QGOpt: Riemannian optimization for quantum technologies

I. A. Luchnikov[1,2], A. Ryzhov[2], S. N. Fillipov[1,3,4] H. 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 Steklov Mathematical Institute of Russian Academy of Sciences, Gubkina Street 8, Moscow 119991, Russia
4 Valiev Institute of Physics and Technology of Russian Academy of Sciences, Nakhimovskii Prospect 34, Moscow 117218, Russia
* h.ouerdane@skoltech.ru

November 7, 2020

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

Application of the variational method in quantum physics is akin to a constrained optimization problem. For example, the ground state of a quantum system with Hamiltonian H can be found by use of the variational principle [1]:

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

where $|\psi\rangle$ is a trial, non-normalized state, $|\Omega\rangle$ is the non-normalized ground state. This formulation of a ground state search problem was successfully used for characterization 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 optimization of variational energy one can utilize optimization algorithms such as density matrix renormalization group [12, 13], time evolving block decimation [14–16] for tensor network architectures, quantum natural gradient [17], and adaptive first order optimization methods like Adam optimizer [18] for neural networks based quantum parametrization.

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

Some problems of quantum mechanics require nonstandard optimization methods. For example, a well known entanglement renormalization technique, that is used for characterization of quantum phase transitions, requires to perform optimization over isometric matrices. To solve this problem, Vidal and Evenbly suggested an algorithm [26–28] that does not have analogs in standard optimization theory. Further, some optimization problems such as entanglement renormalization or quantum tomography, require preservation of natural "quantum" constraints, such as the completely positive and trace preserving (CPTP) property of quantum channels [29] or the orthogonality constraints of isometric or unitary matrices. Preservation of constraints can be achieved by introduction of a particular parametrization or by adding a regularization term to a loss function that enforces the satisfaction of constraints. However, a naive parametrization may lead to over-parametrization, which causes optimization slowing down.

Highly specialized algorithms such as the Vidal–Evenbly algorithm, are only suitable for a restricted set of problems. Additional regularization terms in a loss function are also not a universal solution because they provide only approximate constraints preservation. One therefore needs a universal approach to the optimization in quantum technology. As many natural "quantum" constraints can be seen as Riemannian manifolds, Riemannian optimization is a right candidate for the role of a universal framework for constrained optimization in quantum mechanics. In the present work, we introduce QGOpt (Quantum Geometric Optimization) [30], our 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 utilization in Sec. 5, with two examples: quantum gate decomposition and quantum channel tomography.

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. Optimization on curved spaces requires a generalization of these elementary operations in a certain way. As an example we consider a case of gradient descent with momentum [31] and its Riemannian generalization [32, 33]. Here, we keep our overview simple. For an in-depth introduction into the topic, we recommend the following references [34, 35].

Let us assume that we aim to minimize the value of a function $f : \mathbb{R}^n \longmapsto \mathbb{R}$, 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. Calculating 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. The sign before η indicates whether we search for a local minimum or maximum.

Let us assume now that a function f is defined on a Riemannian manifold $\mathcal{M}$ that is embedded in the Euclidean space: $f : \mathcal{M} \longmapsto \mathbb{R}$. 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 a 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 then 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 tangent vector 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 curved line step that is called retraction. The retraction $R_x(v)$ is a result of the transportation of x along a curve that completely lies in the manifold and directed along v.

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 reads:

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

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 a Choi matrices set 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: $\Phi : \mathbb{C}^{n\times n} \longmapsto \mathbb{C}^{n\times n}$. 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 $\mathrm{Tr}_p(C) = \mathbb{1}$, where Tr_p is a partial trace over the first subsystem and $\mathbb{1}$ is the identity matrix. To make the notion of the partial trace less abstract, let us consider a piece of TensorFlow code, that computes a partial trace of a Choi matrix. First of all, 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 $C_{\mathrm{resh}} \in \mathbb{C}^{n\times n\times n\times n}$ 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}. Partial trace of a Choi matrix can be calculated by using C_{resh} as follows $[\mathrm{Tr}_p(C)]_{i_1 i_2} = \sum_j [C_{\mathrm{resh}}]_{i_1 j i_2 j}$. Practically it can be done by running of the following line of code:

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

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

The Choi–Jamiołkowski isomorphism [36] 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

$$C = \mathbb{1} \otimes \Phi \left|\Psi^+\right\rangle \left\langle\Psi^+\right|, \tag{2}$$

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 essentially is a quantum channel itself, we consider the representation of Eq.(2) in terms of tensor diagrams [37, 38]. The reshaped version of a Choi matrix $[C_{\mathrm{resh}}]_{i_1 j_1 i_2 j_2}$ is shown in Fig. 1. Thus, one can conclude that the Choi matrix is a quantum channel itself.

One can see that 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 $\mathbb{C}^{n^2\times n^2}$

$$C_n = \left\{ C \in \mathbb{C}^{n^2\times n^2} \middle| C \geq 0,\ \mathrm{Tr}_p(C) = \mathbb{1} \right\}, \tag{3}$$

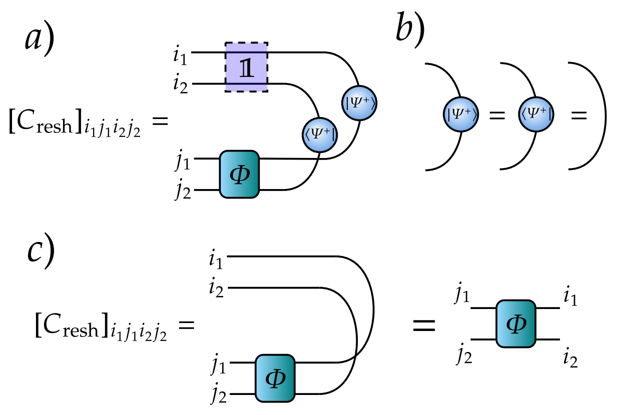


Figure 1: a) Diagrammatic representation of the Choi matrix. b) One can note that the state of a two-component quantum system $|\Psi^+\rangle$ can be seen as the identity matrix. c) Finally, we note that the Choi matrix is a quantum channel itself.

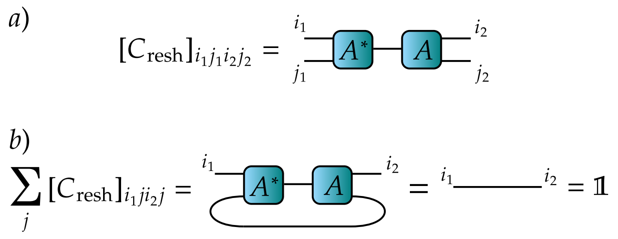

a)
$[C_{\text{resh}}]_{i_1 j_1 i_2 j_2} =$
A^*
A
b)
$\sum_j [C_{\text{resh}}]_{i_1 j i_2 j} =$
$= \mathbb{1}$

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

where $C \geq 0$, $\text{Tr}_p(C) = \mathbb{1}$ 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. Now, we may parametrize the Choi matrix with an auxiliary matrix $A \in \mathbb{C}^{n^2 \times n^2}$

$$C = A^\dagger A. \tag{4}$$

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

$$[\text{Tr}_p(C)]_{i_1 i_2} = [\text{Tr}_p(A^\dagger A)]_{i_1 i_2} = \sum_{kj} [A_{\text{resh}}]^*_{k i_1 j} [A_{\text{resh}}]_{k i_2 j} = \delta_{i_1 i_2} \tag{5}$$

which shows that a set of all A_{resh} is the set of isometric tensors. Indeed, one can notice that convolution between A_{resh} and its complex conjugated version A^*_{resh} is the identity matrix which means that any A_{resh} is an isometric tensor. A diagrammatic form of the Eq.(5) is shown in Fig. 2. The set of all complex isometric matrices of fixed size forms a Riemannian manifold called complex Stiefel manifold [39] that we denote as St.

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 QA, where Q is an arbitrary unitary matrix, correspond to the same Choi matrix; in other words we have an equivalence relation $QA \sim A$. Indeed

$$C = A^\dagger Q^\dagger Q A = A^\dagger A. \tag{6}$$

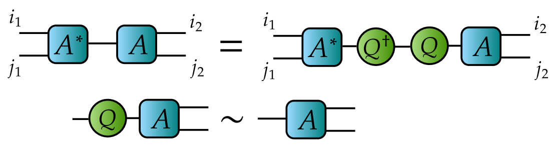
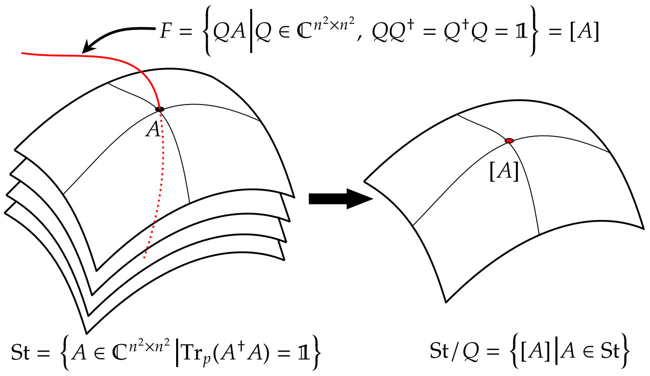


Figure 3: Diagrammatic representation of the Eq.(6.)



$F = \left\{QA \middle| Q \in \mathbb{C}^{n^2\times n^2},\ QQ^\dagger = Q^\dagger Q = \mathbb{1}\right\} = [A]$
A
$[A]$
$\mathrm{St} = \left\{A \in \mathbb{C}^{n^2\times n^2} \middle| \mathrm{Tr}_p(A^\dagger A) = \mathbb{1}\right\}$
$\mathrm{St}/Q = \left\{[A] \middle| A \in \mathrm{St}\right\}$

Figure 4: 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. Red curve represents a particular equivalence class F that is also called a fiber.

A diagrammatic version of Eq.(6) is depicted in Fig. 3. It shows that for any A there is a family of equivalent matrices $[A] = \{QA|Q \in \mathbb{C}^{n^2\times n^2},\ Q^\dagger Q = QQ^\dagger = \mathbb{1}\}$ that is called equivalence class of A, and leads to the same Choi matrix. One can eliminate this symmetry by turning to a quotient manifold $\mathrm{St}/Q = \{[A]|A \in \mathrm{St}\}$ that consists of equivalence classes. This rather abstract construction can be imagined as a projection of a manifold along surfaces representing equivalence classes (see Fig. 4). Having a map $\pi(A) = [A]$ and a map called horizontal lift [34], that connects tangent spaces of St/Q and tangent spaces of St, one can describe the abstract manifold St/Q through St. It allows one to perform a Riemannian optimization on the abstract quotient manifold St/Q.

The example Choi matrices manifold 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.

- 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 [40, 41]; thus, one can use this manifold to optimize such tensor networks.

- Quotient manifold of density matrices $\varrho_n = \left\{\varrho \in \mathbb{C}^{n\times n}\middle|\varrho = \varrho^\dagger,\ \mathrm{Tr}(\varrho) = 1,\ \varrho \succeq 0\right\}$ is a set of all fixed-size 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.

- Quotient manifold of Choi matrices $C_n = \left\{C \in \mathbb{C}^{n^2\times n^2}\middle|C = C^\dagger,\ \mathrm{Tr}_p(C) = \mathbb{1},\ C \succeq 0\right\}$ is a set of all fixed-size 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.

- Manifold of Hermitian matrices $H_n = \left\{H \in \mathbb{C}^{n\times n} \middle| H = H^\dagger\right\}$ is essentially 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.

- Manifold of hermitian positive definite matrices $\mathbb{S}^n_{++} = \left\{S \in \mathbb{C}^{n\times n} \middle| S = S^\dagger,\ S \succ 0\right\}$ 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.

- Quotient manifold of positive operator-valued measure (POVM) $\mathrm{POVM}_{m,n} = \left\{\{E_i\}_{i=1}^n \in \mathbb{C}^{n\times d\times d} \middle| E_i = E_i^\dagger,\ E_i \succeq 0,\ \sum_{i=1}^n E_i = \mathbb{1}\right\}$ can be considered as a fixed-size tensor with hermitian positive semi-definite 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.

4 QGOpt API

4.1 Manifolds API

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. Any manifold has a set of typical methods that are used in Riemannian optimization. This list of methods allows one not to pay particular attention to the underlying Riemannian geometry details.

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.

```python
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:

```python
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 that 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. 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. 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 the following line of code:

```
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. Riemannian gradient is the search direction that takes into account the metric of a manifold and the tangent space in a given point. To calculate the Riemannian gradient one can use the following piece of code:

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

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

4.2 Optimizers

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. An example of manifold 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 [42].

If m is a manifold element and lr is a learning rate, then the Adam and SGD optimizers can be initialized as follows:

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

Note, that some other attributes, like value of momentum of SGD optimizer or AMSGrad modification of Adam optimizer, can be passed exactly as for TensorFlow Adam and SGD optimizers.

4.3 Auxiliary functions

It is important to have 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 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 the real representation to the complex representation and back. Let us assume that we initialize a complex-valued tensor, which represents a point from a manifold by using 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. To make this transition simple, we introduced two auxiliary functions that allow performing conversion from the real representation to the complex and back:

```
# 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 a quantum gate decomposition. It is known, that any two qubit quantum gate U can be decomposed in the following way [43]:

$$U = [\tilde{u}_{11} \otimes \tilde{u}_{12}]U_{\rm CNOT}[\tilde{u}_{21} \otimes \tilde{u}_{22}] \times U_{\rm CNOT}[\tilde{u}_{31} \otimes \tilde{u}_{32}]U_{\rm CNOT}[\tilde{u}_{41} \otimes \tilde{u}_{42}], \tag{7}$$

where $U_{\rm CNOT}$ 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 the sake of simplicity let us denote the decomposition introduced above in the following way

$$D\left(u_{ij}\right) = [u_{11} \otimes u_{12}]U_{\rm CNOT}[u_{21} \otimes u_{22}] \times U_{\rm CNOT}[u_{31} \otimes u_{32}]U_{\rm CNOT}[u_{41} \otimes u_{42}] \tag{8}$$

The problem of gate decomposition can be formulated as the following optimization problem:

$$\|U - D(u_{ij})\|_F \to \min_{\{u_{ij}\}_{i,j=1}^{4,2}} \tag{9}$$

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

Before considering the main part of the code that solves the problem above, 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.
```

Then we define an example of the complex Stiefel manifold:

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

As a target gate that we want to decompose, we use a randomly generated one:

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

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

```
cnot = tf.constant([[1, 0, 0, 0],
                    [0, 1, 0, 0],
                    [0, 0, 0, 1],
                    [0, 0, 1, 0]],
                    dtype=tf.complex128).
```

As the next step, we initialize Riemannian Adam optimizer:

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

and run the forward pass of computations:

```
with tf.GradientTape() as tape:
    # turning u back into its
    # complex representation
    uc = qgo.manifolds.real_to_complex(u)
    # decomposition
    D = kron(uc[0, 0], uc[0, 1])
    D = cnot @ D
    D = kron(uc[1, 0], uc[1, 1]) @ D
    D = cnot @ D
    D = kron(uc[2, 0], uc[2, 1]) @ D
    D = cnot @ D
    D = kron(uc[3, 0], uc[3, 1]) @ D
    # loss function
    L = tf.linalg.norm(D - U) ** 2
    # is equivalent to casting to a real dtype
    L = tf.math.real(L).
```

The final step is to minimize the loss function $L = \|D(u_{ij}) - U\|_F^2$ that is calculated on the previous step. We calculate gradient of L 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])).
```

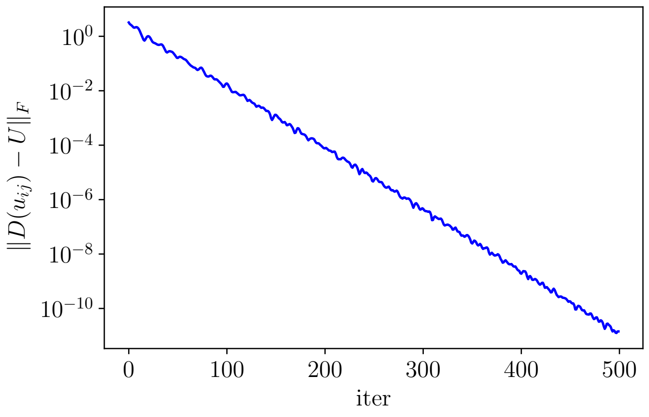


Figure 5: Frobenius distance between a gate and its decomposition. One can see that the distance rapidly decreases with the number of iteration towards almost a machine zero.

Adam optimizer performs one optimization step keeping 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. 5. One can see that at the end of the optimization process, the error is completely negligible. This section in the form of tutorial is available at the QGOpt documentation web-page [44].

5.2 Quantum tomography

Another typical problem that can be addressed by Riemannian optimization is the quantum tomography of states [45, 46] and channels [47, 48]. 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 $\Phi_{\rm true}$ and observes a set of measurement outcomes $\left\{M^{\rm tetra}_{k_i^1} \otimes \cdots \otimes M^{\rm tetra}_{k_i^n}\right\}_{i=1}^{N}$, where $M^{\rm tetra}_k$ is a tetrahedral POVM [49]:

$$M^{\rm tetra}_k = \frac{1}{4}\left(\mathbb{1} + \boldsymbol{s}_k^T \boldsymbol{\sigma}\right), \; k \in (0,1,2,3), \tag{10}$$

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

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

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

$$\sum_{i=1}^{N} \log\left(M^{\rm tetra}_{k_i^1} \otimes \cdots \otimes M^{\rm tetra}_{k_i^n} \Phi(\rho_i)\right) \to \max_{\Phi \text{ is CPTP}}. \tag{11}$$

For the sake of 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, an unknown channel has Kraus rank 2 and is generated randomly, initial density matrices are pure and also generated randomly.

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

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

The manifold of Choi matrices is represented through the quadratic parametrization with equivalence relation discussed in Section 3. Thus we initialize a variable, that represents the parametrization of a Choi matrix:

```
# random initial parametrization
A = m.random((2**(2*n), 2**(2*n)),
dtype=tf.complex128)
# variable should be real
# to make an optimizer work correctly
A = qgo.manifolds.complex_to_real(A)
# variable
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:

```
with tf.GradientTape() as tape:
    # complex representation of parametrization
    # shape=(2**2n, 2**2n)
    Ac = qgo.manifolds.real_to_complex(A)

    # reshape parametrization
    # (2**2n, 2**2n) --> (2**n, 2**n, 2**2n)
    Ac = tf.reshape(Ac, (2**n, 2**n, 2**(2*n)))

    # Choi tensor (reshaped Choi matrix)
    choi = tf.tensordot(Ac,
                        tf.math.conj(Ac),
                        [[2], [2]])

    # turning Choi tensor to the
    # corresponding quantum channel
    phi = tf.transpose(choi, (1, 3, 0, 2))
    phi = tf.reshape(phi, (2**(2*n), 2**(2*n)))

    # reshape initial density
    # matrices to vectors
    rho_resh = tf.reshape(rho_in, (N, 2**(2*n)))

    # passing density matrices
    # through a quantum channel
    rho_out = tf.tensordot(phi,
                           rho_resh,
                           [[1], [1]])
    rho_out = tf.transpose(rho_out)
    rho_out = tf.reshape(rho_out,
                         (N, 2**n, 2**n))

    # probabilities of measurement outcomes
    # (povms is a set of POVM elements
    # came true of shape (N, 2**n, 2**n))
    p = tf.linalg.trace(povms @ rho_out)
```

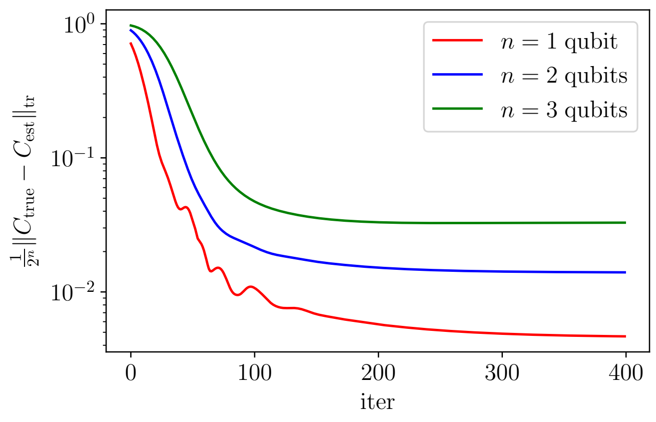


Figure 6: Dependence between Jamiołkowski process distance and number of iteration. Number of measurement outcomes $N = 600000$ for all experiments.

```

    # negative log likelihood (to be minimized)
    L = -tf.reduce_mean(tf.math.log(p)).
```

Finally, we calculate the logarithmic likelihood gradient with respect to the parametrization of the Choi matrix:

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

and apply the optimizer that makes 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 Jamiołkowski process distance [50]:

$$J(\Phi_{\rm true}, \Phi_{\rm est}) = \frac{1}{2^n}\|C_{\rm true} - C_{\rm est}\|_{\rm tr}, \tag{12}$$

where $\Phi_{\rm true}(\Phi_{\rm est})$ is the true (estimated) quantum channel, $C_{\rm true}(C_{\rm est})$ is the corresponding Choi matrix, $\|\cdot\|_{\rm tr}$ is the trace norm and $0 \leq J(\Phi_{\rm true}, \Phi_{\rm est}) \leq 1$. One can see in Fig. 6 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 in the form of tutorial is available at QGOpt documentation web-page [44].

6 Discussion and concluding remarks

The range of applications of the QGOpt library to different problems of quantum technology is not limited only by quantum gate decomposition and quantum tomography. The six manifolds implemented in QGOpt give rise to different interesting scenarios of constrained optimization usage in quantum technology. For example, the complex Stiefel manifold can be used to address different control problems, where one needs to find an optimal set of unitary gates driving a quantum system to a desirable quantum state. Is is also possible to use a complex Stiefel manifold to perform entanglement renormalization [40, 41], machine learning by unitary tensor networks [51] or non-Markovian quantum dynamics identification [24]. Besides quantum tomography, quotient manifolds of 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 an optimal generalized measurement scheme, that gives maximal information gain. Finally, all these manifolds can be combined in one optimization task, which allows to address multi-component problems.

To conclude, we introduce QGOpt library that is aimed at solving constrained optimization problems with natural quantum constraints. We introduce and discuss quite an abstract concept, such as quotient manifolds, which lie under the hood of QGOpt. We go through QGOpt API and cover the most important features of it. We also sort out two examples of code solving two illustrative quantum technology problems.

Acknowledgements

The authors thank Stephen Vintskevich and Mikhail Krechetov for fruitful discussions.

Funding information Authors are required to provide funding information, including relevant agencies and grant numbers with linked author's initials. Correctly-provided data will be linked to funders listed in the Fundref registry.

References

[1] J. Toulouse, R. Assaraf and C. J. Umrigar, *Introduction to the variational and diffusion monte carlo methods*, In *Advances in Quantum Chemistry*, vol. 73, pp. 285–314. Elsevier (2016).

[2] D. Ceperley, G. V. Chester and M. H. Kalos, *Monte carlo simulation of a many-fermion study*, Physical Review B **16**(7), 3081 (1977).

[3] D. Bressanini, G. Morosi and M. Mella, *Robust wave function optimization procedures in quantum monte carlo methods*, The Journal of chemical physics **116**(13), 5345 (2002).

[4] U. Schollwöck, *The density-matrix renormalization group in the age of matrix product states*, Annals of physics **326**(1), 96 (2011).

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

[6] R. Orús, *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*, Annals of Physics **349**, 117 (2014).

[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*, Advances in Physics **57**(2), 143 (2008).

[8] F. Verstraete and J. I. Cirac, *Renormalization algorithms for quantum-many body systems in two and higher dimensions*, arXiv preprint cond-mat/0407066 (2004).

[9] G. Carleo and M. Troyer, *Solving the quantum many-body problem with artificial neural networks*, Science **355**(6325), 602 (2017).

[10] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko and J. Carrasquilla, *Recurrent neural network wave functions*, Physical Review Research **2**(2), 023358 (2020).

[11] K. Choo, T. Neupert and G. Carleo, *Two-dimensional frustrated j 1- j 2 model studied with neural network quantum states*, Physical Review B **100**(12), 125124 (2019).

[12] S. R. White, *Density matrix formulation for quantum renormalization groups*, Physical Review Letters **69**(19), 2863 (1992).

[13] U. Schollwöck, *The density-matrix renormalization group*, Reviews of modern physics **77**(1), 259 (2005).

[14] G. Vidal, *Efficient classical simulation of slightly entangled quantum computations*, Physical Review Letters **91**(14), 147902 (2003).

[15] G. Vidal, *Efficient simulation of one-dimensional quantum many-body systems*, Physical Review Letters **93**(4), 040502 (2004).

[16] R. Orus and G. Vidal, *Infinite time-evolving block decimation algorithm beyond unitary evolution*, Physical Review B **78**(15), 155117 (2008).

[17] J. Stokes, J. Izaac, N. Killoran and G. Carleo, *Quantum natural gradient*, Quantum **4**, 269 (2020).

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

[19] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko and G. Carleo, *Neural-network quantum state tomography*, Nature Physics **14**(5), 447 (2018).

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

[21] J. Carrasquilla, G. Torlai, R. G. Melko and L. Aolita, *Reconstructing quantum states with generative models*, Nature Machine Intelligence **1**(3), 155 (2019).

[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**(11), 1091 (2019).

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

[24] I. A. Luchnikov, S. V. Vintskevich, D. A. Grigoriev and S. N. Filippov, *Machine learning non-markovian quantum dynamics*, Physical Review Letters **124**(14), 140502 (2020).

[25] L. Banchi, E. Grant, A. Rocchetto and S. Severini, *Modelling non-markovian quantum processes with recurrent neural networks*, New Journal of Physics **20**(12), 123030 (2018).

[26] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization*, Physical Review B **79**(14), 144108 (2009).

[27] G. Evenbly and G. Vidal, *Algorithms for entanglement renormalization: boundaries, impurities and interfaces*, Journal of Statistical Physics **157**(4-5), 931 (2014).

[28] G. Vidal, *Entanglement renormalization*, Physical Review Letters **99**(22), 220405 (2007).

[29] A. S. Holevo, *Quantum systems, channels, information: a mathematical introduction*, vol. 16, Walter de Gruyter (2012).

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

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

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

[33] J. Li, L. Fuxin and S. Todorovic, *Efficient riemannian optimization on the stiefel manifold via the cayley transform*, arXiv preprint arXiv:2002.01113 (2020).

[34] N. Boumal, *An introduction to optimization on smooth manifolds*, Available online (2020).

[35] P.-A. Absil, R. Mahony and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press (2009).

[36] A. Jamiołkowski, *Linear transformations which preserve trace and positive semidefiniteness of operators*, Reports on Mathematical Physics **3**(4), 275 (1972).

[37] J. C. Bridgeman and C. T. Chubb, *Hand-waving and interpretive dance: an introductory course on tensor networks*, Journal of Physics A: Mathematical and Theoretical **50**(22), 223001 (2017).

[38] J. Biamonte and V. Bergholm, *Tensor networks in a nutshell*, arXiv preprint quant-ph/arXiv:1708.00006 (2017).

[39] A. Edelman, T. A. Arias and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM journal on Matrix Analysis and Applications **20**(2), 303 (1998).

[40] I. Luchnikov, M. Krechetov and S. Filippov, *Riemannian optimization and automatic differentiation for complex quantum architectures*, arXiv preprint arXiv:2007.01287 (2020).

[41] M. Hauru, M. Van Damme and J. Haegeman, *Riemannian optimization of isometric tensor networks*, arXiv preprint arXiv:2007.03638 (2020).

[42] H. Robbins and S. Monro, *A stochastic approximation method*, The Annals of Mathematical Statistics **22**(3), 400 (1951).

[43] V. V. Shende, I. L. Markov and S. S. Bullock, *Minimal universal two-qubit controlled-not-based circuits*, Physical Review A **69**(6), 062321 (2004).

[44] I. Luchnikov, M. Krechetov and A. Ryzhov, `https://qgopt.readthedocs.io/en/latest/` (2020).

[45] R. Blume-Kohout, *Optimal, reliable estimation of quantum states*, New Journal of Physics **12**(4), 043034 (2010).

[46] Y. S. Teo, *Introduction to quantum-state estimation*, World Scientific (2016).

[47] G. C. Knee, E. Bolduc, J. Leach and E. M. Gauger, *Quantum process tomography via completely positive and trace-preserving projection*, Physical Review A **98**(6), 062336 (2018).

[48] M. Mohseni, A. T. Rezakhani and D. A. Lidar, *Quantum-process tomography: Resource analysis of different strategies*, Physical Review A **77**(3), 032322 (2008).

[49] J. M. Renes, R. Blume-Kohout, A. J. Scott and C. M. Caves, *Symmetric informationally complete quantum measurements*, Journal of Mathematical Physics **45**(6), 2171 (2004).

[50] A. Gilchrist, N. K. Langford and M. A. Nielsen, *Distance measures to compare real and ideal quantum processes*, Physical Review A **71**(6), 062310 (2005).

[51] D. Liu, S.-J. Ran, P. Wittek, C. Peng, R. B. García, G. Su and M. Lewenstein, *Machine learning by unitary tensor network of hierarchical tree structure*, New Journal of Physics **21**(7), 073059 (2019).