stokes, J. Izaac, N. Killoran and G. Carleo, *Quantum natural gradient*, Quantum **4**, 269 (2020), doi:10.22331/q-2020-05-25-269.

[18] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2014), arXiv:1412.6980.

[19] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko and G. Carleo, *Neural-network quantum state tomography*, Nat. Phys. **14**, 447 (2018), doi:10.1038/s41567-018-0048-5.

[20] P. Cha, P. Ginsparg, F. Wu, J. Carrasquilla, P. L. McMahon and E.-A. Kim, *Attention-based quantum tomography* (2020), arXiv:2006.12469.

[21] J. Carrasquilla, G. Torlai, R. G. Melko and L. Aolita, *Reconstructing quantum states with generative models*, Nat. Mach. Intell. **1**, 155 (2019), doi:10.1038/s42256-019-0028-1.

[22] I. A. Luchnikov, A. Ryzhov, P.-J. Stas, S. N. Filippov and H. Ouerdane, *Variational autoencoder reconstruction of complex many-body physics*, Entropy **21**, 1091 (2019), doi:10.3390/e21111091.

[23] G. Torlai, C. J. Wood, A. Acharya, G. Carleo, J. Carrasquilla and L. Aolita, *Quantum process tomography with unsupervised learning and tensor networks* (2020), arXiv:2006.02424.

[24] I. A. Luchnikov, S. V. Vintskevich, D. A. Grigoriev and S. N. Filippov, *Machine learning non-Markovian quantum dynamics*, Phys. Rev. Lett. **124**, 140502 (2020), doi:10.1103/PhysRevLett.124.140502.

[25] L. Banchi, E. Grant, A. Rocchetto and S. Severini, *Modelling non-Markovian quantum processes with recurrent neural networks*, New J. Phys. **20**, 123030 (2018), doi:10.1088/1367-2630/aaf749.

[26] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization*, Phys. Rev. B **79**, 144108 (2009), doi:10.1103/PhysRevB.79.144108.

[27] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization: Boundaries, impurities and interfaces*, J. Stat. Phys. **157**, 931 (2014), doi:10.1007/s10955-014-0983-1.

[28] G. Vidal, *Entanglement renormalization*, Phys. Rev. Lett. **99**, 220405 (2007), doi:10.1103/PhysRevLett.99.220405.

[29] A. S. Holevo, *Quantum systems, channels, information*, Walter de Gruyter, Berlin, Germany, ISBN 9783110273250 (2012), doi:10.1515/9783110273403.

[30] I. Luchnikov, M. Krechetov and A. Ryzhov, *QGOpt GitHub repository* (2020), https://github.com/LuchnikovI/QGOpt.

[31] S. Ruder, *An overview of gradient descent optimization algorithms* (2016), arXiv:1609.04747.

[32] G. Bécigneul and O.-E. Ganea, *Riemannian adaptive optimization methods* (2018), arXiv:1810.00760.

[33] J. Li, L. Fuxin and S. Todorovic, *Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform* (2020), arXiv:2002.01113.

[34] N. Boumal, *An introduction to optimization on smooth manifolds* (2020), http://sma.epfl.ch/~nboumal/book/index.html.

[35] P.-A. Absil, R. Mahony and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, ISBN 9781400830244 (2008), doi:10.1515/9781400830244.

[36] P. Pokorny, *Geodesics revisited*, Chaotic Modeling and Simulation, Prague Institute of Chemical Technology, Prague, Czech Republic, 281 (2012).

[37] A. Jamiołkowski, *Linear transformations which preserve trace and positive semidefiniteness of operators*, Rep. Math. Phys. **3**, 275 (1972), doi:10.1016/0034-4877(72)90011-0.

[38] J. C. Bridgeman and C. T. Chubb, *Hand-waving and interpretive dance: an introductory course on tensor networks*, J. Phys. A: Math. Theor. **50**, 223001 (2017), doi:10.1088/1751-8121/aa6dc3.

[39] J. Biamonte and V. Bergholm, *Tensor networks in a nutshell* (2017), arXiv:1708.00006.

[40] A. Edelman, T. A. Arias and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. & Appl. **20**, 303 (1998), doi:10.1137/S0895479895290954.

[41] I. Luchnikov, M. Krechetov and S. Filippov, *Riemannian geometry and automatic differentiation for optimization problems of quantum physics and quantum technologies* (2020), arXiv:2007.01287.

[42] M. Hauru, M. Van Damme and J. Haegeman, *Riemannian optimization of isometric tensor networks*, SciPost Phys. **10**, 040 (2021), doi:10.21468/SciPostPhys.10.2.040.

[43] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statist. **22**, 400 (1951), doi:10.1214/aoms/1177729586.

[44] V. V. Shende, I. L. Markov and S. S. Bullock, *Minimal universal two-qubit controlled-NOT-based circuits*, Phys. Rev. A **69**, 062321 (2004), doi:10.1103/PhysRevA.69.062321.

[45] I. Luchnikov, M. Krechetov and A. Ryzhov, *QGOpt's documentation* (2020), https://qgopt.readthedocs.io/en/latest/.

[46] R. Blume-Kohout, *Optimal, reliable estimation of quantum states*, New J. Phys. **12**, 043034 (2010), doi:10.1088/1367-2630/12/4/043034.

[47] Y. Siah Teo, *Introduction to quantum-state estimation*, World Scientific, Singapore, ISBN 9789814678834 (2015), doi:10.1142/9617.

[48] G. C. Knee, E. Bolduc, J. Leach and E. M. Gauger, *Quantum process tomography via completely positive and trace-preserving projection*, Phys. Rev. A **98**, 062336 (2018), doi:10.1103/PhysRevA.98.062336.

[49] M. Mohseni, A. T. Rezakhani and D. A. Lidar, *Quantum-process tomography: Resource analysis of different strategies*, Phys. Rev. A **77**, 032322 (2008), doi:10.1103/PhysRevA.77.032322.

[50] J. M. Renes, R. Blume-Kohout, A. J. Scott and C. M. Caves, *Symmetric informationally complete quantum measurements*, J. Math. Phys. **45**, 2171 (2004), doi:10.1063/1.1737053.

[51] A. Gilchrist, N. K. Langford and M. A. Nielsen, *Distance measures to compare real and ideal quantum processes*, Phys. Rev. A **71**, 062310 (2005), doi:10.1103/PhysRevA.71.062310.

[52] S. J. Glaser et al., *Training Schrödinger's cat: quantum optimal control*, Eur. Phys. J. D **69**, 279 (2015), doi:10.1140/epjd/e2015-60464-1.

[53] M. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich and C. Koch, *Krotov: A Python implementation of Krotov's method for quantum optimal control*, SciPost Phys. **7**, 080 (2019), doi:10.21468/SciPostPhys.7.6.080.

[54] M. H. Goerz, F. Motzoi, K. Birgitta Whaley and C. P. Koch, *Charting the circuit QED design landscape using optimal control theory*, npj Quantum Inf. **3**, 37 (2017), doi:10.1038/s41534-017-0036-0.

[55] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. Blázquez García, G. Su and M. Lewenstein, *Machine learning by unitary tensor network of hierarchical tree structure*, New J. Phys. **21**, 073059 (2019), doi:10.1088/1367-2630/ab31ef.

[56] W. Huang, P.-A. Absil and K. A. Gallivan, *A Riemannian BFGS method for nonconvex optimization problems*, in Lecture Notes in Computational Science and Engineering, Springer International Publishing, Cham, ISBN 9783319399270 (2016), doi:10.1007/978-3-319-39929-4_60.

[57] H. Sato, *Riemannian conjugate gradient method for complex singular value decomposition problem*, In 53rd IEEE Conference on Decision and Control, 5849 (2014), doi:10.1109/CDC.2014.7040305.

[58] Z. Lin, *Riemanniangeometry of symmetric positive definite matrices via Cholesky decomposition*, SIAM J. Matrix Anal. Appl. **40**, 1353 (2019), doi:10.1137/18M1221084.

[59] M. Journée, F. Bach, P.-A. Absil and R. Sepulchre, *Low-rank optimization on the cone of positive semidefinite matrices*, SIAM J. Optim. **20**, 2327 (2010), doi:10.1137/080731359.

[60] N. Boumal, B. Mishra, P.-A. Absil and R. Sepulchre, *Manopt, a matlab toolbox for optimization on manifolds*, JMLR **15**, 1455 (2014).

[61] B. Mishra, H. Kasai and P. Jawanpuria, *Riemannian optimization on the simplex of positive definite matrices* (2019), arXiv:1906.10436.

[62] E. Massart and P.-A. Absil, *Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices*, SIAM J. Matrix Anal. Appl. **41**, 171 (2020), doi:10.1137/18M1231389.

[63] S. Yatawatta, *Radio interferometric calibration using a riemannian manifold*, In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 3866 (2013), doi:10.1109/ICASSP.2013.6638382.