SCHOOL OF NATURAL SCIENCES

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Quantum Science and Technology

A Riemannian Approach to the Lindbladian Dynamics of a Locally Purified Tensor Network

Cristian Emiliano Godínez Ramírez

SCHOOL OF NATURAL SCIENCES

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Quantum Science and Technology

A Riemannian Approach to the Lindbladian Dynamics of a Locally Purified Tensor Network

Ein Riemannscher Ansatz zur Lindbladschen Dynamik eines lokal purifizierten Tensornetzwerks

Author:	Cristian Emiliano Godínez Ramírez
Supervisor:	Christian Mendl
Advisor:	Richard Milbradt
Submission Date:	02.04.2024

Acknowledgments

I want to thank Prof. Dr. Christian Mendl for all the guidance, support, and teachings as my thesis supervisor, professor, and academic advisor. Likewise, I want to thank Richard Milbradt for the time dedicated to supporting my project and for all the fruitful conversations. Thank you both for your amiability throughout the whole process.

I also want to thank the friends I made over the past two years, with a special mention to Fiona Fröhler for all the intellectually stimulating conversations and the ideas that sparked from them. Whether by proofreading my thesis, sharing a laugh with me, or being there when I needed you the most, you have all been a crucial part of my time as a master's student.

Finally, I want to thank my father for teaching me the importance of hard work and the beauty of math, my mother for teaching me how to smile through the toughest of times, and my grandparents for showing me unconditional support, in body and soul. I love you all immensely. This achievement is also yours.

Thank you,

Cristian Emiliano Godínez Ramírez

Abstract

Tensor networks offer a valuable framework for implementing Lindbladian dynamics in many-body open quantum systems with nearest-neighbor couplings. In particular, a tensor network ansatz known as the Locally Purified Density Operator employs the local purification of the density matrix to guarantee the positivity of the state at all times. Within this framework, the dissipative evolution utilizes the Trotter-Suzuki splitting, yielding a second-order approximation error. However, due to the Lindbladian dynamics' nature, employing higher-order schemes results in non-physical quantum channels. In this work, we leverage the gauge freedom inherent in the Kraus representation of quantum channels to improve the splitting error. To this end, we formulate an optimization problem on the Riemannian manifold of isometries and find a solution via the second-order trust-region algorithm. We validate our approach using two nearest-neighbor noise models and achieve an improvement of orders of magnitude compared to other positivity-preserving schemes. In addition, we demonstrate the usefulness of our method as a compression scheme, helping to control the exponential growth of computational resources, which thus far has limited the use of the locally purified ansatz.

Kurzfassung

Tensor-Netzwerke bieten einen wertvollen Rahmen für die Implementierung von Lindblad-Dynamiken in offenen Vielteilchen-Quantensystemen mit Nachbarwechselwirkungen. Insbesondere verwendet ein Tensor-Netzwerk-Ansatz, bekannt als der lokal purifizierte Dichtematrix-Operator, die lokale Purifizierung der Dichtematrix, um die Positivität des Zustands zu jeder Zeit zu garantieren. Innerhalb dieses Rahmens nutzt die dissipative Evolution die Trotter-Suzuki-Aufspaltung, was einen Fehler zweiter Ordnung ergibt. Aufgrund der Natur der Lindblad-Dynamik führt jedoch die Verwendung von Schemata höherer Ordnung zu nicht-physikalischen Quantenkanälen. In dieser Arbeit nutzen wir die Eichfreiheit, die in der Kraus-Darstellung von Quantenkanälen inhärent ist, um den Aufspaltungsfehler zu verbessern. Zu diesem Zweck formulieren wir ein Optimierungsproblem auf der Riemannschen Mannigfaltigkeit der Isometrien und finden eine Lösung über den Trust-Region-Algorithmus zweiter Ordnung. Wir validieren unseren Ansatz anhand von zwei Nachbar-Rauschmodellen und zeigen, dass wir in einigen Fällen Verbesserungen um Größenordnungen im Vergleich zu anderen Positivität-erhaltenden Schemata erreichen können. Darüber hinaus demonstrieren wir die Nützlichkeit unserer Methode als Kompressionsschema, um das exponentielle Wachstum der Rechenressourcen zu kontrollieren, was bisher die Verwendung des lokal purifizierten Ansatzes eingeschränkt hat.

Contents

1. Introduction

The study of many-body quantum systems is of utmost importance in various branches of physics, including quantum information and quantum computing. Moreover, to understand realistic physical systems, we ought to consider how these interact with their environment, i.e. we need to understand how they evolve under the influence of noise. This shifts our focus to what is known as *open quantum systems* [1]. In particular, we are interested in the evolution under Markovian noise, giving rise to the *Lindblad master equation* [1, 2]. Due to the limited analytical methods available to describe such systems, numerical methods play a pivotal role in our understanding of the so-called *dissipative dynamics*. In this context, tensor networks variational ansätze [3] emerge as an efficient approach capable of describing several systems of interest. One such structure, known as the *Locally Purified Density Operator* (LPDO), exploits the main features of tensor networks ansätze while preserving the physical properties of open quantum systems [4, 5]. Werner et al. [6] introduced an algorithm based on the ubiquitous Trotter-Suzuki splitting [7] to simulate the Lindblad evolution with nearest-neighbor interactions using the LPDO ansatz. However, due to the semigroup property of Markovian dynamics, it remains an open problem to simulate the dissipative evolution yielding higher-order approximation errors.

In this research, we aim to enhance the accuracy of the second-order Trotter splitting method, which is used in the nearest-neighbor Lindbladian dynamics. To this end, we leverage the gauge freedom present in the Kraus representation of quantum channels to formulate an optimization problem on the *Riemannian manifold* of isometry matrices, i.e. the *Stiefel manifold*. We then proceed to solve it via the second-order *trust-region* algorithm, taking advantage of its global convergence properties [8].

The rest of this work is structured as follows. In Chapter 2, we cover the theory needed to understand and formulate the main optimization problem addressed in this work. We begin by discussing the Markovian dynamics of open quantum systems, followed by a description of tensor networks, both as a diagrammatic language and as a numerical backend. Then, we explore the basics of optimization algorithms on Riemannian manifolds, focusing on the Stiefel manifold and the components required for implementing the trust-region algorithm. Next, in Chapter 3, we leverage this theory to formulate an optimization problem acting on a real-valued function defined on the Stiefel manifold. The caveats of the implementation along with a summary of the main algorithm are discussed in Chapter 4. We conclude this work by presenting the numerical simulations in Chapter 5, where we study the effects of the input parameters on the algorithm performance, benchmark our work against other known schemes, and analyze the scope of our approach. In all simulations, we have employed two nearest-neighbor models, namely the *Kitaev wire* [9] and a modification of the sparse Pauli-Lindblad model [10] we call *Pseudo Sparse Pauli-Lindblad* (PSPL).

2. Theory

Across this work we will follow a convention similar to the one in [11] with the following definitions:

Definition 2.0.1 (Hilbert spaces)**.** $\{\mathscr{H}_i\}_{i\in\mathbb{N}}$ are finite dimensional complex Hilbert spaces of dimension d_i. We will omit the index i whenever we can infer the Hilbert space from the context without ambiguity.

Definition 2.0.2 (Linear operators)**.** $\mathscr{L}(\mathscr{H}_1, \mathscr{H}_2)$ is the space of bounded linear operators from $\mathscr{H}_1$ to $\mathscr{H}_2$

$$\sigma : \mathscr{H}_1 \to \mathscr{H}_2 \tag{2.1}$$

with $\mathscr{L}(\mathscr{H}_1) := \mathscr{L}(\mathscr{H}_1, \mathscr{H}_1)$. Writing $\sigma \in \mathscr{L}(\mathscr{H}_1, \mathscr{H}_2)$ means that σ is a $d_1 \times d_2$ matrix.

Definition 2.0.3 (Superoperators)**.** Also known as *operator maps*, $\mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ is the space of linear maps from operators to operators

$$\Phi : \mathscr{L}(\mathscr{H}_1) \to \mathscr{L}(\mathscr{H}_2) \tag{2.2}$$

with $\mathscr{T}(\mathscr{H}_1) := \mathscr{T}(\mathscr{H}_1, \mathscr{H}_1)$.

2.1. Open quantum systems

The study of closed quantum systems is fully governed by Schrödinger's equation, giving rise to a deterministic process. However, closed systems undergoing a unitary evolution are rather a mathematical artifact. In reality, quantum systems interact with their environment, which means that to accurately depict their time dynamics, we must look at the interaction between the quantum system of interest and its surroundings. We label these systems as *open quantum systems*. The state of these systems is described by a **stochastic** process, since the environment's effect on the system is to produce stochastic transitions between energy levels and to randomize the phase between states in the system of interest. A comprehensive resource describing the physical concepts and mathematical techniques surrounding open quantum system dynamics can be found in [1].

Based on this fundamental difference, while state vectors are used to describe closed systems, the density matrix formalism is the standard method to describe the state and dynamics of an open system, as the density matrix ρ^{sys} represents a probability distribution of the system's state

$$\rho^{\text{sys}} = \sum_i p_i |\psi_i\rangle\langle\psi_i| \,, \quad \sum_i p_i = 1 \,. \tag{2.3}$$

In this equation, $|\psi_i\rangle$ are the possible pure states the system can be in, each with a classical probability of occurring p_i. In complete analogy to Schrödinger's equation being used to describe the time evolution of a closed system, one can use the so-called *master equations* to describe the evolution of an open system. The particular form of this equation depends on the assumptions done on the nature of the system, the environment, and their interactions. However, the general recipe focuses on a reduced description of the full system, by tracing out the degrees of freedom of the environment. This leaves us with the system of interest being affected implicitly by its environment

$$\rho^{\text{sys}} = \text{Tr}_{\text{env}}\left[\rho^{\text{total}}\right] . \tag{2.4}$$

A particularly common instance of these equations, partly due to its wide applicability, is the *Lindbladian master equation*. The particular assumptions giving rise to this equation are [1, 12]:

1. Separability: At the initial time t_0, which we define to be zero without loss of generality, there are no correlations between the system and its environment. Hence, we start with a product state $\rho_{t_0}^{\text{total}} = \rho_{t_0}^{\text{sys}} \otimes \rho_{t_0}^{\text{env}}$.
2. Born approximation: The back reaction of the system onto the environment is negligible (weak interaction), and the separability condition is preserved (environment is much larger than the system), i.e. $\rho_t^{\text{total}} \approx \rho_t^{\text{sys}} \otimes \rho_{t_0}^{\text{env}}$.
3. Markov approximation: The correlation functions of the environment decay way faster than the smallest correlations of the system. This is known as a *short memory environment*.
4. Secular approximation: We neglect the fast terms in the interaction picture.

Remark. These conditions are concisely described in the "Lindbladian master equation solver" tutorial by *QuTip* [13, 14].

While a thorough review on the Lindblad equation and its different derivations can be found in [2], here we focus on a more mathematical description as outlined in [15] and [16], as this will allow us to introduce some useful concepts that we will employ later on. Under the aforementioned assumptions, the stochastic process describing our open quantum system becomes a Markov process, i.e. a process whose next state depends only on the current state. Analogous to the equation describing the evolution of a classical markov process, the differential equation for our system's density matrix ρ is

$$\frac{d\rho}{dt} = \mathcal{L}(\rho), \tag{2.5}$$

where we dropped the index and the time dependence for convenience. The operator $\mathcal{L}$ in Eq. (2.5) is an example of a superoperator as defined in Def. (2.0.3). Similarly to the classical linear equation of the form

$$\frac{dx}{dt} = Ax,$$

where the solution is $x = e^{At}$, the formal solution of Eq. (2.5) is given by,

$$|\rho_t\rangle\rangle = \Lambda_t\,|\rho_0\rangle\rangle, \qquad \Lambda_t = e^{t\hat{\mathcal{L}}}, \qquad t \geq 0, \tag{2.6}$$

with Λ_t a superoperator whose structure we will explore soon, and $|\rho_i\rangle\rangle$ and $\hat{\mathcal{L}}$ the *vectorizations* of ρ_i and $\mathcal{L}$, respectively.

Definition 2.1.1 (Vectorization)**.** A vectorization is a linear mapping that transforms a matrix into a vector. In literature [17], we can find two conventions: row-wise and column-wise vectorization, depending on whether we stack the matrix row by row, or column by column, respectively. In the rest of this work, we will follow the **row-wise** convention. For a matrix $A = \sum_{i,j} a_{i,j}|v_i\rangle\langle w_j|$ belonging to $\mathscr{L}(\mathscr{H}_1, \mathscr{H}_2) \cong \mathbb{C}^{d_1\times d_2}$, its row-wise vectorization is defined as

$$\mathrm{vec}(A) = |A\rangle\rangle := \sum_{i,j} a_{i,j}|w_j\rangle \otimes |v_i\rangle \in \mathbb{C}^{d_2} \otimes \mathbb{C}^{d_1}. \tag{2.7}$$

For completeness, the column-wise vectorization is

$$\mathrm{vec}(A)_C = |A\rangle\rangle_C := \sum_{i,j} a_{i,j}|v_i\rangle \otimes |w_j\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}. \tag{2.8}$$

This transformation expresses the isomorphism between matrices and vector spaces:

$$\mathbb{C}^{m\times n} := \mathbb{C}^m \otimes \mathbb{C}^n \cong \mathbb{C}^{mn} \tag{2.9}$$

The next step involves finding out the structure that $\mathcal{L}$ should follow, in order for Eq. (2.6) to have physical meaningfulness. To begin with, for ρ to represent a physical system, we will require that it satisfies the following conditions:

1. Hermitian: $\rho = \rho^\dagger \iff \rho = UDU^\dagger$, with U a unitary matrix, and D a diagonal matrix of the eigenvalues λ_i, such that all of them satisfy $\lambda_i = \lambda_i^*$ (complex conjugate) $\implies$ all the eigenvalues are real: $\lambda_i \in \mathbb{R}$.
2. Unit trace: $\mathrm{Tr}[\rho] = 1$, so that the probabilities p_i in Eq. (2.3) are normalized.
3. Positive semi-definite: $\rho \geq 0$, which further restricts the eigenvalues λ_i to be positive or zero, hence, allowing us to associate them with physical probabilities.

Definition 2.1.2 (Positive semi-definite)**.** An operator $\rho \in \mathscr{L}(\mathscr{H})$ is called positive semidefinite, and we write $\rho \geq 0$, if it satisfies any of the equivalent definitions [18]:

1. $\langle x|\rho|x\rangle \geq 0, \forall x \in \mathscr{H}$.
2. $\rho = XX^\dagger$, for some $X \in \mathscr{L}(\mathscr{H})$.
3. $\rho = \rho^\dagger \wedge \mathrm{spec}(\rho) \subseteq [0, \infty)$.

If ρ_0 is a valid density matrix, we want Λ_t to output a valid density matrix ρ_t at all times. In other words, we want Λ_t to generate a *Completely Positive Trace Preserving* (CPTP) linear map, i.e. a *quantum channel* Φ. Therefore, the map Φ sending density matrices from $\mathscr{H}_1$ to $\mathscr{H}_2$, see Def. (2.0.3), must satisfy the conditions stated in Section 9.8 of [19] and in [20]:

Definition 2.1.3 (Linearity)**.** Consider an ensemble of systems that can be found in state ρ_1 and ρ_2 with probabilities p_1 and p_2, respectively. With the full ensemble described as: $\rho = p_1\rho_1 + p_2\rho_2$. The map Φ defined above is a linear map if

$$\Phi(\rho) = p_1\Phi(\rho_1) + p_2\Phi(\rho_2). \tag{2.10}$$

This is equivalent to describing the map as "respecting the mixing of states".

Definition 2.1.4 (Trace preserving (TP))**.** A linear map Φ is said to be trace preserving if

$$\mathrm{Tr}[\Phi(\rho)] = \mathrm{Tr}[\rho], \qquad \forall \rho \in \mathscr{L}(\mathscr{H}) \tag{2.11}$$

Looking at Eq. (2.3), this implies a preservation of the normalization of the probabilities p_i.

Definition 2.1.5 (Positive)**.** A linear map Φ is positive (or rather, positive preserving) if it satisfies

$$\Phi(\rho) \geq 0, \qquad \forall \rho \geq 0. \tag{2.12}$$

So that it maps positive semi-definite operators to positive semi-definite operators.

Besides requiring it to be positive, the quantum channel should also satisfy a stronger condition, namely, being *completely positive*.

Definition 2.1.6 (Completely positive (CP))**.** A linear map Φ is completely positive if

$$\Phi \otimes \mathrm{id}_n \text{ is positive}, \qquad \forall n \in \mathbb{N}, \tag{2.13}$$

where id_n is the identity map acting on the auxiliary Hilbert space $\mathscr{H}^n$ with dimension n. From this definition we see that CP implies positivity, but the converse is not always true.

We can get an intuition on why complete positivity is required, and not only positivity, by thinking about an entangled state ρ_{AB} on a bipartite system AB. If we decide to act with the quantum channel Φ **only** on the subsystem A, and let B untouched, we would like to get as a result another physical state ρ'_{AB}. However, due to the entanglement in the system, even this innocent-looking operation may result in nonphysical results, i.e. negative eigenvalues on the output density matrix. For a common example of a map satisfying positivity but not CP, see the *transpose* map in example 1.18 of [18].

Definition 2.1.7 (Completely positive trace preserving (CPTP))**.** A linear map Φ is CPTP, if it satisfies Defs. (2.1.4) and (2.1.6).

A CPTP linear map ensures that if the spectrum of the original density matrix encodes a valid probability distribution, so does the final state [20]. The final piece of our puzzle to find out the structure of $\mathcal{L}$ is to realize that the one-parameter family of maps $\{\Lambda_t, t \geq 0\}$ forms a *semigroup*, with $\mathcal{L}$ as its *generator*.

Definition 2.1.8 (Semigroup)**.** The family of maps $\{\Lambda_t, t \geq 0\}$ together with *matrix multiplication* as the composition operation form a semigroup [21]. Therefore, Λ_t satisfies the semigroup (markov) property or composition rule

$$\Lambda_{t_2+t_1} = \Lambda_{t_2}\Lambda_{t_1}, \qquad \forall t_1, t_2 \geq 0. \tag{2.14}$$

Contrary to a full group, the positivity constraint on t implies that the inverse term $\Lambda_{-t} = \mathrm{e}^{-t\mathcal{L}}$ is not necessarily a member of the group.

This important distinction is the manifestation of *irreversible dynamics*, allowing us to distinguish the future from the past [15], and will play a pivotal role in the development of our main algorithm. An alternative way of arriving to the semigroup property of Λ_t is to observe the linearity in Eq. (2.5), implying the divisibility in its solution.

Finally, (and remarkably) just by requiring Λ_t to be a CPTP map satisfying the semigroup property in Eq. (2.14), Lindblad, Gorini, Kossakowski, and Sudarshan (L-GKS) introduced the most general form of the generator $\mathcal{L}$ in [12].

Theorem 2.1.1 (Lindblad's theorem)**.** The generator $\mathcal{L}$ of any quantum operation satisfying the semigroup property must have the form

$$\begin{aligned}\mathcal{L}(\rho) =& - i(H\rho - \rho H) + \sum_k \left(L_k \rho L_k^\dagger - \frac{1}{2} L_k^\dagger L_k \rho - \frac{1}{2} \rho L_k^\dagger L_k \right) \\ =& \mathcal{H}\left(\rho\right) + \mathcal{D}\left(\rho\right) ,\end{aligned} \tag{2.15}$$

where the first term $\mathcal{H}\left(\rho\right)$ corresponds to the von Neumann equation (unitary evolution) through the Hamiltonian $H = \sum_j H_j$ of the system, and second term is known as the *dissipative* term $\mathcal{D}(\rho)$, with the *jump operators* L_k modelling the interaction of the system with its environment. Any evolution generated from this equation is guaranteed to be CPTP (physical). Conversely, any CPTP and divisible is guaranteed to have a generator of this form.

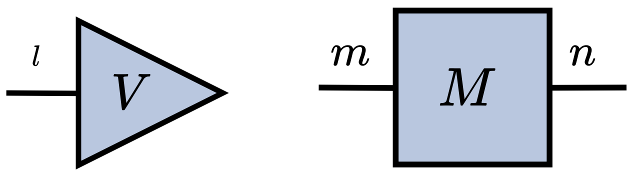
In this work, we will focus on the typical scenario where both the unitary and the dissipative terms are made up of two-local operators, i.e. the individual Hamiltonian and Lindblad terms only couple neighboring sites, $H_j^{[l,l+1]}$ and $L_k^{[l,l+1]}$, respectively.

2.2. A diagrammatic language

Tensor networks have become a ubiquitous technique in many areas of physics, ranging from condensed matter to quantum information, and other areas of application such as machine learning. In particular, using tensor networks as backends for numerical simulations has become a well established approach when trying to assess the state-of-the-art classical methods [22, 23, 24].

However, it is also important to recall that tensor networks go far beyond a numerical tool. They provide us with an intuitive graphical language that can be used in formal reasoning and proofs. A great resource on this topic are the lecture notes of J.Biamonte [25]. In this work, we intend to use this diagrammatic reasoning language whenever possible. Therefore, this section will begin with a brief overview of the tensor techniques we will use across the work. Previous familiarity with the tensor network notation and basic operations, as well as the basic tensor network structures, is assumed. Great introductory materials to these topics include [26] and [27], as well as a personal favourite in the work of V. Bergholm and J.Biamonte [28].

Let us begin by recalling two well known tensor diagrams, a rank-one and a rank-two tensor, i.e. a vector and a matrix, respectively.

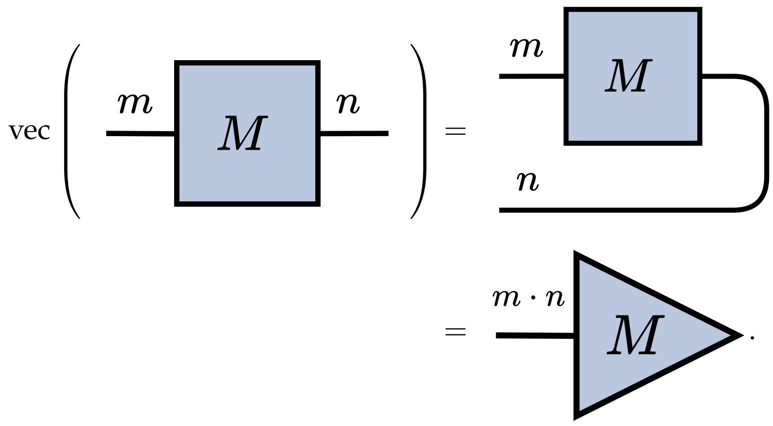


(a) A rank-one tensor (b) A rank-two tensor

Figure 2.1.: Two basic tensor diagrams: (a) shows a rank-one tensor, which is a vector, and (b) shows a rank-two tensor, a matrix.

Remark. The indices m and n of the matrix in Fig. (2.1b) correspond to the row and column dimensions, respectively.

Back in Def. (2.1.1), we described the concept of vectorization of a matrix in mathematical terms. Conversely, now we look at this transformation through the tensor diagrams lens. The diagrammatic representation of the isomorphism in Eq. (2.9) is



(2.16)

This process can be understood as the *bending of wires* of the original tensor. For the row-convention we introduced earlier, this means *grabbing* the right leg of the matrix, bending it **clockwise** to the left side, and *merging* the two left-oriented legs to obtain a vector. The resulting tensor from this procedure is the vector $|v\rangle = |M\rangle\rangle$. A natural question to ask at this point is "*What would happen if instead we bend the right wire anti-clockwise?*", as shown in

Fig. (2.2). Interestingly, this transformation results again in a vector of dimension $m \cdot n$, but instead corresponds to the column convention for vectorization!

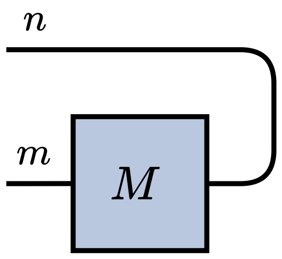


Figure 2.2.: Vectorization of matrix M using the column-wise convention.

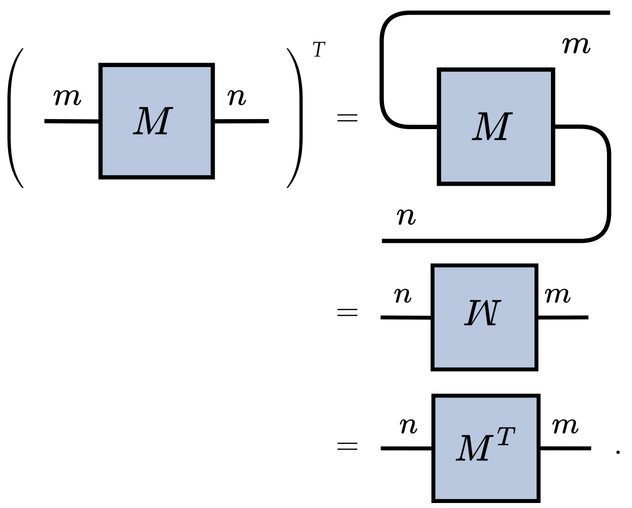
Remark. We can get an intuition on why this process works by looking back to Eqs. (2.7) and (2.8), and realizing that bending the right wire (corresponding to the column dimension) clock or anti-clock wise is equivalent to writing $|w_j\rangle \otimes |v_i\rangle$ or $|v_i\rangle \otimes |w_j\rangle$, respectively.

In addition, we can ask ourselves "*What would happen if instead we bend the left wire towards the right side?*". In fact, this transformation results in a different vector known as the *dual-vector* $\langle v|$. For a more detailed description of this duality, known as *Penrose wire bending duality*, see Section 2 of [25].

A second illustrative example of the use of this graphical language is the transpose operation. For a matrix $M = \sum_{i,j} m_{i,j} |v_i\rangle\langle w_j|$, this operation is defined as

$$M^T = \sum_{i,j} m_{i,j} |w_j\rangle\langle v_i| \ . \tag{2.17}$$

Based on the intuition we just gained from the previous example, we can deduce how the diagram would look like for this operation. The exact procedure is depicted as

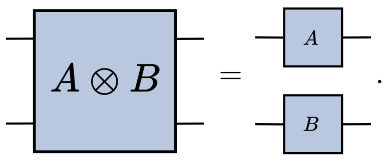


(2.18)

In this transformation, we start by bending the left wire of M to the right, and the right wire to the left. If we then *pull* the wires to *straighten* the tensor, we obtain an upside-down version of our matrix. We can finish by relabeling our tensor as the transpose of the original matrix M^T, resulting in the last equality.

Remark. Notice that in this case, bending both wires in the opposite direction (anti-clockwise) would result in the same diagram! Many interesting implications and applications stem from this simple observation. To learn more about *equivalence classes* in tensor diagrams see section 2 of [25].

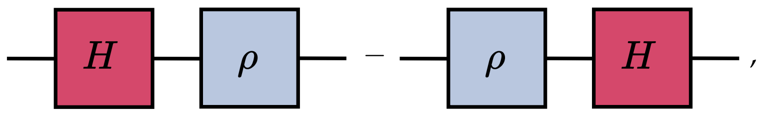
Although there is so much more to learn to unveil the full potential of these diagrammatic representations, we finish this introductory section by highlighting another structure that is used very often in what follows, namely the creation of *multi-partite tensors*. We construct the composite system of two matrices A and B, as $A \otimes B$, and represent it as the vertical juxtaposition of their tensor diagrams



(2.19)

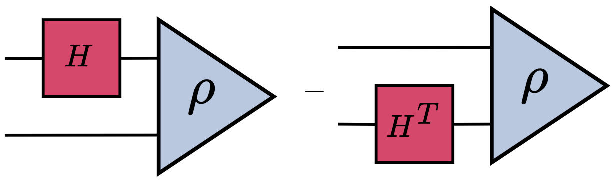
2.3. Tensor networks and the Lindbladian master equation

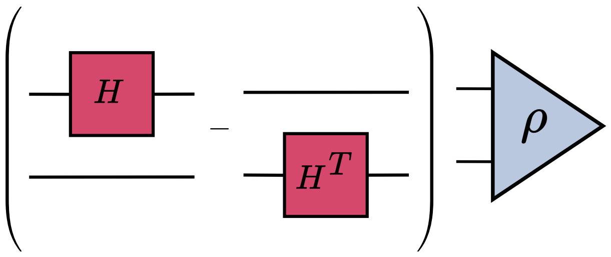
Now that we have developed intuition for the building blocks of diagrammatic representations used in this work, we can delve into the subject matter and apply this knowledge to obtain useful representations for equations of interest. First, we look back to the Lindblad operator from Eq. (2.15), or rather to its vectorization which we introduced back in Eq. (2.6) but did not define. We will now derive each vectorized term using what we have learned about the tensor diagrammatic language. The Hamiltonian term (commutator) would be represented by the diagram

$H\rho - \rho H =$ 

(2.20)

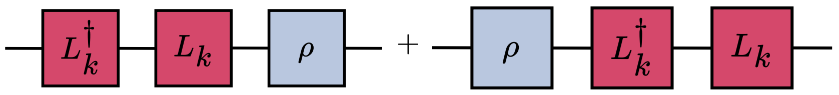
where we have omitted the prefactors for simplicity, and which we will continue doing for the following diagrams. We can vectorize each term by grabbing the rightmost leg and bending it clockwise, just like we did in Eq. (2.16).

$(H \otimes \mathbb{1} - \mathbb{1} \otimes H^T)\,|\rho\rangle\rangle =$ 

$\hat{\mathcal{H}}\,|\rho\rangle\rangle =$ 

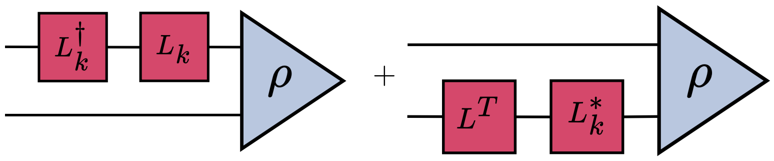
(2.21)

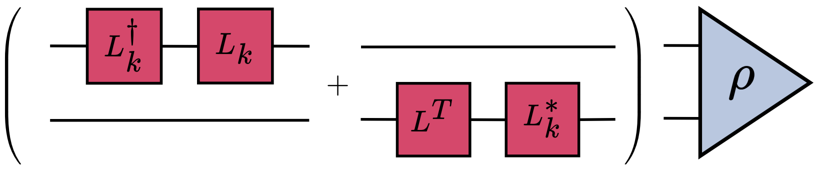
Notice that in the second term we used the relabeling of the upside-down term for H, just like in Eq. (2.18). This yields the vectorization $\hat{\mathcal{H}}$ of the commutator term. Similarly, we look at the anti-commutator term from the dissipative component

$L_k^\dagger L_k \rho + \rho L_k^\dagger L_k =$  .

(2.22)

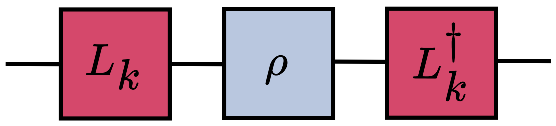
And its vectorized representation is

$(L_k^\dagger L_k \otimes \mathbb{1} + \mathbb{1} \otimes L_k^T L_k^*)\,|\rho\rangle\rangle =$ 

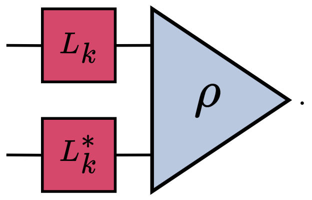
$=$  ,

(2.23)

where we used $(L_k^\dagger L_k)^T = L_k^T L_k^*$ for the second term of the anti-commutator. Then, the first dissipative term, which we will soon recognize as a *quantum channel*, has the following diagram representation:

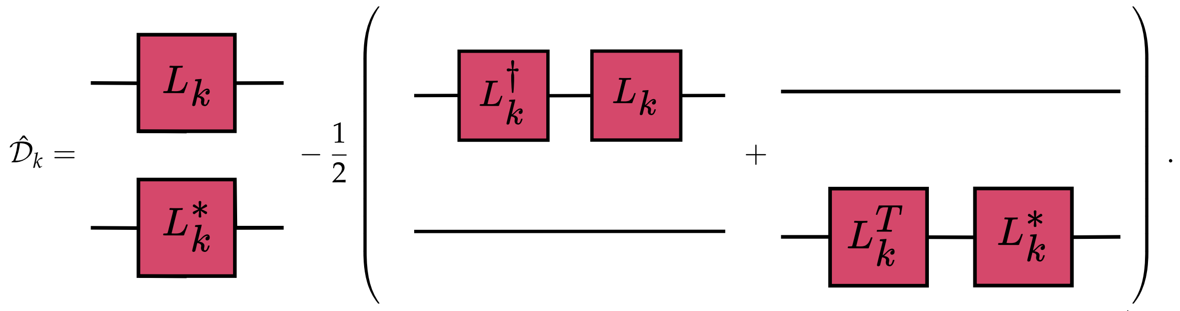
$L_k \rho L_k^\dagger =$  (2.24) with the corresponding vectorization as:

$$(L_k \otimes L_k^*)\,|\rho\rangle\rangle =$$



(2.25)

Putting together Eqs. (2.23) and (2.25) yields the vectorization of a single Lindblad operator L_k as

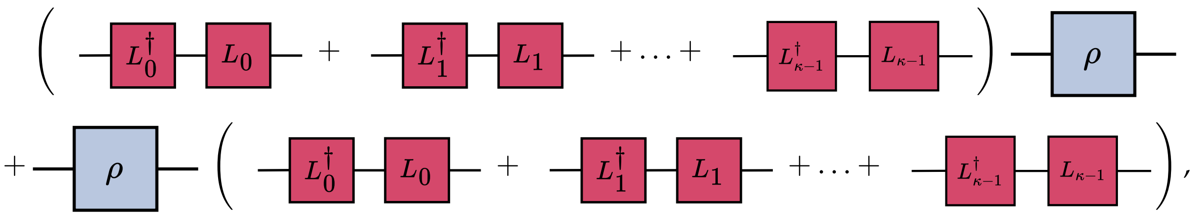


(2.26)

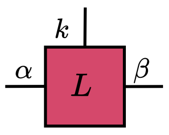
Therefore, assembling together Eqs. (2.21) and (2.26) we obtain an expression for the vectorization of the Lindbladian operator $\mathcal{L}$,

$$\begin{aligned}\hat{\mathcal{L}} &= -i(H \otimes \mathbb{1} - \mathbb{1} \otimes H^T) + \sum_k \left[L_k \otimes L_k^* - \frac{1}{2}(L_k^\dagger L_k \otimes \mathbb{1} + \mathbb{1} \otimes L_k^T L_k^*) \right] \\ &= \hat{\mathcal{H}} + \hat{\mathcal{D}}\,. \end{aligned} \tag{2.27}$$

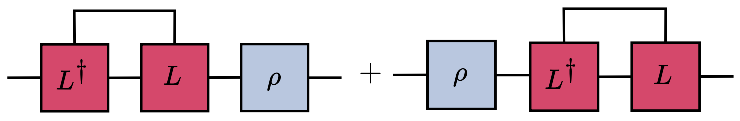
Notice, however, that the diagrams we introduced correspond to a single instance of all the jump operators L_k, so in order to obtain the full vectorized diagram associated with $\hat{\mathcal{D}}$ we ought to sum over all of them. Writing out the sum for the anti-commutator term would result in the following diagram:



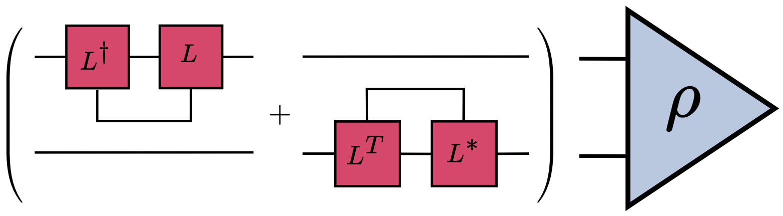
where we assumed there are κ-many jump operators. Fortunately, we can leverage the power of tensor representations to depict this more efficiently. Similarly of how we create the rank-three tensors in an MPS representation from an ensemble of matrices (see Section 3.1 of [29]), we can group all the jump operators $L_k \in \mathbb{C}^{d\times d}$, $d \times d$ dimensional matrices, into a rank-three tensor with dimensions $\kappa \times d \times d$

$$L^k_{\alpha,\beta} =$$ 
. (2.28)

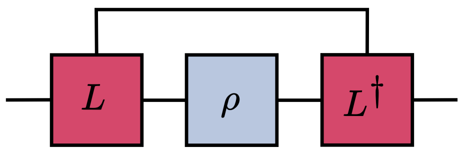
The compressed representation can be obtained by summing over index k, i.e. contracting this index,


(2.29)

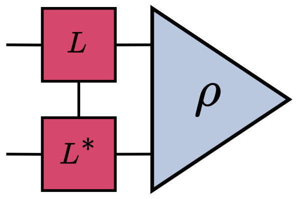
with the corresponding vectorized representation,


. (2.30)

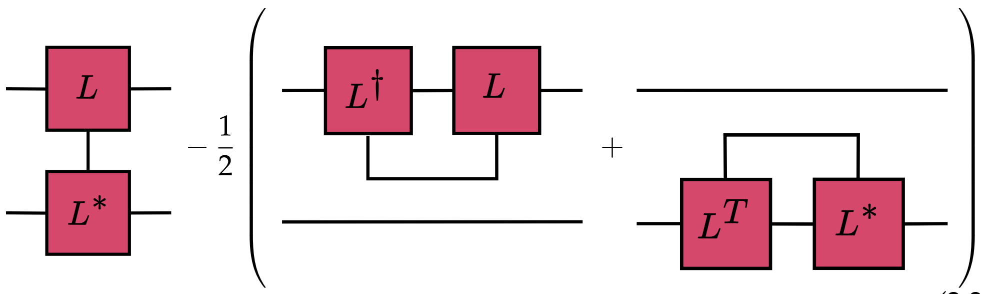
By the same token, the sum over all operators in the first term of $\hat{\mathcal{D}}$ is


(2.31)

and its vectorization


. (2.32)

We can then write the diagram for $\hat{\mathcal{D}}$ as

$$\hat{\mathcal{D}} = \sum_k \hat{\mathcal{D}}_k =$$ 
. (2.33)

2.4. Tensor networks and quantum channels

In the previous sections, we introduced the concept of *quantum channels* as the CPTP maps corresponding to the evolution of an open quantum system, i.e. maps with a valid density operator as their output. In this section, we will review three mathematical representations for these maps, and employ the diagrammatic techniques we have learned to transform among them. First, we formally introduce the Kraus [30] or operator-sum [31] representation.

Definition 2.4.1 (Kraus/Operator-sum representation)**.** Kraus' theorem [30] states that a linear map $\Phi \in \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ is CP if and only if it can be written in the Kraus representation

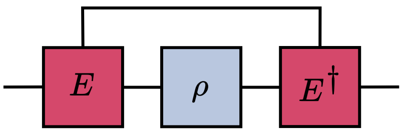
$$\Phi(\rho) = \sum_{q=0}^{K-1} E_q \rho E_q^\dagger\,, \tag{2.34}$$

where $E_q \in \mathscr{L}(\mathscr{H}_1, \mathscr{H}_2)$ are the *Kraus operators*. In addition, we can impose the *completeness relation*

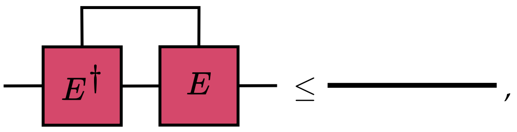
$$\sum_{q=0}^{K-1} E_q^\dagger E_q \leq \mathbb{1}\,, \tag{2.35}$$

where the equality corresponds to Φ being TP.

Analogously to the procedure followed in Eqs. (2.28), (2.29), and (2.31), we can group all the E_q matrices into a rank-three tensor E. The tensor representation of Eq. (2.34) is then

$\Phi(\rho) =$  (2.36)

and the completeness relation diagram is

 (2.37)

where we made use of the diagram of the identity operator

$$\mathbb{1} = \sum_i |i\rangle\langle i| = \text{———}\,. \tag{2.38}$$

Remark. One must not fall for the fallacy of thinking that the number of jump operators L_k, is the same as the number of Kraus operators E_q. To emphasize this distinction, we write the total number of operators as κ and K, for the former and the latter, respectively.

We are in particular interested in the case where $\mathscr{H}_1 \cong \mathscr{H}_2 \cong \mathbb{C}^d$, where the maximum number of Kraus operators needed to specify Φ is d^2, and the minimum is 1, corresponding to a unitary evolution. We observe the implications of this relation later on in § 5.2.

The matrices $\{E_q\}$ do not uniquely determine Φ, as it will become more evident when we introduce the other channel representations. All the possible sets of Kraus operators are related to each other by a **unitary transformation** [11], and it is in fact this degree of freedom that we will exploit in our main algorithm! However, a particularly useful instance of these sets of Kraus operators is the *Canonical Kraus representation* [25, 32], which satisfies the orthogonality condition

$$\mathrm{Tr}\left[K_\alpha^\dagger K_\beta\right] = \lambda_\alpha \delta_{\alpha\beta}\,. \tag{2.39}$$

Remark. Each of the Kraus operators in this unique representation corresponds to the eigenvectors of the Choi matrix to be defined in Def. (2.4.3).

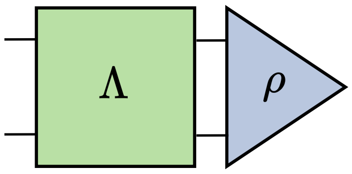
The second channel representation was previously introduced in § 2.1 when studying the solution of the Lindblad equation, Eq. (2.5), namely the *superoperator* representation.

Definition 2.4.2 (Superoperator representation)**.** The superoperator Λ of a quantum channel $\Phi \in \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ is the linear map arising from the vectorization of $\Phi(\rho)$, and is defined as

$$\Lambda : \mathscr{H}_1 \otimes \mathscr{H}_1 \to \mathscr{H}_2 \otimes \mathscr{H}_2 : |\rho\rangle\rangle \to |\Phi(\rho)\rangle\rangle, \tag{2.40}$$

with the diagrammatic depiction

$$|\Phi(\rho)\rangle\rangle = \tag{2.41}$$

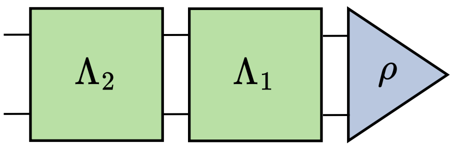


Expressing our quantum channel through its superoperator representation provides us with a practical composition method. Given two quantum channels Φ_1 and Φ_2, giving rise to the superoperators Λ_1 and Λ_2, respectively, applying them to the state ρ

$$\Phi_2 \circ \Phi_1(\rho) \equiv \Phi_2\left(\Phi_1\left(\rho\right)\right) \tag{2.42}$$

can be expressed in superoperator form as

$$|\Phi_2 \circ \Phi_1(\rho)\rangle\rangle = \Lambda_2 \cdot \Lambda_1 \, |\rho\rangle\rangle = \tag{2.43}$$



where $\cdot$ is the standard matrix multiplication, which can be seamlessly implemented numerically. For the last representation, we will make use of a ubiquitous transformation in quantum information theory, namely the *Choi-Jamiołkowski isomorphism* [33, 34], also known as channel-state duality.

Definition 2.4.3 (Choi representation)**.** The Choi-Jamiołkowski isomorphism defines a correspondence between a linear map $\Phi \in \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ and a linear operator $\mathcal{C} \in \mathscr{L}(\mathscr{H}_2 \otimes \mathscr{H}_1)$, which we call the *Choi matrix* or *Choi state*. The relation between them can be obtained by thinking of the map Φ as acting on the first half of the unnormalized Bell-state, $|\phi^+\rangle := \sum_i |i\rangle \otimes |i\rangle \in \mathscr{H}_1 \otimes \mathscr{H}_1$, while leaving the second half untouched. The resulting state is

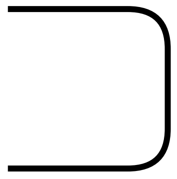
$$\begin{aligned} \mathcal{C} &= (\Phi \otimes \mathrm{id})|\phi^+\rangle\langle\phi^+| , \qquad \mathrm{id} \in \mathscr{T}(\mathscr{H}_1) && (2.44) \\ &= (\Phi \otimes \mathrm{id}) \left(\sum_i |i\rangle \otimes |i\rangle \sum_j \langle j| \otimes \langle j| \right) \\ &= \sum_{i,j} \Phi(|i\rangle\langle j|) \otimes |i\rangle\langle j| \ . && (2.45) \end{aligned}$$

The Choi-Jamiołkowski isomorphism is analogous to the vectorization transformation, but instead of mapping linear operators in $\mathscr{L}(\mathscr{H}_1, \mathscr{H}_2)$ to vectors in $\mathscr{H}_2 \otimes \mathscr{H}_1$, it is the map

$$\mathscr{T}(\mathscr{H}_1, \mathscr{H}_2) \to \mathscr{L}(\mathscr{H}_2 \otimes \mathscr{H}_1) : \Phi \to \mathcal{C} \, . \tag{2.46}$$

Remark. It is worth noting that Eqs. (2.44–2.46) assume a row-wise vectorization, as defined in Def. (2.1.1). The respective relations for the column-wise vectorization could be obtained by assuming Φ acts on the second half of the Bell-state, i.e. $(\mathrm{id} \otimes \Phi)$ so that $\mathcal{C} \in \mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2)$ instead.

In order to construct the diagrammatic representation of $\mathcal{C}$ we first introduce the (very fundamental) diagram of the unnormalized Bell-state

$$|\phi^+\rangle =$$  . (2.47)

To see why this diagram corresponds to the Bell state $|\phi^+\rangle$, we can recall the definition of $\mathbb{1}$, Eq. (2.38), and by looking at its vectorization

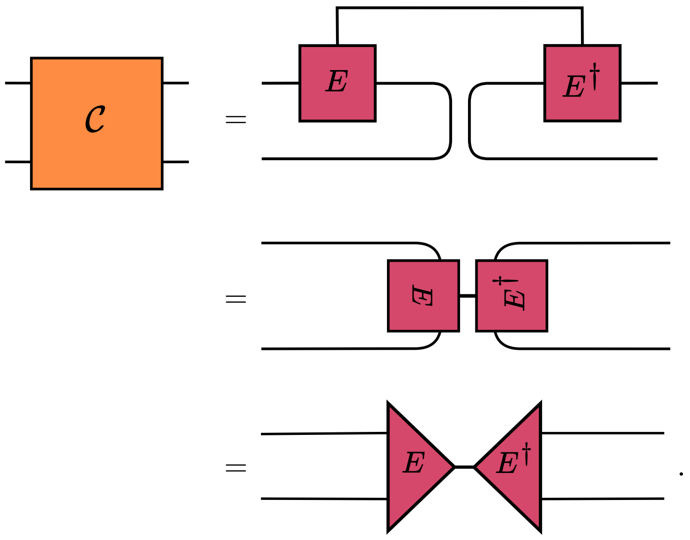
$$|\mathbb{1}\rangle\rangle = \sum_i |i\rangle \otimes |i\rangle$$

we obtain the relation

$$|\mathbb{1}\rangle\rangle = |\phi^+\rangle. \tag{2.48}$$

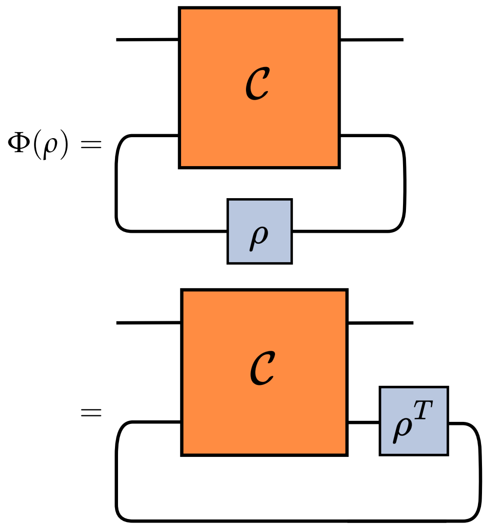
Hence, the diagram in Eq. (2.47) is the vectorization of the identity diagram in Eq. (2.38)! Applying these relations in the definition of $\mathcal{C}$, Eq. (2.44), with the action of Φ depicted by its Kraus operator representation as shown in Eq. (2.34), we obtain the diagram for the

Choi-matrix,



(2.49)

In addition to providing us with a visual representation for $\mathcal{C}$, this diagram shows the relation between the Kraus and the Choi representation. In order to obtain an expression for $\Phi(\rho)$ in terms of the Choi-matrix, we compare the diagrams in Eqs. (2.36) and (2.49) and notice that the *input* legs for the channel correspond to the two bottom legs of $\mathcal{C}$. Hence, we obtain



(2.50)

$$= \mathrm{Tr}_{\mathscr{H}_1}\left[\mathcal{C}(\mathbb{1}_{\mathscr{H}_2} \otimes \rho^T)\right],$$

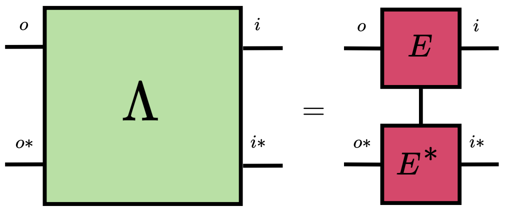
where $\mathrm{Tr}_{\mathscr{H}_1}$ is the partial trace over the subsystem $\mathscr{H}_1$, since $\mathcal{C} \in \mathscr{L}(\mathscr{H}_2 \otimes \mathscr{H}_1)$. This representation is well suited to verify certain properties of quantum channels, such as CP.

Theorem 2.4.1 (Choi's theorem)**.** Given a quantum channel $\Phi \in \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ and $\mathcal{C}$ its Choi-matrix representation, *Choi's theorem* [33] states

$$\Phi \text{ is CP } \iff \mathcal{C} \geq 0\,. \tag{2.51}$$

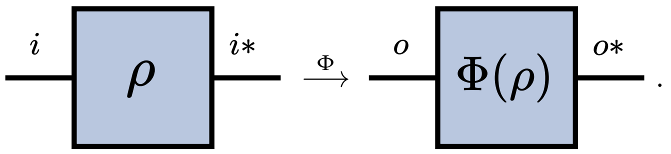
The right-hand side of this equation will be useful during this work as an operational definition for CP maps together with Def. (2.1.5).

Using the tensor networks that we have developed for each representation, it is straightforward to define the relations between each of them. We have already seen how to convert between the Kraus and the Choi-matrix representations in Eq. (2.49). Now, we can follow a similar approach to derive the relation between the Kraus and superoperator representation. If we vectorize Eq. (2.36) and compare it to Eq. (2.41), we obtain



(2.52)

We see the right-hand side is equivalent to superoperator in Eq. (2.32) for the Lindbladian term acting as a quantum channel. We have introduced labels on the tensor legs to make the relations easier to follow. Labels i and o are the *input* and *output* legs of the quantum channel, respectively, while $i*$ and $o*$ are their *conjugated* versions, arising from the vectorization of ρ. This corresponds to labeling the input ρ and output $\Phi(\rho)$ density matrices as:



(2.53)

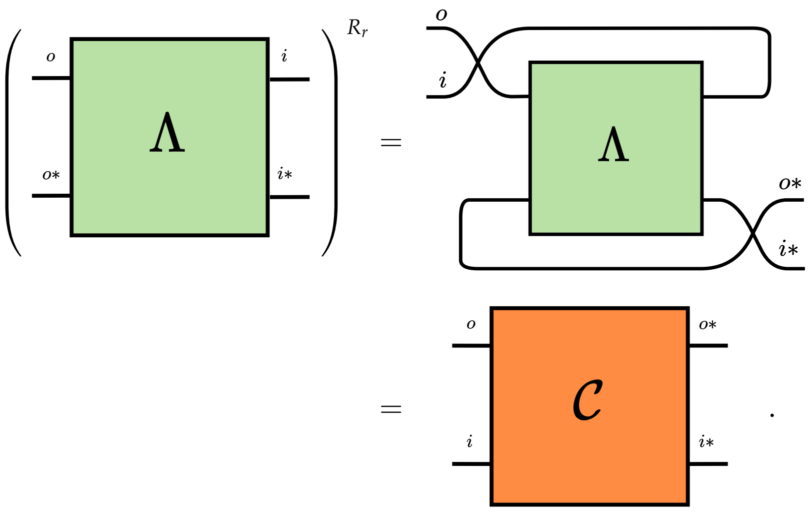
To relate the superoperator with the Choi-matrix, we introduce the bipartite *row-reshuffling* operation [25].

Definition 2.4.4 (Row-reshuffling)**.** Given a bipartite matrix $\Lambda \in \mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2)$ the row-reshuffling transformation R_r is defined as the map

$$R_r : \mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2) \to \mathscr{L}(\mathscr{H}_2 \otimes \mathscr{H}_2, \mathscr{H}_1 \otimes \mathscr{H}_1) : \Lambda \to (\Lambda)^{R_r}\,. \tag{2.54}$$

By writing Λ in its tensor notation, this map can be seen as $\Lambda_{m\mu,n\nu} \to \Lambda_{mn,\mu\nu}$, i.e. the exchange of indices $\mu \leftrightarrow n$.

By applying this transformation to our superoperator, we obtain



(2.55)

These transformations allow us to define a useful algorithm to obtain the Kraus operators of a quantum channel from its superoperator representation, as shown in Alg. (1). In step two of this algorithm we have used the decomposition of a positive-semidefinite matrix $\mathcal{C}$ into its *factors* X and $X^\dagger$

$$\mathcal{C} = XX^\dagger \tag{2.56}$$

as given by Def. (2.1.2). This factorization is known as the *Cholesky decomposition* [35].

Algorithm 1 Superoperator to Kraus

```
Input: $\Lambda$: Quantum channel in superoperator form $\in \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$
Output: $\{E_q\}_q$: rank-three tensor with each $E_q \in \mathscr{L}(\mathscr{H}_1, \mathscr{H}_2)$, the channel's Kraus operators
1: $\mathcal{C}$: Choi representation of the channel $\leftarrow$ Apply row-reshuffling to $\Lambda : \Lambda^{R_r}$.
2: $X \leftarrow$ Cholesky decompose $\mathcal{C}$.
3: $\{E_q\}_q \leftarrow$ Unvectorize and stack columns of $X$: each column is a $|E_q\rangle\rangle$.    ▷ See Eq. (2.49).
```

Remark. In Def. (2.4.1), we noted the Kraus representation is not unique, where two equivalent representations of a channel might even differ in the number of Kraus operators. However, there is a minimal number of Kraus operators a quantum channel may be represented with and is known as the *Choi rank*. The right-hand side of Eq. (2.49) shows the vectorized Kraus tensor $|E_q\rangle\rangle$, where the channel index q corresponds to the Choi rank. Hence, this decomposition yields the minimum number of Kraus operators of the channel. This agrees with the observation that the columns of the factorization X shown in Eq. (2.56) are the vectorization of each Kraus operator, meaning they form an orthogonal set.

2.5. Locally purified tensor network

Due to the limited analytical methods available to describe open quantum systems, numerical methods play a pivotal role towards our understanding of dissipative dynamics. In this light, tensor networks variational ansätze [3] arise as an efficient approach capable of describing several systems of interest, with the Matrix Product States (MPS) [36, 37] as the archetype class used to study one-dimensional pure states.

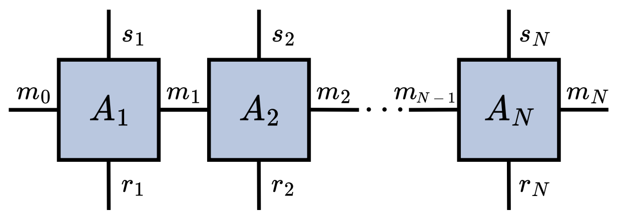
On the other hand, one-dimensional mixed states are still a relatively unexplored frontier by tensor network techniques. A first attempt at this employs the class of Matrix Product Density Operators (MPDO) [4, 38]. However, locally verifying the positivity of the global MPDO is NP-hard [39], so that performing local truncations usually destroys the positive definiteness of the state, yielding numerical instabilities. A second approach uses the so-called *unravelling of the master equation* [1, 40] to simulate mixed states by means of MPS methods, together with stochastic quantum trajectories. This approach comes at the cost of a sampling overhead with the number of trajectories. In this work, we will leverage a third structure introduced in [4, 5], known as the *Locally Purified Density Operator* (LPDO).

Definition 2.5.1 (Locally Purified Density Operator)**.** Given a density matrix ρ, its local purification operator X satisfies

$$\rho = XX^{\dagger}, \tag{2.57}$$

and its variational tensor network representation for N sites is

$$[X]^{s_1,\dots,s_N}_{r_1,\dots,r_N} = \sum_{m_1,\dots,m_{N-1}} A^{[1]s_1,r_1}_{m_0,m_1} A^{[2]s_2,r_2}_{m_1,m_2} \dots A^{[N]s_N,r_N}_{m_{N-1},m_N}$$

$=$ 

. (2.58)

We use the following notation for the tensor legs' dimensions

$$[n] := \{1, 2, \dots, n\}, \tag{2.59}$$

thus, we can write each rank-four tensor $\{A^{[l]}\}_{l=1}^{N}$ as

$$A^{[l]} = \left(A^{[l]s_l,r_l}_{m_{l-1},m_l}\right)_{s_l\in[d],\, r_l\in[K],\, m_{l-1}\in[D_{l-1}],\, m_l\in[D_l]}, \tag{2.60}$$

where s_l is the *physical* index, r_l is the *Kraus* or *rank* index, and m_l is the *bond* index, with dimensions d, K, and D, respectively. For uniformity, the first and last tensor have bond dimensions $m_0 = m_N = 1$. Notice how the representation of ρ in terms of tensors $\{A^{[l]}\}$ is highly non-unique, since in addition to the usual gauge freedom present in MPS along the bond index, this structure has a gauge freedom along the Kraus index. The latter is employed in [6] to achieve a mixed normalized form of the $A^{[l]}$ tensors.

Remark. Notice that Eq. (2.57) is a result of Def. (2.1.2) for the mixed state ρ, so our global tensor network will be **positive** by construction. Contrary to Def. (2.1.2), here X is in general a rectangular matrix $X \in \mathbb{C}^{d^N \times K^N}$, with d^N the full Hilbert space dimension of the N-body system, and its $\text{rank}(\rho) = K^N$.

The construction of the LPDO representation from a general MPDO is studied in [5]. However, if we start with a pure state represented by an MPS —as will be the case in this work— to obtain the LPDO representation all we need is to add on each site l an extra index r_l with dimension $K = 1$ [41]. Werner et al. [6] described a practical algorithm that we will employ to simulate the Markovian dynamics of one-dimensional open quantum systems in the LPDO structure. Similar to the Time Evolving Block Decimation (TEBD) [42] scheme, for a small time-step τ, this algorithm splits the superoperator $\mathrm{e}^{\tau\hat{\mathcal{L}}}$ from Eq. (2.6) into several Trotter-Suzuki layers [43] made up of mutually commuting terms from the coherent $\hat{\mathcal{H}}$ and dissipative $\hat{\mathcal{D}}$ components of $\hat{\mathcal{L}}$, see Eq. (2.27). As a point of comparison, Alg. (2) outlines the scheme followed when the Hamiltonian terms act on nearest-neighbors and the couplings L_k act on single sites.

Algorithm 2 LPDO Dynamics: Single-site Couplings

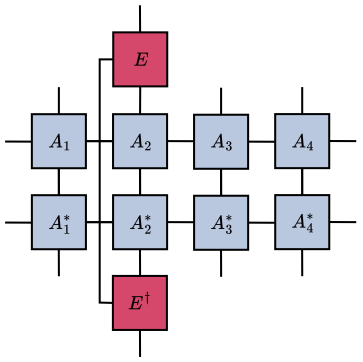
```
Input:
X: purification operator in LPDO form, $\hat{\mathcal{L}}$: vectorized Lindblad operator
$\tau$: final evolution time
Output: $X_\tau$ : evolved purification operator in LPDO form
 1: $\mathrm{e}^{\tau\hat{\mathcal{H}}_o/2}\mathrm{e}^{\tau\hat{\mathcal{H}}_e/2}\mathrm{e}^{\tau\hat{\mathcal{D}}}\mathrm{e}^{\tau\hat{\mathcal{H}}_e/2}\mathrm{e}^{\tau\hat{\mathcal{H}}_o/2} \leftarrow$ Use second-order Trotter to approximate $\mathrm{e}^{\tau\hat{\mathcal{L}}}$.
 2: Use TEBD to perform coherent dynamics for $\hat{\mathcal{H}}_o$ and $\hat{\mathcal{H}}_e$ layers.   ▷ See Fig. (2.3a)
 3: Update each $A^{[l]}$: contract two-site tensors from TEBD with local $A^{[l]}$.
 4: Use SVD to recover purified tensors and to compress along bond index.
 5: for $l \leftarrow 1$ to $N$ do
 6:     Use $\mathrm{e}^{\tau\hat{\mathcal{D}}} = \bigotimes_l \mathrm{e}^{\tau\hat{\mathcal{D}}^{[l]}}$ for dissipative layer.   ▷ Each $\hat{\mathcal{D}}^{[l]}$ is defined as in Eq. (2.27).
 7:     Find the single-site Kraus tensor $E_q^{[l]}$ from $\mathrm{e}^{\tau\hat{\mathcal{D}}^{[l]}}$ using Alg. (1).
 8:     Contract $E_q^{[l]}$ with $A^{[l]}$: merge the Kraus leg $r_l$ with the channel index $q$.   ▷ See Fig. (2.3b)
 9:     Use SVD to compress along Kraus index.
10: end for
11: Repeat steps 2-4.
```

The approximation arising from this second-order scheme is

$$\mathrm{e}^{\tau\hat{\mathcal{L}}} = \mathrm{e}^{\tau\hat{\mathcal{H}}_o/2}\mathrm{e}^{\tau\hat{\mathcal{H}}_e/2}\mathrm{e}^{\tau\hat{\mathcal{D}}}\mathrm{e}^{\tau\hat{\mathcal{H}}_e/2}\mathrm{e}^{\tau\hat{\mathcal{H}}_o/2} + O(\tau^3)\,, \tag{2.61}$$

where $\hat{\mathcal{H}}_o$ and $\hat{\mathcal{H}}_e$ are the odd and even terms of $\hat{\mathcal{H}}$. As discussed in [6], these coherent layers can be implemented with the usual nearest-neighbor TEBD for Hamiltonian evolution, shown in Fig. (2.3a). Contracting $\mathrm{e}^{-i\tau H^{[l,l+1]}}$ with $A^{[l]}$ and $A^{[l+1]}$ increases the bond dimension D of index m_l. On the other hand, the action of the local Kraus tensor $E_q^{[l]}$ arising from the dissipative layer $\hat{\mathcal{D}}^{[l]}$ increases the Kraus dimension K of r_l after merging it with the channel rank index q. The latter can be understood by looking at the action of a local Kraus channel acting on a mixed state ρ in its LPDO form

$$\Phi(\rho) = \sum_q E_q^{[l]} \rho E_q^{[l]\dagger} = \qquad , \tag{2.62}$$



where the diagram shows the case when $l = 2$. The action of $E_q^{[l]}$ on the purification X is shown in Fig. (2.3b), resulting in the increase of the Kraus dimension of $A^{[l]}$. Both the increase in m_l and r_l can be taken care of via a SVD-based compressing scheme along the corresponding dimension [6]. However, due to the local nature of this truncation, it cannot prevent the exponential growth of the Kraus dimensions in the LPDO structure [44, 45].

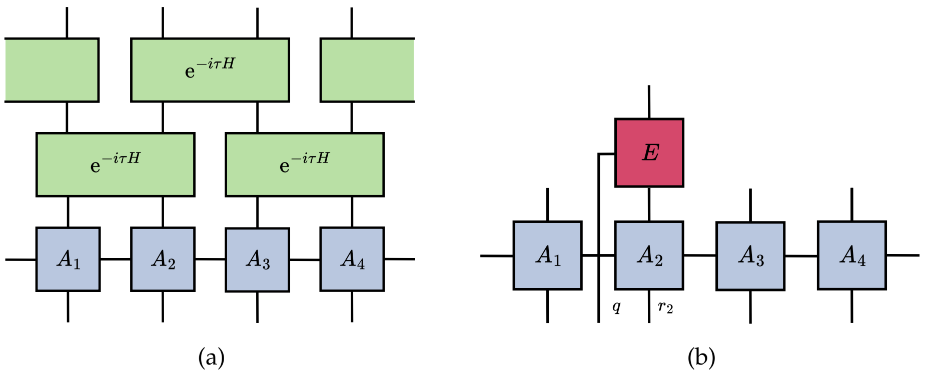


Figure 2.3.: Markovian time evolution of the LPDO for a four-site mixed state. Panel (a) shows the coherent evolution, where the tensors correspond to the $\hat{\mathcal{H}}_o$ and $\hat{\mathcal{H}}_e$ layers trotterized from Eq. (2.61) in a TEBD-like manner. Panel (b) shows the contraction of a local Kraus channel, resulting in the increase of the Kraus dimension of $A^{[l]}$.

A main advantage this scheme provides is the ability to track the Frobenius norm error introduced at every compression step and due to the Trotter approximation. Appendix D of [6] provides an explicit bound for the trace norm error introduced on the whole mixed state ρ. We now shift our focus to the main case of interest in this work, i.e. where both the Hamiltonian $H^{[l,l+1]}$ and jump operators $L^{[l,l+1]}$ act non-trivially only on nearest-neighbors. Therefore, the Lindblad superoperator acting on all N sites can be written as

$$\hat{\mathcal{L}} = \sum_{l=1}^{N} \hat{\mathcal{L}}^{[l,l+1]}, \tag{2.63}$$

and we use *periodic boundary conditions* (PBC) such that for $l = N$ we get $\hat{\mathcal{L}}^{[N,1]}$, similarly to Fig. (2.3a). Each $\hat{\mathcal{L}}^{[l,l+1]}$ is a $d^{2N} \times d^{2N}$ superoperator acting on the whole system, but only non-trivially on sites l and $l+1$, and takes the form shown in Eq. (2.27). From Lemma 4 in [6], the Trotter-Suzuki approximation for the Markovian dynamics for nearest-neighbors couplings is

$$e^{\tau\hat{\mathcal{L}}} = e^{\tau\hat{\mathcal{L}}_o/2} e^{\tau\hat{\mathcal{L}}_e} e^{\tau\hat{\mathcal{L}}_o/2} + O(\tau^3)\,, \tag{2.64}$$

with the odd and even terms made up of mutually commuting terms

$$\hat{\mathcal{L}}_o := \sum_l \hat{\mathcal{L}}^{[2l-1,2l]} \quad \text{and} \quad \hat{\mathcal{L}}_e := \sum_l \hat{\mathcal{L}}^{[2l,2l+1]}, \tag{2.65}$$

which allows us to write the exponentials as

$$e^{\tau\hat{\mathcal{L}}_o} = \prod_l e^{\tau\hat{\mathcal{L}}^{[2l-1,2l]}} \quad \text{and} \quad e^{\tau\hat{\mathcal{L}}_e} = \prod_l e^{\tau\hat{\mathcal{L}}^{[2l,2l+1]}}. \tag{2.66}$$

By concentrating on the dissipative terms and employing Eq. (2.33) to define the $d^4 \times d^4$ two-local dissipative superoperators $\hat{\mathcal{D}}^{[2l-1,2l]}$ and $\hat{\mathcal{D}}^{[2l,2l+1]}$ for the odd and even layers, respectively, we obtain the following expressions

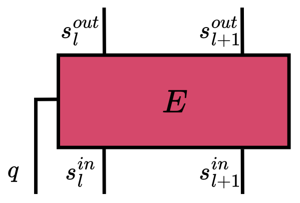
$$e^{\tau\hat{\mathcal{L}}_o} = \bigotimes_l e^{\tau\hat{\mathcal{D}}^{[2l-1,2l]}} \quad \text{and} \quad e^{\tau\hat{\mathcal{L}}_e} = \bigotimes_l e^{\tau\hat{\mathcal{D}}^{[2l,2l+1]}}. \tag{2.67}$$

See § 3.1 for an in-detail treatment of this expression. The scheme for the Markovian dynamics under nearest-neighbor Lindblad is then outlined in Alg. (3). We take advantage of Eq. (2.67) to focus on the two-local superoperators acting on odd and even sites

$$e^{\tau\hat{\mathcal{D}}^{[2l-1,2l]}} \text{ and } e^{\tau\hat{\mathcal{D}}^{[2l,2l+1]}}, \quad \forall l \in [1, N/2]. \tag{2.68}$$

Contrary to the single-site channels from Alg. (2), the Kraus tensor $E_q^{[l,l+1]}$ obtained from these superoperators acts on two sites at a time,

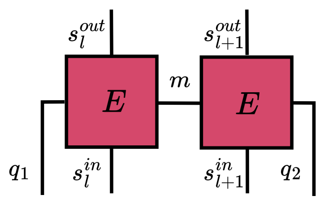
$$E_q^{[l,l+1]} = \left(E_{q,s_l^{in},s_{l+1}^{in}}^{[l,l+1]\,s_l^{out},s_{l+1}^{out}} \right)_{s_l^{out},s_{l+1}^{out},s_l^{in},s_{l+1}^{in} \in [d]}^{q \in [k]} = \tag{2.69}$$



.

So, its contraction increases the dimensions of more than one index of X at a time, namely m, r_l, and r_{l+1}. To preserve the purified structure, we follow the procedure shown in Fig. (2.4) to split the tensor $E_q^{[l,l+1]}$ into $E_{q_1,m}^{[l]}$ and $E_{q_2,m}^{[l+1]}$ [6], so that

$$E_{q,s_l^{in},s_{l+1}^{in}}^{[l,l+1]\,s_l^{out},s_{l+1}^{out}} = \sum_{m=1}^{D'} E_{q_1,m}^{[l]s_l^{out},s_l^{in}} \cdot E_{q_2,m}^{[l+1]s_{l+1}^{out},s_{l+1}^{in}} = \tag{2.70}$$



,

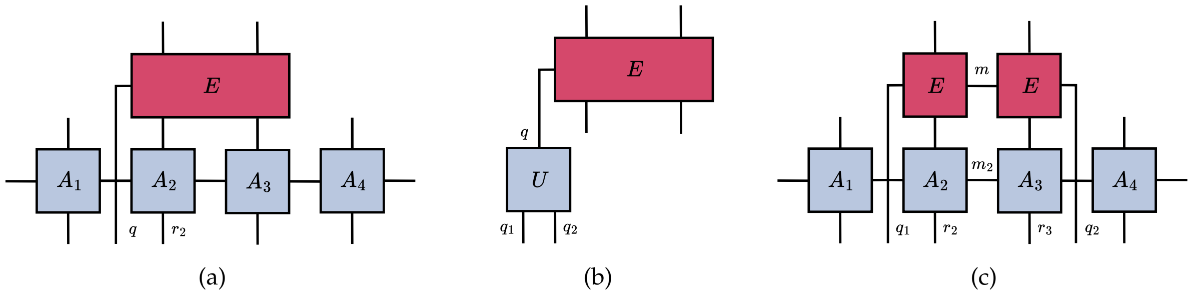


Figure 2.4.: Markovian dynamics of a four-site mixed state in LPDO form. Panel (a) shows the Kraus tensor E being contracted with sites two and three of X. Panel (b) outlines the application of a unitary U on the Kraus rank q, in order to minimize the resulting Kraus bond dimension m. Finally, panel (c) shows the contraction of the single-site Kraus channels, hence conserving the locally purified structure.

with the individual Kraus ranks $q_1 \in [k_1]$ and $q_2 \in [k_2]$ satisfying $k = k_1 \cdot k_2$.

In addition, as outlined in Appendix C of [6], Fig. (2.4b) shows the intermediate optimization of the rank distribution $q \to q_1, q_2$ and the minimization of the resulting *Kraus bond* m by leveraging the unitary gauge freedom with respect to the channel rank q, which we also discussed when introducing the Kraus channel representation in Def. (2.4.1).

2.6. A brief detour on multivariate calculus

We now introduce some basic definitions from multivariate calculus that will come in handy to understand the generalized concepts defined on Manifolds in § 2.7. Here, we follow majorly the definitions presented in Appendix A.5 of [8].

Definition 2.6.1 (Directional derivative)**.** The directional derivative of a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in \mathbb{R}^n$ along a vector $v \in \mathbb{R}^n$ is given by

$$Df(x)[v] = \nabla_v f(x) := \lim_{h \to 0} \frac{f(x + hv) - f(x)}{h} . \tag{2.71}$$

It generalizes the idea of a *partial derivative*, which corresponds to the case where v points along one of the coordinate axis. The directional derivative can be thought as the rate of change of f when we "move" in direction of v at x. If f is differentiable at x, then for any unit vector v we have

$$\nabla_v f(x) = \langle \nabla f(x), v \rangle , \tag{2.72}$$

where $\nabla f(x)$ is the *gradient* of f at x.

Definition 2.6.2 (Gradient)**.** The gradient of a real valued function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in \mathbb{R}^n$ is the *vector field*, i.e. a vector-valued function, depicted as a "field of arrows", one

Algorithm 3 LPDO Dynamics: Nearest-neighbor Couplings

```
Input:
X: purification operator in LPDO form, $\hat{\mathcal{L}}$: Lindblad operator vectorized
τ: final time of evolution
Output: $X_\tau$ : evolved purification operator in LPDO form
 1: $e^{\tau\hat{\mathcal{L}}_o/2}e^{\tau\hat{\mathcal{L}}_e}e^{\tau\hat{\mathcal{L}}_o/2} \leftarrow$ Use second-order Trotter to approximate $e^{\tau\hat{\mathcal{L}}}$.
 2: for $l \leftarrow 1$ to $N/2$ do
 3:     Calculate $e^{\tau\hat{\mathcal{D}}^{[2l-1,2l]}/2}$ numerically.
 4:     Find the two-site Kraus tensor $E_q^{[2l-1,2l]}$ from $e^{\tau\hat{\mathcal{D}}^{[2l,2l+1]}/2}$ using Alg. (1).
 5:     Find the optimal unitary along $q$ minimizing $m$, such that $E_q^{[2l-1,2l]}$ splits into $E_{q_1,m}^{[2l-1]}$
      and $E_{q_2,m}^{[2l]}$.
 6:     Contract $E_{q_1,m}^{[2l-1]}$ with $A^{[2l-1]}$ and $E_{q_2,m}^{[2l]}$ with $A^{[2l]}$:
 7:         merge the Kraus bond $m$ with the tensor bond $m_{2l-1}$, and
 8:         the Kraus legs $r_{2l-1}$ and $r_{2l}$ with the channel indices $q_1$ and $q_2$, respectively.
 9: end for
10: Repeat step 2 to 6 for even sites with $e^{\tau\hat{\mathcal{D}}^{[2l,2l+1]}}$.
11: Repeat step 2 to 6 for odd sites.
```

assigned to every point of the space. Expressed in a coordinate system this is

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}, \tag{2.73}$$

such that $\nabla f(x) : \mathbb{R}^n \to \mathbb{R}^n$. The direction of $\nabla f(x)$ is the direction of the fastest increase of f from x, and its magnitude is the rate of increase.

Definition 2.6.3 (Hessian)**.** The square matrix containing all the second-order partial derivatives of a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is known as the Hessian Matrix,

$$H_f = H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}. \tag{2.74}$$

Notice that this is a *matrix-valued* function. We can evaluate this object at $x \in \mathbb{R}^n$, such that $H_f : \mathbb{R}^n \to \mathbb{R}^{n\times n} : x \to H_f(x)$. We can then think of $H_f(x)$ as the linear map

$$H_f(x) : \mathbb{R}^n \to \mathbb{R}^n : v \to H_f(x) \cdot v\,. \tag{2.75}$$

The i-th column of $H_f(x)$ tells us how the basis vectors $e_i \in \mathbb{R}^n$ transform under this matrix

$$H_f(x)_i = H_f(x)e_i\,. \tag{2.76}$$

In other words, we can reveal the i-th column of $H_f(x)$ by multiplying it with the i-th basis vector. By using the *Jacobian J* applied to the gradient of f, we can write the Hessian as $H_f(x) = J(\nabla f(x))^T$.

Definition 2.6.4 (Jacobian)**.** The generalization of the gradient for a vector field $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is known as the Jacobian, and its $m \times n$ matrix representation evaluated at $x \in \mathbb{R}^n$ is

$$D\mathbf{f}(x) = J_{\mathbf{f}}(x) = J(\mathbf{f}(x)) = \begin{pmatrix} \frac{\partial \mathbf{f}_1}{\partial x_1}(x) & \frac{\partial \mathbf{f}_1}{\partial x_2}(x) & \cdots & \frac{\partial \mathbf{f}_1}{\partial x_n}(x) \\ \frac{\partial \mathbf{f}_2}{\partial x_1}(x) & \frac{\partial \mathbf{f}_2}{\partial x_2}(x) & \cdots & \frac{\partial \mathbf{f}_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_m}{\partial x_1}(x) & \frac{\partial \mathbf{f}_m}{\partial x_2}(x) & \cdots & \frac{\partial \mathbf{f}_m}{\partial x_n}(x) , \end{pmatrix} \tag{2.77}$$

where $\mathbf{f}_i$ and x_i are the i-th component of $\mathbf{f}$ and x, respectively. By looking at each $\mathbf{f}_i$ as a real-valued function, we can write the Jacobian as

$$J_f(x) = \begin{bmatrix} \nabla^T \mathbf{f}_1 \\ \nabla^T \mathbf{f}_2 \\ \vdots \\ \nabla^T \mathbf{f}_m \end{bmatrix} = \nabla \mathbf{f}(x)^T , \tag{2.78}$$

where the latter equality is slight abuse of notation of the differential operator ∇. Notice how the basis vector e_i transforms under the Jacobian: $\frac{\partial \mathbf{f}}{\partial x_i}(x) = J_{\mathbf{f}}(x)e_i$. Finally, the best linear approximation of $\mathbf{f}(y)$ in the neighborhood of x is the map $\phi : y \to \mathbf{f}(x) + J_{\mathbf{f}}(x)(y - x)$.

We can then use these operators in the local *Taylor expansion* of f as

$$f(x + \Delta x) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T H_f(x) \Delta x + \mathcal{O}(\Delta x^3) . \tag{2.79}$$

In the next three sections, we will develop the equivalent components on manifolds to define an equivalent expression to Eq. (2.79). Another "classical" definition that will be generalized in § 2.7 is the idea of a *linear* or *vector* space [46].

Definition 2.6.5 (Linear space)**.** A linear or vector space $\mathcal{E}$ is a set over the reals equipped with (and closed under) vector addition and scalar multiplication by real numbers. A prime example is the space of column vectors of length d, denoted as $\mathbb{R}^d$. However, contrary to what the name suggests, a vector space can also be a set of matrices, such as the set of $n \times p$ real matrices (and its linear subspaces), labeled as $\mathbb{R}^{n \times p}$. This last example will be of special interest for us.

The dimension d of a linear space $\mathcal{E}$ is determined by the number of basis elements $\{e_i\}_{i \in [d]}$, such that every element $x \in \mathcal{E}$ can be written as linear combinations of these

$$x = \sum_i a_i e_i . \tag{2.80}$$

To construct a basis for the linear space $\mathbb{R}^{n\times p}$, we look at the vectorization of an element $x \in \mathbb{R}^{n\times p}$

$$x \to \text{vec}(x)\,. \tag{2.81}$$

Then, analogous to $\mathbb{R}^d$, the basis vectors are obtained from the so-called *unit matrices* $E_{i,j}$

$$\{E_{ij}\}_{n,p} = \begin{cases} 1 & \text{at } (n,p) = (i,j) \\ 0 & \text{elsewhere} \end{cases}\,, \tag{2.82}$$

such that an element $x \in \mathbb{R}^{n\times p}$ can be written in terms of these as

$$\text{vec}(x) = \sum_{i,j} a_{i,j}\text{vec}\left(E_{i,j}\right)\,. \tag{2.83}$$

From these relations, we can infer the linear space $\mathbb{R}^{n\times p}$ has dimension np.

In particular, if we endow a vector space $\mathcal{E}$ with an inner product $\langle .,. \rangle$, we obtain a *Euclidean space*.

Definition 2.6.6 (Euclidean space)**.** The Euclidean space is a finite dimensional vector space endowed with an *inner product* $\langle .,. \rangle : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$. In other words, it is a finite-dimensional real Hilbert space. The standard inner product on linear spaces of matrices such as $\mathbb{R}^{n\times p}$ is the *Euclidean inner product* [46]

$$\langle x_1, x_2 \rangle_e := \text{vec}(x_1)^T\text{vec}(x_2) = \text{tr}\left(x_1^T x_2\right)\,, \tag{2.84}$$

where $x_1, x_2 \in \mathbb{R}^{n\times p}$.

This inner product induces a norm on $\mathcal{E}$ known as the *Frobenius norm.*

Definition 2.6.7 (Frobenius Norm)**.** The Euclidean inner product $\langle .,. \rangle_e$ shown in Eq. (2.84) induces a norm on $\mathcal{E}$ known as the Frobenius norm $\|.\|_F$ and is defined as

$$\|x\|_F^2 = \text{tr}\left(x^T x\right) = \sum_{i,j}(x_{i,j})^2\,. \tag{2.85}$$

Hereafter, we will omit the subscript F and assume the norm to be the Frobenius norm unless stated otherwise.

2.7. A hitchhiker's guide to manifolds and tangent spaces

Optimization methods are techniques used to find the best solution to a problem from a set of feasible solutions by minimizing or maximizing an objective function. These methods play a crucial role in various fields, including engineering, machine learning, and quantum science. Across the different branches of the latter, some prime examples that have benefited from optimization techniques (often involving tensor networks methods) include: ground state finding [47], describing quantum phase transitions [48], quantum state [49] and quantum process tomography [50].

However, more often than not, we cannot naively apply optimization techniques to find the best solution. Instead, we must take into account a set of constraints that reduce the feasible solution space, and force us to modify the methods defined in the Euclidean space.

An approach to constrained optimization is the use of *Riemannian optimization* methods, being particularly suitable for quantum mechanics as many constraints appearing there can be cast into Riemannian manifolds [51]. This is exactly the approach we intend to follow in this work by casting our problem into the Stiefel manifold (to be defined in Def. (2.7.7)). This section does not attempt to be a rigorous review of differential geometry, but rather to provide the basic tools needed to navigate through the main optimization problem we will introduce in this work, focusing on the **intuition**, i.e. we intend to show the *why* and *how* we formulate and tackle the problem. Let us begin!

Roughly, our goal can be understood as trying to solve a problem of the form

$$\min_{x \in \mathcal{M}} f(x) \tag{2.86}$$

where $\mathcal{M}$ is a manifold and the real-valued map $f : \mathcal{M} \to \mathbb{R}$ is known as the *cost function*. Therefore, let us start by answering *"What is a Manifold?"*.

Definition 2.7.1 (Manifold)**.** A d-dimensional real manifold or d-manifold $\mathcal{M}$ is [8]

- a set covered by suitable coordinate "patches" called *charts* (see Def. (2.7.3)) that allow us to identify subsets of $\mathcal{M}$ with open subsets of $\mathbb{R}^d$, the d-dimensional Euclidean space.
- We then say that each point of a manifold has a neighborhood that is *homeomorphic* to $\mathbb{R}^d$ (see Def. (2.7.2)).
- **Intuitively**, we can think of a manifold as a space such that at every point it locally looks like a flat Euclidean space.
- Examples of manifolds of interest include the manifold of unitary matrices in $\mathbb{R}^{n\times n}$ and the manifold of isometries in $\mathbb{R}^{n\times p}$. The latter is also known as the *Stiefel manifold*.

Definition 2.7.2 (Homeomorphism)**.** A space is homeomorphic to another space if [52]

- there exists a bijective, continuous function f between them, so that its inverse f^{-1} is also continuous.
- **Intuitively**, this means that we can smoothly transform one into the other without any cutting or gluing. The archetype example is a donut and a mug, since we can smoothly transform one into the other, we say "a donut is homeomorphic to a mug" or they "look alike topologically".

Definition 2.7.3 (Chart)**.** Let the chart $\varphi \equiv (\mathcal{U}, \varphi)$ be the homeomorphism mapping elements from a subset $\mathcal{U}$ of $\mathcal{M}$ onto an open subset of $\mathbb{R}^d$, such that

$$\varphi(x) \in \mathbb{R}^d, \quad x \in \mathcal{U} \tag{2.87}$$

are the coordinates of x in the chart $(\mathcal{U}, \varphi)$ [8]. This allows to transform a function $f : \mathcal{U} \to \mathbb{R}$ into $f \circ \varphi^{-1} : \mathbb{R}^d \to \mathbb{R}$. A (non-unique) collection of charts covering $\mathcal{M}$ is called an *atlas*.

We will focus on special kinds of manifolds called *linear manifolds* [8].

Definition 2.7.4 (Linear manifold)**.** A linear manifold $\mathcal{M}$ is

- a subset of a vector space $\mathcal{E}$, as defined in Def. (2.6.5), that is also closed under addition and scalar multiplication.
- Every vector space $\mathcal{E}$ is a linear manifold by definition.

In order to arrive at a numerical algorithm, we will focus on representing our problems in terms of matrix manifolds, and in particular embedded submanifolds of $\mathbb{R}^{n\times p}$. As discussed in Def. (2.6.5), the set $\mathbb{R}^{n\times p}$ of $n \times p$ real matrices is a vector space, and therefore a linear manifold.

As we will see in Chapter 3, our cost function will take as input not one, but many isometries. Therefore, we need to introduce the concept of *product of manifolds*.

Definition 2.7.5 (Product manifold)**.** Given two manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ of dimensions d_1 and d_2, respectively, the set $\mathcal{M}_1 \times \mathcal{M}_2$ is the set of pairs (x_1, x_2) with $x_1 \in \mathcal{M}_1$ and $x_2 \in \mathcal{M}_2$. Let $(\mathcal{U}_1, \varphi_1)$ and $(\mathcal{U}_2, \varphi_2)$ be charts of M_1 and M_2, respectively. Then, a chart for the set $\mathcal{M}_1 \times \mathcal{M}_2$ is

$$\varphi_1 \times \varphi_2 : \mathcal{U}_1 \times \mathcal{U}_2 \to \mathbb{R}_1^{d_1} \times \mathbb{R}_2^{d_2} : (x_1, x_2) \mapsto (\varphi_1(x_1), \varphi_2(x_2)) \ . \tag{2.88}$$

These charts form an atlas covering the set $\mathcal{M}_1 \times \mathcal{M}_2$, giving rise to the product manifold of $\mathcal{M}_1$ and $\mathcal{M}_2$.

As aforesaid, in this work, we aim to focus on *embedded submanifolds* of the manifold $\mathbb{R}^{n\times p}$.

Definition 2.7.6 (Submanifold)**.** Roughly speaking, $\mathcal{N}$ is an embedded submanifold of $\mathcal{M}$ if [8]

- it is a subset $\mathcal{N} \subset \mathcal{M}$,
- it is itself a manifold,
- and its topology coincides with the subspace topology induced (inherited) from $\mathcal{M}$, thereby removing the freedom to choose a differentiable structure on $\mathcal{N}$..

Intuitively, we can think of an embedded submanifold as a surface smoothly embedded in a higher dimensional space. The manifold $\mathcal{M}$ is known as the *ambient space* or *embedding space* of $\mathcal{N}$. For the case when $\mathcal{M}$ is $\mathbb{R}^{n\times p}$, $\mathcal{N}$ is called a *matrix submanifold*. [8].

As it will become evident when we formulate our optimization problem, a particular matrix manifold of interest is the *Stiefel manifold*.

Definition 2.7.7 (Stiefel manifold)**.** The *real* Stiefel manifold is the set

$$\mathrm{St}(n,p) = \{X \in \mathbb{R}^{n\times p} : X^T X = \mathbb{1}_p\}, \quad p \le n, \tag{2.89}$$

i.e. the set of "tall" real isometries or orthonormal matrices. This is an embedded manifold of $\mathbb{R}^{n\times p}$, inheriting its topology.

Remark. To emphasize an element is part of the Stiefel manifold, we will label them as $X \in \mathrm{St}(n,p)$, contrary to the lowercase $x \in \mathcal{M}$ for a general manifold $\mathcal{M}$.

In order to define the concept of gradient of a function acting on a manifold, we need to generalize the idea of the Euclidean directional derivative from Eq. (2.71). To this end, we introduce the *tangent space*.

Definition 2.7.8 (Tangent space)**.** For a d-manifold $\mathcal{M}$, the tangent space $T_x\mathcal{M}$ is a d-dimensional vector space with its origin at x, i.e. the point of tangency [53]. **Intuitively**, the tangent space $T_x\mathcal{M}$ can be thought of as the plane tangent to $\mathcal{M}$ at x. Fig. (2.5a) depicts the relation between them, along with its orthogonal complement known as the *normal space*, see Eq. (2.116). It is useful to think of these spaces in the context of a sphere, where the tangent space is perpendicular to the radii and the normal space is radial.

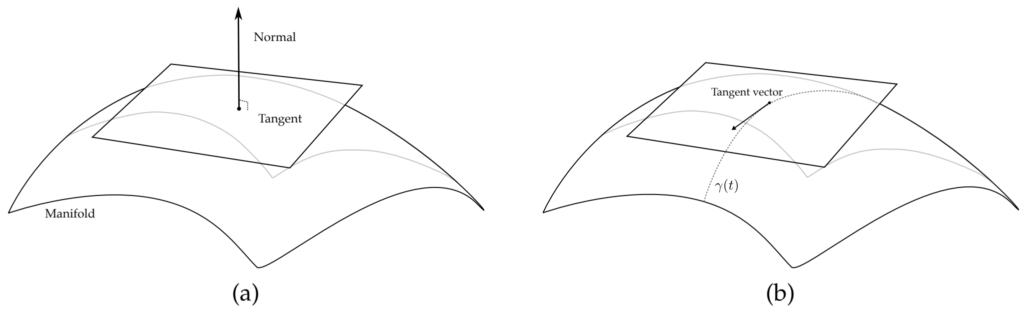


Figure 2.5.: Panel (a) shows an abstract manifold, together with the tangent and normal spaces. Panel (b) shows a curve $\gamma(t)$ on the manifold and the tangent vector to it in the tangent space.

Every element of the tangent space $T_x\mathcal{M}$ is known as a *tangent vector*.

Definition 2.7.9 (Tangent vector)**.** A rigorous definition of a tangent vector $\xi_x \in T_x\mathcal{M}$ at the point $x \in \mathcal{M}$, is given in [8] as the mapping

$$\xi_x : \mathfrak{F}(\mathcal{M}) \to \mathbb{R} : f \to \xi_x f = \dot{\gamma}(0)f := \frac{\mathrm{d}f(\gamma(t))}{\mathrm{d}t}\Big|_{t=0}, \tag{2.90}$$

for all $f \in \mathfrak{F}(\mathcal{M})$, where $\mathfrak{F}(\mathcal{M})$ denotes the set of smooth real-valued functions defined on $x \in \mathcal{M}$, and

$$\gamma : \mathbb{R} \to \mathcal{M} : t \to \gamma(t) \tag{2.91}$$

is a *curve* in $\mathcal{M}$ such that $\gamma(0) = x$. We also write

$$\xi_x f = Df(x)[\gamma'(0)]\,, \tag{2.92}$$

which expresses the action of the tangent vector on a function as a *derivation D*, satisfying *Leibniz's law* [54]. **Intuitively**, the tangent vector is a vector in the tangent space perpendicular to the curve γ with its foot at x, see Fig. (2.5b). We say "γ realizes the tangent vector ξ_x". Since $\gamma'(0)$ does not depend on the curve γ realizing ξ_x, we can write a canonical linear one-to-one correspondence [8]

$$\xi_x \to \gamma'(0)\,. \tag{2.93}$$

The set of all tangent vectors to $\mathcal{M}$ is known as the *tangent bundle*

$$T\mathcal{M} := \bigcup_{x\in\mathcal{M}} T_x\,\mathcal{M}\,. \tag{2.94}$$

Remark. If the manifold $\mathcal{M}$ is an embedded submanifold of $\mathcal{E}$, we can define the *canonical identification* of the tangent space as

$$T_x\mathcal{M} \simeq \mathcal{E}\,. \tag{2.95}$$

As we will see in § 2.9, this will be a cornerstone element in our algorithm formulation as it allows us to approximate a a manifold $\mathcal{M}$ by a linear space around x [46].

Using Eq. (2.93), we can characterize the tangent vectors Z in the tangent space of the Stiefel Manifold $T_X\,\mathrm{St}(n,p)$. To do so, we choose a curve $\gamma(t) = X(t) \in \mathrm{St}(n,p)$ with $\gamma(0) = X$. Then, if we differentiate $X(t)^T X(t) = \mathbb{1}_p$ we get

$$\dot{X}^T(t)X(t) + X^T(t)\dot{X}(t) = 0\,, \tag{2.96}$$

such that at $t = 0$ (at the tangent point) setting $Z = \dot{X}(0)$ results in

$$T_X\,\mathrm{St}(n,p) = \{Z \in \mathbb{R}^{n\times p} : Z^T X + X^T Z = 0\}\,, \tag{2.97}$$

i.e. $X^T Z$ is *skew-symmetric* [53].

Remark. Similarly to the naming of the elements $X \in \mathrm{St}(n,p)$, the tangent vectors of the tangent space $T_X\mathrm{St}(n,p)$ will be denoted by Z, contrary to η_x, ξ_x which are used in the context of a general manifold $\mathcal{M}$.

In Def. (2.6.2), we defined a vector field on a Euclidean space, analogously, here we define the *vector field* on a manifold.

Definition 2.7.10 (Vector field)**.** The vector field ξ is a smooth function from $\mathcal{M}$ to the tangent bundle $T\mathcal{M}$ that assigns a tangent vector $\xi_x \in T_x\mathcal{M}$ to each point $x \in \mathcal{M}$, i.e.

$$\xi : M \to T\mathcal{M} : x \to \xi_x\,. \tag{2.98}$$

We will use the following notation

$$\eta(x) := \eta_x, \quad \forall \eta \in \mathfrak{X}(\mathcal{M})\,. \tag{2.99}$$

where $\mathfrak{X}$ is the set of smooth vector fields on $\mathcal{M}$. Similarly to the euclidean case, if $\mathcal{M}$ is a submanifold of a vector space, the vector field can be thought of as a "collection of arrows", one on each point in $\mathcal{M}$. An alternative definition for the vector field ξ appears by looking at the linear map D_ξ that it gives rise to (a derivation)

$$D_\xi : \mathfrak{F} \to \mathfrak{F} : f \to \xi(f) := \xi f \,. \tag{2.100}$$

This is a map from smooth functions to smooth functions, with $\xi(fg) = \xi(f)g + f\xi(g)$ for any $f, g \in \mathfrak{F}$. Some sort of intuition comes when realizing that ξ maps points x to vectors ξ_x, and vectors ξ_x map functions f to real numbers. The action of ξ on a function f at a point x is [8]

$$D_\xi(f)(x) = (\xi f)(x) := \xi_x(f). \tag{2.101}$$

We can get an intuition on why a vector field ξ can be seen as a derivation by writing it in terms of its local coordinates x_i given by the chart $(\mathcal{U}, \phi)$ (see Eq. (2.87)) for $\mathcal{U} \in \mathbb{R}^n$ [55]

$$\xi = \sum_{i=1}^{n} a_i \frac{\partial}{\partial x_i} \,, \tag{2.102}$$

for some functions $a_i : \mathcal{U} \to \mathbb{R}$. This means the map in Eq. (2.98) can be written as

$$\xi(x) = \sum_{i=1}^{n} a_i(x) \frac{\partial}{\partial x_i}\Big|_x \,, \tag{2.103}$$

and the map in Eq. (2.100) is

$$\xi f = \sum_{i=1}^{n} a_i \frac{\partial f}{\partial x_i}. \tag{2.104}$$

The set of smooth vector fields on $\mathcal{M}$ is labeled as $\mathfrak{X}(\mathcal{M})$ [8].

To close this section, let us look at two linear manifolds $\mathcal{M}$ and $\mathcal{N}$, a smooth map between them $F : \mathcal{M} \to \mathcal{N}$, and a tangent vector ξ_x at $x \in \mathcal{M}$. By using Eq. (2.95), we define the *differential* or *tangent map* as [8]

$$DF(x)[\xi_x] = \lim_{t \to 0} \frac{F(x + t\xi_x) - F(x)}{t} \,, \tag{2.105}$$

where $DF(x)[\xi_x]$ is a tangent vector to $\mathcal{N}$ at $F(x)$. This shows how the notion of a tangent vector allows us to generalize the concept of a *directional derivative* on a Manifold. Given a differentiable function $F : \mathcal{M} \to \mathcal{N}$, and a vector field ξ on $\mathcal{M}$, we define the map

$$DF[\xi] : \mathcal{M} \to T\mathcal{N} : x \mapsto DF(x)[\xi_x] \,, \tag{2.106}$$

such that for a real function f on $\mathcal{M}$ we have

$$Df[\xi] = \xi f \,. \tag{2.107}$$

2.8. Riemannian manifolds: main ingredients

By endowing the tangent spaces of a manifold with a smoothly varying inner product we obtain what is known as a *Riemannian manifold*.

Definition 2.8.1 (Inner product)**.** The inner product is a bilinear symmetric positive definite *form*, i.e. a *homogeneous polynomial* or a polynomial whose terms have all the same degree [8]. If the inner product varies smoothly with $x \in \mathcal{M}$, it defines an inner product on the tangent space $T_x\mathcal{M}$, inducing a *Riemannian metric* on $\mathcal{M}$ and an associated *Riemannian distance*. The following notations are used interchangeably in literature to denote an inner product between ξ_x and η_x in $T_x\mathcal{M}$:

$$g(\xi_x, \eta_x) = g_x(\xi_x, \eta_x) = \langle \xi_x, \eta_x \rangle = \langle \xi_x, \eta_x \rangle_x . \tag{2.108}$$

From its definition, the inner product is positive for every non-zero element ξ_x. The inner product $\langle ., . \rangle_x$ induces a norm in the tangent space

$$\|\xi_x\|_x := \sqrt{\langle \xi_x, \xi_x \rangle_x} . \tag{2.109}$$

This provides us with a notion of length in $T_x\mathcal{M}$, and, as we will see in Eq. (2.114), it allows us to define a normalized direction of the steepest increase of a function $f : \mathcal{M} \to \mathbb{R}$.

A submanifold $\mathcal{M}$ of a Riemannian manifold $\overline{\mathcal{M}}$ has an *embedded Riemannian metric* g induced by the metric $\overline{g}$ of $\overline{\mathcal{M}}$,

$$g_x(\xi, \eta) = \overline{g}_x(\xi, \eta), \quad \xi, \eta \in T_x M , \tag{2.110}$$

which makes $\mathcal{M}$ a Riemannian manifold, or Riemannian submanifold of $\overline{\mathcal{M}}$. If the manifold $\mathcal{M}$ is embedded in a Euclidean space $\mathcal{E}$ with the Euclidean inner product $\langle ., . \rangle_e$ from Eq. (2.84), the inner product between two vectors $\xi_x, \eta_x \in T_x\mathcal{M}$ is [56]

$$g(\xi_x, \eta_x) = \mathrm{tr}\{\xi_x^T G_x \eta_x\} = \langle \xi_x, G_x \eta_x \rangle_e , \tag{2.111}$$

where the symmetric, positive definite matrix G_x is the matrix representation of g at x. The (i, j) element of G_x is given by $g_{i,j}\Big|_x$ with

$$g_{i,j} := g(E_i, E_j) , \tag{2.112}$$

where E_i and E_j are *coordinate vector fields* [8], i.e. the equivalent of basis vectors in the tangent space, cf. Eq. (2.82).

Definition 2.8.2 (Riemannian manifold)**.** A Riemannian manifold $\mathcal{M}$ is the tuple $(\mathcal{M}, g)$ where g is the Riemannian metric. As we saw in Eq. (2.84) the Euclidean space is a particular case of a Riemannian manifold.

The generalization of the gradient defined in Eq. (2.73) for a function f defined on a Riemannian manifold $\mathcal{M}$ is the *Riemannian gradient* [8].

Definition 2.8.3 (Riemannian gradient)**.** The Riemannian gradient of a function $f : \mathcal{M} \to \mathbb{R}$ is the unique element $\operatorname{grad} f(x) \in T_x\mathcal{M}$ satisfying

$$\langle \operatorname{grad} f(x), \xi \rangle_x = Df(x)[\xi], \quad \forall \xi \in T_x\mathcal{M}\,. \tag{2.113}$$

where $Df(x)[\xi]$ is the differential of f as defined in Eq. (2.106). It is worth highlighting that the gradient depends on the choice of metric but the differential does not [46]. Compared to the Euclidean definition in Eq. (2.73), the inner product $\langle .,. \rangle_x$ used in this definition in general need not be the Euclidean inner product, but any smoothly varying inner product as defined in Def. (2.8.1). Two properties of the Riemannian gradient of a function $f : \mathcal{M} \to \mathbb{R}$ at $x \in \mathcal{M}$ that will be leveraged in our optimization algorithm are:

- Its direction is the normalized direction of the steepest increase of f,

$$\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|_x} = \underset{\xi_x \in T_x\mathcal{M} : \|\xi_x\|=1}{\arg\max} \; Df(x)[\xi_x] \tag{2.114}$$

  where $Df(x)[\xi_x]$ is defined as in Eq. (2.105).

- Its norm is the steepest slope of f,

$$\|\operatorname{grad} f(x)\| = Df(x)\left[\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|_x}\right]. \tag{2.115}$$

Remark. We will use the notation grad instead of ∇ for the gradient within the context of manifolds. This will help to avoid confusions with the to-be-defined *affine connection* in Def. (2.8.7).

Using the metric g we can define the *normal space* of a manifold.

Definition 2.8.4 (Normal space)**.** The normal space at $x \in \mathcal{M}$ is the *orthogonal complement* of the tangent space $T_x\mathcal{M}$ embedded in $T_x\overline{\mathcal{M}}$, given by

$$N_x\mathcal{M} = \{N \in \mathbb{R}^{n\times p} : \overline{g}_x(N, \xi_x) = 0, \forall \xi_x \in T_x\mathcal{M}\}. \tag{2.116}$$

We can then write any tangent vector $\eta \in T_x\overline{\mathcal{M}}$ in terms of the tangent and normal space as

$$\eta = \pi_{T,g}(\eta) + \pi_{N,g}(\eta), \tag{2.117}$$

where $\pi_{T,g}$ and $\pi_{N,g}$ are the orthogonal projections onto the tangent and normal spaces of $\mathcal{M}$, respectively. The subscript g in the projections π indicates that these in general depend on the Riemannian metric g, while the explicit x dependence was omitted for simplicity.

Using the matrix representation described in Eq. (2.111) for g, together with the projection $\pi_{T,g}$ onto the tangent space $T_x\mathcal{M}$ of a manifold $\mathcal{M}$ embedded in the Euclidean space $\mathcal{E}$, the Riemannian gradient can be evaluated as [56]

$$\operatorname{grad} f(x) = \pi_{T,g}(G_x^{-1} \operatorname{grad} \overline{f}(x))\,, \tag{2.118}$$

where $\operatorname{grad} \overline{f}(x)$ is the *ambient gradient* of f, and here it coincides with the Euclidean gradient defined in Eq. (2.73). In the case of the Stiefel manifold $\mathrm{St}(n, p)$, a natural choice for the Riemannian metric comes from its embedding space, i.e. the Euclidean metric from Eq. (2.84). Under this choice, given a point $X \in \mathrm{St}(n, p)$, the Riemannian gradient is of the form [8]

$$\operatorname{grad} f(X) = \pi_{T,X}(\operatorname{grad} \overline{f}(X)) , \tag{2.119}$$

where $\pi_{T,X}$ is the (metric independent) tangent space projector

$$\pi_{T,X}(\xi) = (I - XX^T)\xi + X\mathrm{skew}(X^T\xi) , \tag{2.120}$$

with

$$\mathrm{skew}(A) := \frac{1}{2}(A - A^T) . \tag{2.121}$$

We will see in § 4.5 that this gradient is not unique, and as shown in Eq. (2.118), it depends on the choice of the metric.

Optimization methods in the Euclidean space like the ubiquitous *gradient descent* algorithm [57] are instances of *line search* methods based on the update formula

$$x_{k+1} = x_k + t_k\eta_k , \tag{2.122}$$

where η_k is the *search direction* and t_k is the *step size*. Unfortunately, this scheme cannot be applied naively to a function $f : \mathcal{M} \to \mathbb{R}$, since in general η_k is not a tangent vector to $\mathcal{M}$ and does not take the metric g into account; in addition, the scheme would take the x_{k+1} out of the manifold. The former issue can be solved by looking at the definition of the Riemannian gradient Def. (2.8.3), and defining the search direction as

$$\eta := -\operatorname{grad} f(x). \tag{2.123}$$

However, moving in this direction in the Euclidean sense would still take a point out of the manifold. Therefore, we need to generalize this concept for the Riemannian geometry. This can be achieved through the *exponential map* at $x \in \mathcal{M}$

$$\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M} : \eta \to \mathrm{Exp}_x\eta = \gamma(1, x, \eta) , \tag{2.124}$$

where γ is a *geodesic* [8, 51].

Definition 2.8.5 (Geodesic)**.** The concept of a straight line in the Euclidean space $\mathbb{R}^n$ is generalized to a manifold by the geodesic curve $\gamma(t)$. For every point $x \in \mathcal{M}$ with the tangent vector $\eta \in T_x\mathcal{M}$, the geodesic $\gamma(t, x, \eta)$ satisfies

- $\gamma(0) = x$,
- $\dot{\gamma}(0) = \eta$.

Nevertheless, computing the exponential map is highly computationally expensive, and in many cases, might turn out to be more difficult than the original Riemannian optimization problem. For this reason, we instead look at *retractions*.

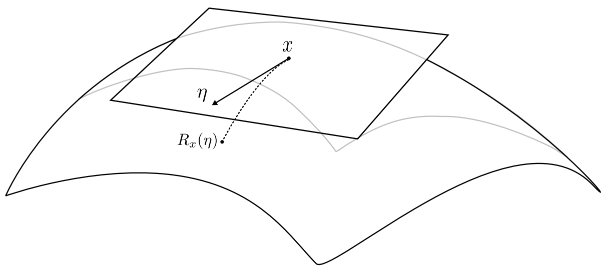


Figure 2.6.: Riemannian manifold $\mathcal{M}$, showing a tangent space at a point x, with tangent vector $\eta \in T_x\mathcal{M}$, and the retraction point $R_x(\eta)$.

Definition 2.8.6 (Retraction)**.** The idea of moving along a tangent vector $\eta \in T_x\mathcal{M}$ while staying in a manifold $\mathcal{M}$ is realized up to first-order [58] by the smooth retraction map $R_x : T_x\mathcal{M} \to \mathcal{M}$ satisfying

1. $R_x(0_x) = x$, with 0_x the zero element of $T_x\mathcal{M}$,
2. $DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$, where $\mathrm{id}_{T_x\mathcal{M}}$ is the identity mapping in the tangent space.

The second condition is known as the *local rigidity condition*, which can be understood as the preservation of the gradients at x. As a result, for every tangent vector $\eta \in T_x\mathcal{M}$, moving along a curve

$$\gamma_\eta : t \to R_x(t\eta, \quad \dot{\gamma}_\eta(0) = \eta \tag{2.125}$$

can be thought of as moving along η while staying in the manifold $\mathcal{M}$. The retraction at x along η is depicted in Fig. (2.6), where we see the procedure as:

1. Start at a point $x \in \mathcal{M}$.
2. Move along η to the point $x + \eta \in T_x\mathcal{M}$.
3. Project back to the manifold $\mathcal{M}$ using R_x.

For a real-valued function f defined on a Riemannian manifold $\mathcal{M}$, the *pullback* of f through the retraction R_x in the neighborhood of $x \in \mathcal{M}$ is the real-valued function acting on the tangent space $T_x\mathcal{M}$

$$\hat{f}_x := f \circ R_x : T_x\mathcal{M} \to \mathbb{R}\,. \tag{2.126}$$

Using the local rigidity condition from Def. (2.8.6) and the canonical identification in Eq. (2.95) yields

$$\operatorname{grad} \hat{f}_x(0_x) = \operatorname{grad} f(x)\,. \tag{2.127}$$

For the Stiefel manifold $\mathrm{St}(n, p)$, a projection-based retraction R_X based on the polar decomposition is given in [8] as

$$R_X(\eta) = (X + \eta)(I_p + \eta^T\eta)^{-1/2}\,, \tag{2.128}$$

where $(A)^{-1/2}$ can be implemented through the eigenvalue decomposition of A. As we will discuss in § 4.6, there exist alternative retractions for the Stiefel manifold, and some of them are preffered due to their numerical stability. We now introduce the last main ingredient of this section, the *affine connections* [8].

Definition 2.8.7 (Affine connection)**.** An affine connection ∇ on a manifold $\mathcal{M}$ is a mapping

$$\nabla : \mathfrak{X} \times \mathfrak{X} \to \mathfrak{X} : (\eta, \xi) \to \nabla_\eta \xi \,, \tag{2.129}$$

We say the vector field $\nabla_\eta \xi$ is the *covariant derivative* of ξ with respect to η for the affine connection ∇. For the case when $\mathcal{M} = \mathbb{R}^n$, ∇ becomes the *canonical* (Euclidean) connection giving rise to the classical directional derivative

$$(\nabla_\eta \xi)_x = \lim_{t \to 0} \frac{\xi_{x+t\eta_x} - \xi_x}{t} \,, \tag{2.130}$$

for $\xi_x, \eta_x \in T_x\mathcal{M}$ and $\xi, \eta \in \mathfrak{X}(\mathcal{M})$, cf. Eq. (2.71). All affine connections ∇ satisfy:

- linearity in η,
- linearity in ξ,
- and Leibniz's law (see Section 5.2 in [8] for details).

Intuitively, ∇ on a manifold allows us to differentiate a vector field ξ along another vector field η, i.e. it is the directional derivative of a vector field. This is in general not possible since the tangent vectors in Eq. (2.130) belong to different spaces. The operator ∇ is exactly what provides this *connection* between different tangent spaces of a manifold.

In a Riemannian manifold the main connection of interest is known as the *Levi-Civita* or *Riemannian connection*, due to its symmetry and invariance of the metric properties (see Section 5.3.2 of [8]).

Given the decomposition of a vector $\xi_x \in Tx\overline{\mathcal{M}}$ from Eq. (2.117), where $\overline{\mathcal{M}}$ is the embedded ambient Riemannian manifold of the Riemannian submanifold $\mathcal{M}$, and let $\overline{\nabla}$ and ∇ be the Riemannian connections of $\overline{\mathcal{M}}$ and $\mathcal{M}$, by proposition 5.3.2 of [8] we get the relation

$$\nabla_{\eta_x} \xi = \pi_{T,g} \left(\overline{\nabla}_{\eta_x} \xi \right) \,, \tag{2.131}$$

for all $\eta_x \in T_x\mathcal{M}$ and $\xi \in \mathfrak{X}(\mathcal{M})$. The ambient Riemannian connection $\overline{\nabla}$ is given in Theorem 3.1 of [56] as

$$\overline{\nabla}_{\eta_x} \xi := D\xi(x)[\eta_x] + G_x^{-1} \mathcal{K}(\eta_x, \xi) \tag{2.132}$$

in terms of the *Christoffel metric term* $\mathcal{K} \in \overline{\mathcal{M}}$

$$\mathcal{K}(\eta, \xi) := \frac{1}{2} \left[(DG_x[\eta])\xi + (DG_x[\xi])\eta - \mathcal{X}(\eta, \xi) \right] \,. \tag{2.133}$$

The *cross term* $\mathcal{X}(\eta, \xi) \in \overline{\mathcal{M}}$ in this expression is a bilinear form acting on a pair of vector fields (η, ξ) such that for any vector field ξ_0

$$\langle \mathcal{X}(\eta, \xi), \xi_0 \rangle_{\hat{x}} = \langle \eta, (DG_x[\xi_0])\xi \rangle_{\hat{x}} \,, \tag{2.134}$$

with $\langle .,. \rangle_{\bar{x}}$ the metric of $\overline{\mathcal{M}}$. When $\overline{\mathcal{M}}$ is the Euclidean space $\mathbb{R}^{n\times p}$ and $\mathcal{M}$ inherits the *constant* Euclidean metric, Eq. (2.132) simplifies to

$$\overline{\nabla}_{\eta_x}\xi = (D\xi(x)[\eta_x]) \ . \tag{2.135}$$

Thus, using the projection in Eq. (2.120), the Riemannian connection for $X \in \mathrm{St}(n,p)$ with the embedded Euclidean metric is given for $\eta_x \in T_X \mathrm{St}(n,p)$ and $\xi \in \mathfrak{X}(\mathrm{St}(n,p))$ as

$$\nabla_{\eta_X}\xi = \pi_{T,X}\left(D\xi(X)[\eta_X]\right) , \tag{2.136}$$

i.e. an orthogonal projection onto the tangent space applied on the classical directional derivative from Eq. (2.71) [8].

Recall from Def. (2.6.3) the definition of the Hessian matrix in $\mathbb{R}^n$. For an abstract Euclidean space $\mathcal{E}$ it is useful to think of the Hessian at x as a linear map: $H : \mathcal{E} \to \mathcal{E}$, given in the orthonormal basis $(e_i)_{i=1,\dots,n}$ of $\mathcal{E}$ by

$$\mathrm{Hess}\, f(x)[z] := \sum_{ij} \partial^2_{ij}\hat{f}(x^1,\dots,x^n) z^j e_i \, , \tag{2.137}$$

where $z = \sum_j z^j e_j$ and $\hat{f}(x^1,\dots,x^n) = f(x^1 e_1 + \dots x^n e_n)$. Generalizing this concept to a Riemannian manifold, we get the *Riemannian Hessian* [8].

Definition 2.8.8 (Riemannian Hessian)**.** The Riemannian Hessian of a function f defined on a Riemannian manifold $\mathcal{M}$ at a point $x \in \mathcal{M}$ is the linear mapping $\mathrm{Hess}\, f(x) : T_x\mathcal{M} \to T_x\mathcal{M}$ given by

$$\mathrm{Hess}\, f(x)[\xi_x] = \nabla_{\xi_x} \mathrm{grad}\, f(x) \, , \tag{2.138}$$

for all tangent vectors $\xi_x \in T_x\mathcal{M}$, with ∇ the Riemannian connection from Eq. (2.131) and grad the Riemannian gradient from Eq. (2.118). The Riemannian Hessian is symmetric under the Riemannian metric $\langle .,. \rangle_x$

$$\langle \mathrm{Hess}\, f(x)[\xi], \eta \rangle_x = \langle \xi, \mathrm{Hess}\, f(x)[\eta] \rangle_x \, , \tag{2.139}$$

for all vector fields $\xi, \eta \in \mathfrak{X}(\mathcal{M})$.

Remark. Similar as for the gradient, we will use the notation Hess instead of H for the Hessian operator within the context of manifolds to make a distinction from the $\mathbb{R}^n$ case.

2.9. Trust-region optimization on Riemannian manifolds

In the previous sections, we developed the ingredients from Riemannian optimization that will come in handy for our main optimization algorithm applied to the Stiefel manifold $\mathrm{St}(n,p)$. Here, we will discuss the precise algorithm employed in this work, together with a rationale for its use.

In works like the one from Brieger et al. [59], the Newton algorithm [8] was used to optimize a cost function defined on $\mathrm{St}(n,p)$. Here, instead, we employ the *trust-region* algorithm, as done in [60] for the unitary manifold, as it combines the local superlinear rate of convergence (i.e. faster than linear) of the Newton method with global convergence properties. Further, the trust-region algorithm provides a decrease in the computational cost, since it avoids the expensive inverse Hessian computation present in Newton-like methods. It is beneficial to think of this algorithm as selecting the next point x_{k+1} as the *critical point*, i.e. points x_* where $\operatorname{grad} f(x_*) = 0$, of the *quadratic model* corresponding to the second-order Taylor expansion of $f(x_k)$. This is an alternative viewpoint of the *line search* iteration described in Eq. (2.122).

Definition 2.9.1 (Models)**.** A model m_x of a real-valued function f on a manifold $\mathcal{M}$ around $x \in \mathcal{M}$ is a function defined on the neighborhood of x such that

1. m_x is a "good enough" approximation of f around x, where the quality of the model is defined in terms of the distance between $f(y)$ and $m_x(y)$, as shown in Eq. (2.140).
2. m_x has a "simple" form that makes it suitable for optimization algorithms. Often, like in this work, this means we choose m_x to be a quadratic model, as several algorithms are available to minimize a quadratic function under trust-region constraints [8].

For $q > 0$, a model m_x is of order q if there exists a neighborhood $\mathcal{U}$ around x and a constant $c > 0$ satisfying

$$|f(y) - m_x(y)| \leq c\,(\operatorname{dist}(x,y))^{q+1}, \quad \forall y \in \mathcal{U}, \tag{2.140}$$

where $\operatorname{dist}(x,y)$ is the distance induced by the Riemannian metric. For a retraction R_x on $\mathcal{M}$, we get

$$|f(y) - m_x(y)| \leq c\|R_x^{-1}(y)\|^{q+1}, \quad \forall y \in \mathcal{U}. \tag{2.141}$$

As we did in Eq. (2.126), we can define the pullback of f through the retraction R_x as the map from $T_x\mathcal{M}$ to $\mathbb{R}$

$$\hat{f}_x = f \circ R_x .$$

Then, we define $\hat{m}_x$ as the order-q Taylor expansion of $\hat{f}_x$ around the origin element 0_x in the tangent space $T_x\mathcal{M}$. Pushing $\hat{m}_x$ forward through R_x we get the order-q model of f around x

$$m_x = \hat{m}_x \circ R_x^{-1} . \tag{2.142}$$

Using what we learned in § 2.8, we can define the first-order model using

$$\hat{m}_x(\xi) = f(x) + \langle \operatorname{grad} f(x), \xi \rangle + \frac{1}{2}\langle \operatorname{Hess} f(x)[\xi], \xi \rangle, \quad \xi \in T_x\mathcal{M}, \tag{2.143}$$

where grad and Hess are the Riemannian gradient and Hessian, respectively. Compare this equation with the Euclidean equivalent in Eq. (2.79). Due to the local rigidity condition of R_x we get $\operatorname{grad} f(x) = \operatorname{grad} \hat{f}_x(0_x)$. Since, as we saw in Def. (2.8.6), the retraction used in this work is only of first order, we have $\operatorname{Hess} f(x) \neq \operatorname{Hess} \hat{f}_x(0_x)$. Nevertheless, any critical point x_* of f satisfies

$$\operatorname{Hess} f(x_*) = \operatorname{Hess} \hat{f}_{x_*}(0_{x_*}) . \tag{2.144}$$

With this in mind, we can now define the *trust-region method* on a Riemannian manifold [8].

Definition 2.9.2 (Trust-region algorithm)**.** Given a Riemannian manifold $(\mathcal{M}, g)$ with retraction R, a real-valued function f on $\mathcal{M}$, and a current optimization iterate $x_k \in \mathcal{M}$, we map the optimization problem onto the tangent space with the pullback

$$\hat{f}_{x_k} : T_{x_k}\mathcal{M} \to \mathbb{R} : \xi \to f(R_{x_k}(\xi)). \tag{2.145}$$

The trust-region *subproblem* solved by the *inner iteration* is

$$\min_{\eta \in T_{x_k}\mathcal{M}} \hat{m}_{x_k}(\eta) = f(x_k) + \langle \operatorname{grad} f(x_k), \eta \rangle_{x_k} + \frac{1}{2}\langle H_k[\eta], \eta \rangle_{x_k}, \qquad \langle \eta, \eta \rangle_{x_k} \leq \Delta^2, \tag{2.146}$$

where Δ is known as the *trust-region radius*, H_k can in general be any symmetric linear operator on $T_{x_k}\mathcal{M}$, and $\langle ., . \rangle_{x_k}$ is the inner product of the tangent space at x_k from Eq. (2.108). However, in order to achieve superlinear convergence we want $H_k \approx \operatorname{Hess} f(x_k)$. In this work, we will choose $H_k = \operatorname{Hess} f(x_k)$. The inner iteration must be an algorithm defined on $T_x\mathcal{M}$, which in our case is a linear subspace of $\mathbb{R}^{n \times p}$. Once we obtain an approximate solution η_k, we propose a candidate for the next iterate $R_{x_k}(\eta_k)$. To assess the quality of the model m_{x_k}, and hence determine whether to accept the candidate and adapt the radius, we define the quotient

$$\rho_k = \frac{f(x_k) - f(R_{x_k}(\eta_k))}{\hat{m}_{x_k}(0_{x_k}) - \hat{m}_{x_k}(\eta_k)} = \frac{\hat{f}_{x_k}(0_{x_k}) - \hat{f}_{x_k}(\eta_k)}{\hat{m}_{x_k}(0_{x_k}) - \hat{m}_{x_k}(\eta_k)}\,. \tag{2.147}$$

Based on the value of ρ_k we identify the following cases:

- If $\rho_k \ll 1$, the model is very inaccurate: we **reject** $R_{x_k}(\eta_k)$ and **reduce** Δ.
- If ρ_k is small but not as drastically: we **accept** $R_{x_k}(\eta_k)$ but **reduce** Δ.
- If $\rho_k \approx 1$, the model and function have a good agreement: we **accept** $R_{x_k}(\eta_k)$ and **increase** Δ.
- If $\rho_k \gg 1$, the model is inaccurate, but (luckily) the cost function is decreasing: in this case, we can try our luck and **increase** Δ to hope for further decrease in f.

Finally, in this work, we will choose to solve the trust-region subproblem approximately by means of a computationally light algorithm known as the *conjugate-gradient* (CG) method, see 7.3.2 of [8] for details. This algorithm is based on improving upon the *Cauchy Point*.

Definition 2.9.3 (Cauchy Point)**.** The Cauchy point is the solution η_k^C of the one-dimensional problem

$$\eta_k^C = \arg\min_{\eta}\{\hat{m}_{x_k}(\eta) : \eta = -\tau \operatorname{grad} f(x_k),\ \tau > 0,\ \|\eta\| \leq \Delta_k\}\,, \tag{2.148}$$

assuming $\operatorname{grad} f(x_k) \neq 0$. The *Cauchy decrease* is

$$\hat{m}_{x_k}(0) - \hat{m}_{x_k}(\eta_k^C)\,. \tag{2.149}$$

Using the conjugate-gradient method, we can achieve a decrease in the Riemannian subproblem of at least the Cauchy decrease. Further, if $H_k > 0$, given enough iterations, we can *estimate* the minimizer

$$\eta_k^N = (H_k)^{-1} \operatorname{grad} f(x_k) \tag{2.150}$$

of the subproblem, provided this lies within Δ. A main advantage of the CG algorithm is its *inverse-free* nature, as it does not require the computation of H_k^{-1}, but only of the product of H_k with a tangent vector.

3. Problem Formulation

In the previous chapter, we learned about open quantum systems, quantum channels, their representation in terms of tensor networks, and optimization techniques on matrix manifolds, focusing on the Stiefel manifold. Here, we will use these concepts to formulate precisely the problem addressed in this work.

In § 2.5, we saw how to describe a mixed state as a *Locally Purified Density Operator* (LPDO) tensor network ansatz. In particular, we saw how to simulate the dissipative dynamics of nearest-neighbor Lindblad operators, for which Alg. (3) yields the second-order Trotter approximation shown in Eq. (2.64)

$$\mathrm{e}^{\tau\hat{\mathcal{L}}} = \mathrm{e}^{\tau\hat{\mathcal{L}}_o/2}\mathrm{e}^{\tau\hat{\mathcal{L}}_e}\mathrm{e}^{\tau\hat{\mathcal{L}}_o/2} + O(\tau^3)\,.$$

However, finding higher-order Trotter approximations for Markovian dynamics remains an open problem [6]. This challenge arises from the semigroup property of Lindblad superoperators since, contrary to the unitary evolution, the negative coefficients in higher-order Trotter splittings [7, 61, 62] would result in non-physical maps. In this work, we aim to address this limitation by leveraging the gauge freedom of quantum channels described in § 2.5, along with the Riemannian optimization methods on the Stiefel manifold we studied in § 2.8. Our objective is to achieve a better approximation error for the dissipative dynamics of nearest-neighbor Lindbladians while maintaining the second-order Trotter structure of the layers, shown in Eq. (2.64).

3.1. From global to local superoperators

With this goal in mind, let us look in detail at the form of the odd $\mathrm{e}^{\tau\hat{\mathcal{L}}_o}$ and even $\mathrm{e}^{\tau\hat{\mathcal{L}}_e}$ layers in the above splitting for a quantum system with N sites. Each superoperator $\hat{\mathcal{L}}^{[l,l+1]} \in \mathbb{C}^{d^{2N}\times d^{2N}}$ in Eq. (2.63) is of the form described in Eq. (2.27),

$$\begin{aligned}\hat{\mathcal{L}}^{[l,l+1]} &= -i(H^{[l,l+1]}\otimes\mathbb{1}^{\otimes N} - \mathbb{1}^{\otimes N}\otimes H^{[l,l+1]T}) \\ &\quad + \sum_k \left[L_k^{[l,l+1]}\otimes L_k^{[l,l+1]*} - \frac{1}{2}\left(L_k^{[l,l+1]\dagger}L_k^{[l,l+1]}\otimes\mathbb{1}^{\otimes N} + \mathbb{1}^{\otimes N}\otimes L_k^{[l,l+1]T}L_k^{[l,l+1]*}\right)\right]\,, \end{aligned} \tag{3.1}$$

where $\mathbb{1}^{\otimes N}$ is a $d^N \times d^N$ identity matrix. While the final form of the tensor would be the same with the coherent terms, in this work, we focus on the dissipative dynamics. Therefore, the Hamiltonian terms will be omitted from here onwards in our formulation. The jump operators $L_k^{[l,l+1]}$ in Eq. (3.1) are a special case of the ones we saw in Eq. (2.27), as they are defined on a Hilbert space of dimension d^N, while acting non-trivially only on two sites at a time: l and $l+1$. Namely, they are of the form

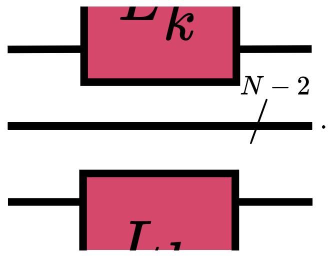
$$L_k^{[l,l+1]} = \mathbb{1}^{\otimes l-1} \otimes L_k \otimes \mathbb{1}^{\otimes N-1-l}, \qquad L_k \in \mathbb{C}^{d^2 \times d^2}, \tag{3.2}$$

for $l \in [1, N-1]$, while for $l = N$, we use *periodic boundary conditions* (PBC) to define

$$L_k^{[N,1]} = L_k^1 \otimes \mathbb{1}^{\otimes N-2} \otimes L_k^N, \quad L_k = L_k^N \otimes L_k^1 \in \mathbb{C}^{d^2 \times d^2}. \tag{3.3}$$

The diagram representation of the boundary operator showing the PBC is

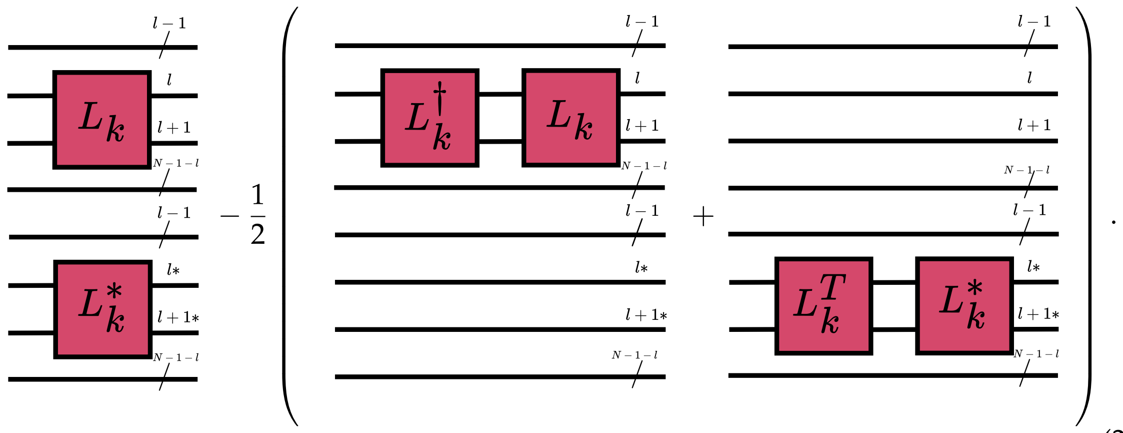
$$L_k^{[N,1]} = \tag{3.4}$$



Using Eqs. (3.1) and (3.2) we can rewrite each $\hat{\mathcal{L}}^{[l,l+1]}$ as

$$\hat{\mathcal{L}}^{[l,l+1]} = \sum_k \Big[\left(\mathbb{1}^{\otimes l-1} \otimes L_k \otimes \mathbb{1}^{\otimes N-1-l}\right) \otimes \left(\mathbb{1}^{\otimes l-1} \otimes L_k^* \otimes \mathbb{1}^{\otimes N-1-l}\right)$$
$$- \left(\mathbb{1}^{\otimes l-1} \otimes L_k^\dagger L_k \otimes \mathbb{1}^{\otimes N-1-l}\right) \otimes \frac{\mathbb{1}^{\otimes N}}{2} - \frac{\mathbb{1}^{\otimes N}}{2} \otimes \left(\mathbb{1}^{\otimes l-1} \otimes L_k^T L_k^* \otimes \mathbb{1}^{\otimes N-1-l}\right) \Big], \tag{3.5}$$

and the tensor representation for the k-th operator is



(3.6)

In this diagram, we recognize the non-identity terms as the vectorization of the k-th dissipative term $\hat{\mathcal{D}}_k^{[l,l+1]}$, as introduced in Eq. (2.26). Contrary to the single-site scenario, the components of this operator in a multi-site system are spread apart (see global vectorization in Eq. (3.15)). This difference invites us to define its components as

$$\mathcal{D}_1^{k[l,l+1]} = L_k + \frac{1}{\sqrt{2}}(L_k^\dagger L_k + \mathbb{1}), \tag{3.7}$$

$$\mathcal{D}_2^{k[l,l+1]*} = L_k^* + \frac{1}{\sqrt{2}}(\mathbb{1} + L_k^T L_k^*). \tag{3.8}$$

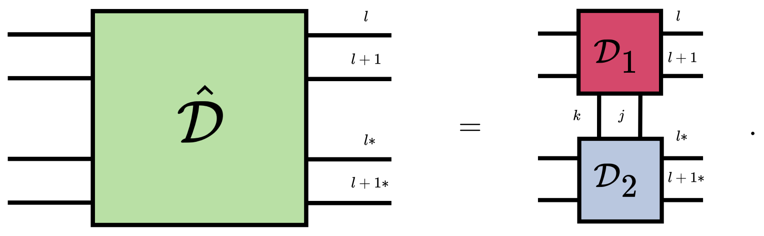
Further, we can stack each of the terms in $\mathcal{D}_1^{k[l,l+1]}$ and $\mathcal{D}_2^{k[l,l+1]*}$ into two rank-three tensors $\mathcal{D}_{1,j}^{k[l,l+1]}$ and $\mathcal{D}_{2,j}^{k[l,l+1]*}$ to arrive at the expression

$$\hat{\mathcal{D}}_k^{[l,l+1]} = \sum_j \mathcal{D}_{1,j}^{k[l,l+1]} \otimes \mathcal{D}_{2,j}^{k[l,l+1]*}. \tag{3.9}$$

By summing over all k we obtain

$$\hat{\mathcal{D}}^{[l,l+1]} = \sum_k \hat{\mathcal{D}}_k^{[l,l+1]} = \sum_{j,k} \mathcal{D}_{1,j}^{k[l,l+1]} \otimes \mathcal{D}_{2,j}^{k[l,l+1]*}, \tag{3.10}$$

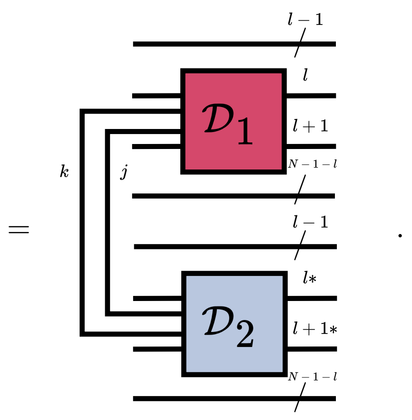
with the tensor diagram representation



(3.11)

This relation helps us rewrite $\hat{\mathcal{L}}^{[l,l+1]}$ as a tensor product of the form

$$\hat{\mathcal{L}}^{[l,l+1]} = \sum_{j,k} \mathbb{1}^{\otimes l-1} \otimes \mathcal{D}_{1,j}^{k[l,l+1]} \otimes \mathbb{1}^{\otimes N-1-l} \otimes \mathbb{1}^{\otimes l-1} \otimes \mathcal{D}_{2,j}^{k[l,l+1]*} \otimes \mathbb{1}^{\otimes N-1-l}$$

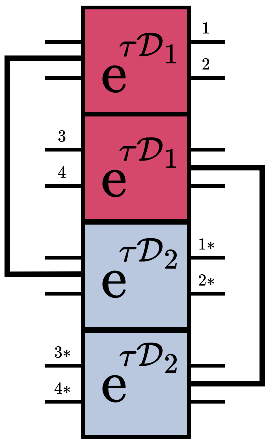
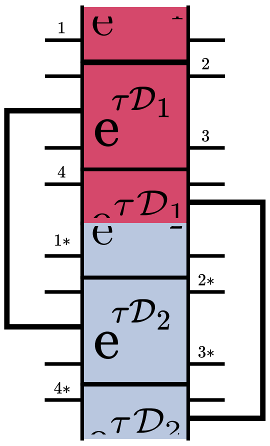


(3.12)

We can then take advantage of the following relation for the exponential of a non-interacting tensor product sum

$$\mathrm{e}^{(A\otimes\mathbb{1})+(\mathbb{1}\otimes B)} = \mathrm{e}^{A\otimes\mathbb{1}} \cdot \mathrm{e}^{\mathbb{1}\otimes B} = (\mathrm{e}^A \otimes \mathbb{1}) \cdot (\mathbb{1} \otimes \mathrm{e}^B) = \mathrm{e}^A \otimes \mathrm{e}^B \tag{3.13}$$

to derive an expression for the odd $\mathrm{e}^{\tau\hat{\mathcal{L}}_o}$ and even $\mathrm{e}^{\tau\hat{\mathcal{L}}_e}$ layers. Namely, using Eqs. (3.10), (3.12), and (3.13) we can depict the tensor diagrams for $N = 4$ as

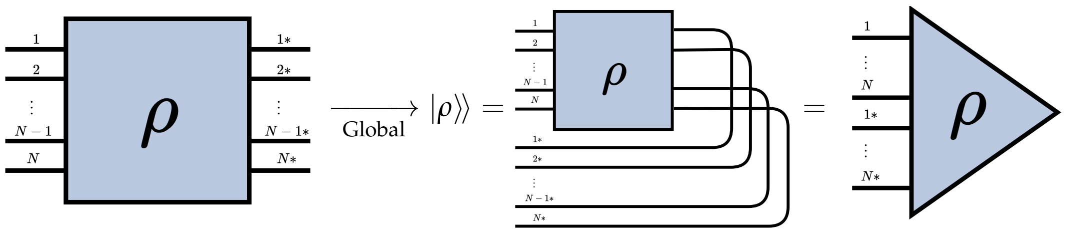
$e^{\tau\hat{\mathcal{L}}_o} =$  and $e^{\tau\hat{\mathcal{L}}_e} =$  , (3.14)

where the single thicker leg going from $e^{\tau\mathcal{D}_1}$ to $e^{\tau\mathcal{D}_2}$ is the channel index q shown in Eq. (2.69). We have used Eq. (3.4) for the PBC in $e^{\tau\hat{\mathcal{L}}_e}$, i.e. the term acting on sites 4 and 1 is shown as having half the operator on each site. A reshuffle of the legs, which we call *global-to-local* transformation, allows us to bring $\mathcal{D}_1^{[l,l+1]}$ and $\mathcal{D}_2^{[l,l+1]*}$ next to each other (see Def. (3.1.1)). This yields the expression shown in Eq. (2.67)

$$e^{\tau\hat{\mathcal{L}}_{o,L}} = \bigotimes_l e^{\tau\hat{\mathcal{D}}^{[2l-1,2l]}} \quad \text{and} \quad e^{\tau\hat{\mathcal{L}}_{e,L}} = \bigotimes_l e^{\tau\hat{\mathcal{D}}^{[2l,2l+1]}},$$

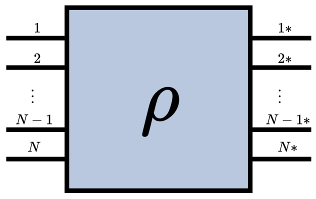
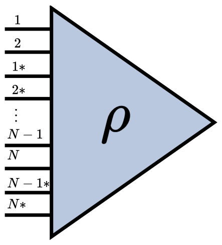
where we have added the L subscript to emphasize that these superoperators come from a localized version of the original $\mathcal{L}_o$ and $\mathcal{L}_e$.

Let us look in detail at the global-to-local transformation. The standard ordering of the legs like the one in $\hat{\mathcal{L}}^{[l,l+1]}$ is based on the *global* vectorization of the N-sites density matrix

$\rho =$  .

(3.15)

In other words, we have sites $[1,\ldots,N]$ followed by their conjugated counterpart $[1*,\ldots,N*]$, as shown in Eq. (3.6). On the other hand, Eq. (2.67) is based on a *local* vectorization where every two sites $[l, l+1]$ are followed by their conjugate versions $[l*, l+1*]$. The local vectorization of the density matrix looks like

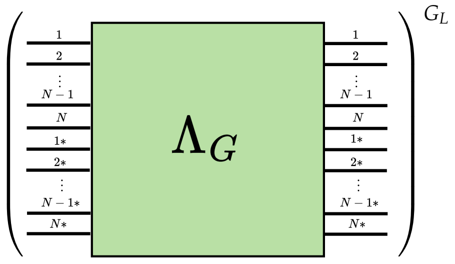
$\rho =$ 
$\xrightarrow[\text{Local}]{} |\rho\rangle\rangle =$ 
. (3.16)

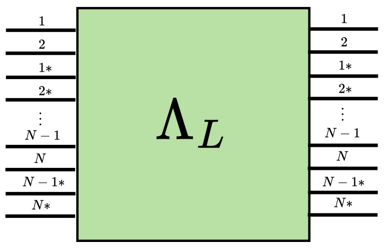
We can formally define this transformation as follows.

Definition 3.1.1 (Global-to-local)**.** Given a multi-partite superoperator $\Lambda_G \in \mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2 \ldots \otimes \mathscr{H}_N \otimes \mathscr{H}_{1*} \otimes \mathscr{H}_{2*} \otimes \ldots \otimes \mathscr{H}_{N*})$, the global-to-local transformation G_L is defined as the map

$$\begin{aligned} G_L : &\mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2 \ldots \otimes \mathscr{H}_N \otimes \mathscr{H}_{1*} \otimes \mathscr{H}_{2*} \otimes \ldots \otimes \mathscr{H}_{N*}) \to \\ &\quad \mathscr{L}(\mathscr{H}_1 \otimes \mathscr{H}_2 \otimes \mathscr{H}_{1*} \otimes \mathscr{H}_{2*} \ldots \otimes \mathscr{H}_{N-1} \otimes \mathscr{H}_N \otimes \mathscr{H}_{N-1*} \otimes \mathscr{H}_{N*}) \\ &: \Lambda_G \to \Lambda_L = (\Lambda_G)^{G_L} . \end{aligned} \tag{3.17}$$

Applying this transformation to our superoperator Λ is depicted as



$=$ 

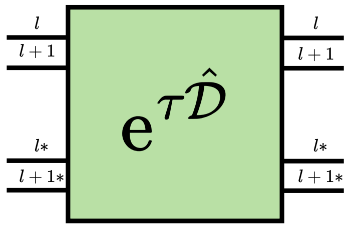
. (3.18)

The inverse map follows trivially, and we call it the *local-to-global* transformation

$$L_G : \Lambda_L \to \Lambda_G = (\Lambda_L)^{L_G} . \tag{3.19}$$

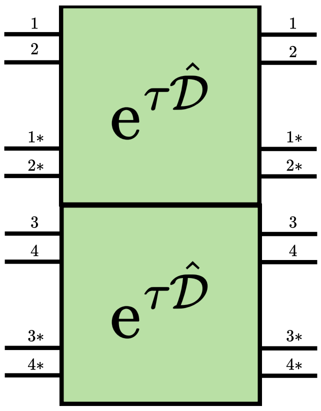
Remark. Notice that the localized version of the superoperator arising from the global-to-local operation is defined in this way because the operators L_k act on two sites. A similar construction would be obtained if instead each operator acted on m sites. In this case, we would have sites $[l, \ldots, l+m-1]$ followed by their conjugated version $[l*, \ldots, l+m-1*]$.

The local vectorization is the natural ordering that arises from the tensor product in Eq. (2.67) where each local superoperator looks like

$e^{\tau\hat{\mathcal{D}}^{[l,l+1]}} =$ 

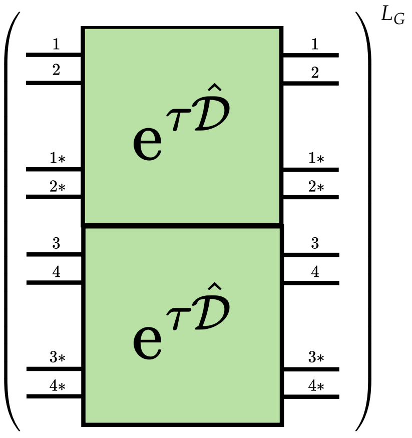
. (3.20)

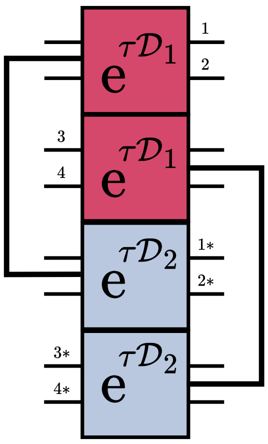
Therefore, if we tensor these operators across all sites we obtain the alternating pattern of the legs we see in Eq. (3.16). For $N = 4$ the localized odd layer diagram is

$e^{\tau\hat{\mathcal{L}}_{o,L}} =$ 

. (3.21)

The even layer follows a similar pattern by taking care of the PBC. Using the local-to-global map L_G we obtain the relation

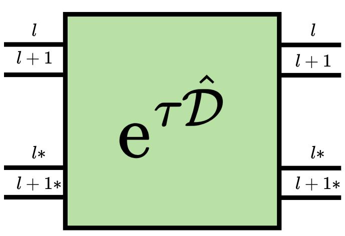


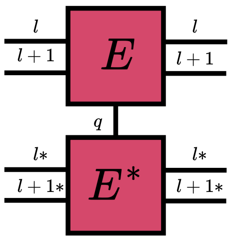
$=$ 

. (3.22)

3.2. From superoperators to isometries

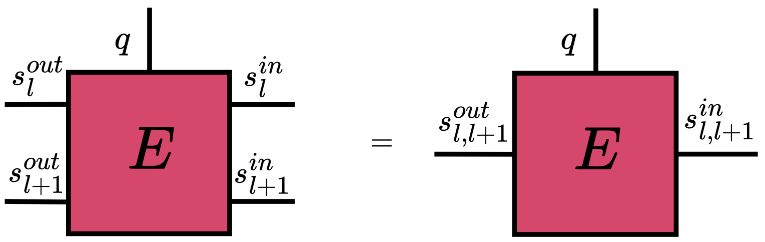
Now that we have transformed our expressions to focus on the local superoperators $e^{\tau\hat{\mathcal{D}}^{[l,l+1]}}$ we can reformulate these to fit into a Riemannian optimization scheme. As anticipated in § 2.8 we will cast these operators into the Stiefel manifold $\mathrm{St}(n, p)$. The first thing to notice is the relation between these superoperators and their Kraus representation. To do so, we employ Eq. (2.52),



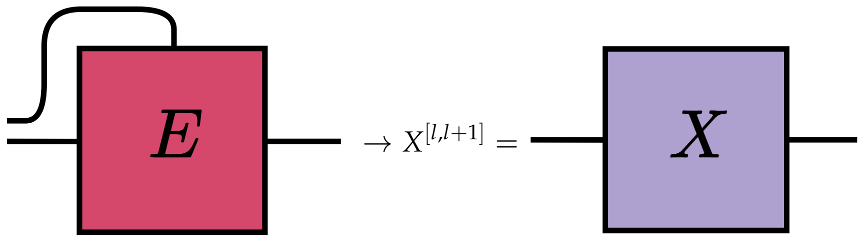
$=$ 

, (3.23)

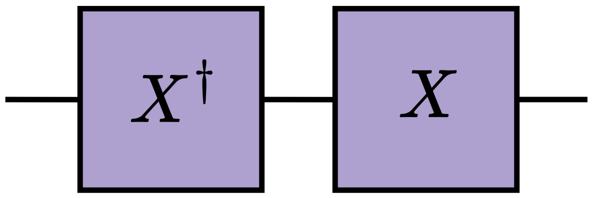
with the Kraus tensor from Eq. (2.69)

$E_q^{[l,l+1]} =$  , (3.24)

where for the second equality we have joined the legs on each side into a single row $s_{l,l+1}^{out} \in [d^2]$ and column $s_{l,l+1}^{in} \in [d^2]$ index. If we now merge the Kraus rank index q with the row dimension $s_{l,l+1}^{out}$, i.e. we *stack* the Kraus operators $\{E_q\}_{q\in[k]}$ along their row dimension, we obtain a $(kd^2) \times d^2$ matrix $X^{[l,l+1]}$

 . (3.25)

In this formulation, the trace-preserving condition of the Kraus channel from Eq. (2.35) can be expressed as

$$\mathbb{1} = X^{[l,l+1]\dagger} X^{[l,l+1]} =$$  . (3.26)

We recognize this relation for $X^{[l,l+1]}$ as the definition of the Stiefel manifold in Def. (2.7.7)! If we set $n = kd^2$ and $p = d^2$, we arrive at

$$X^{[l,l+1]} \in \mathrm{St}(n,p). \tag{3.27}$$

The procedure to transform an isometry $X^{[l,l+1]}$ into its superoperator representation $e^{\tau\hat{\mathcal{D}}^{[l,l+1]}}$ is shown in Alg. (4). We will label the whole isometry-superoperator transformation as $\mathrm{S_P}$, and define the map

$$\mathrm{S_P} : \mathrm{St}(n,p) \to \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2) : X \to \Lambda = (X)^{\mathrm{S_P}}. \tag{3.28}$$

The inverse map $\mathrm{S_T}$ follows immediately as

$$\mathrm{S_T} : \mathscr{T}(\mathscr{H}_1, \mathscr{H}_2) \to \mathrm{St}(n,p) : \Lambda \to X = (\Lambda)^{\mathrm{S_T}}. \tag{3.29}$$

The inverse algorithm to convert a superoperator into an isometry builds on top of Alg. (1) and is shown in Alg. (5).

Algorithm 4 Stiefel to Superoperator

```
Input: X: Isometry matrix ∈ St(n, p) associated to a quantum channel with $n = kd^2$ and $p = d^2$
Output: Λ: Superoperator representation of channel ∈ $\mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ of dimension $d^4 \times d^4$
Require: $n \geq p$
1: $E^{s_{in}}_{s_{out},q} \leftarrow$ Reshape isometry $X$ into a rank-three tensor of dimensions $d^2 \times k \times d^2$.
2: $E^{q}_{s_{out},s_{in}} \leftarrow$ Transpose indices of $E^{s_{in}}_{s_{out},q}$.
3: $\mathcal{C} \leftarrow$ Contract $\sum_q E^{q}_{s_{out},s_{in}} E^{q\dagger}_{s_{out},s_{in}}$ to get the Choi matrix using Eq. (2.49).
4: $\Lambda \leftarrow$ Apply row-reshuffling from Def. (2.4.4) to $\mathcal{C}$.
```

Algorithm 5 Superoperator to Stiefel

```
Input: Λ: Quantum channel in superoperator form ∈ $\mathscr{T}(\mathscr{H}_1, \mathscr{H}_2)$ of dimension $d^4 \times d^4$
Output: X: Isometry matrix ∈ St(n, p)
1: $E^{s_{in}}_{s_{out},q}$: rank-three tensor of dimensions $d^2 \times k \times d^2 \leftarrow$ Use Alg. (1) to obtain the Kraus tensor from Λ.
2: $X$: a matrix of dimensions $kd^2 \times d^2$ with $X^\dagger X = \mathbb{1} \leftarrow$ Merge Kraus index $q$ with $s_{out}$.
3: Define dimensions $n = kd^2$ and $p = d^2$, such that $n \geq p$.
```

3.3. Defining the cost function: approximation error

Now that our operators belong to the Stiefel manifold, we can construct a real-valued *cost function* f on this manifold, i.e.

$$f : \mathrm{St}(n, p) \to \mathbb{R}\,.$$

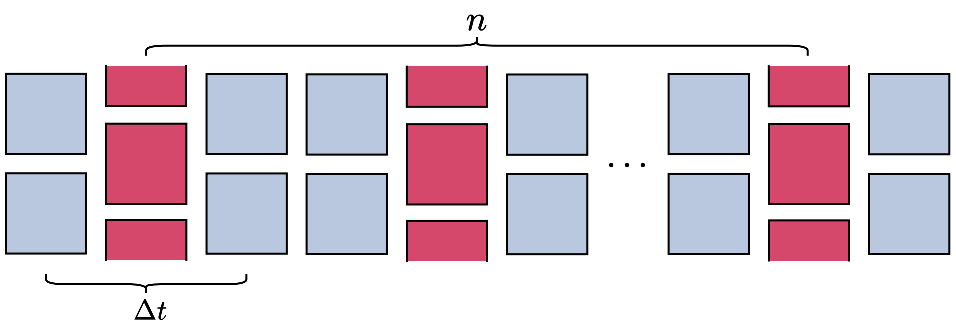
To this end, we look at the difference between the approximation given by the right-hand side of Eq. (2.64) and the full superoperator in the left-hand side. In particular, we use the computationally relevant Frobenius norm from Eq. (2.85)

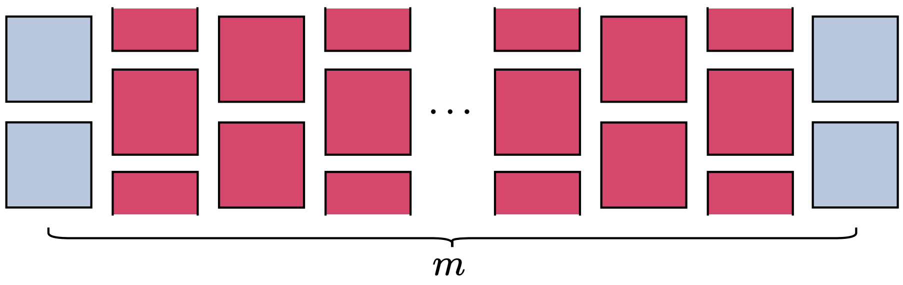
$$\|\mathrm{e}^{\tau\hat{\mathcal{L}}} - \mathrm{e}^{\tau\hat{\mathcal{L}}}_{\mathrm{approx}}\|\,. \tag{3.30}$$

It is customary to divide the time evolution up to a final time τ into smaller *time steps* of size Δt, such that

$$\tau = n \cdot \Delta t\,, \tag{3.31}$$

with $n \in \mathbb{N}$ the number of time steps. If we use the second-order Trotter splitting from Eq. (2.64) for each Δt, instead of the full time τ, we arrive at the following pattern for the approximate superoperators

$$e^{\tau\hat{\mathcal{L}}}_{\text{approx}} =$$  (3.32)

$$=$$  (3.33)

where the *blue* blocks correspond to the terms with *half-time* steps while the *red* blocks correspond to the full steps, i.e. the layers with $\Delta t/2$ and Δt in the exponent, respectively. The diagram in Eq. (3.33) is the result of merging the half-time steps in Eq. (3.32), and m is the *number of layers* in the final splitting

$$m = 3n - (n-1) = 2n + 1\,. \tag{3.34}$$

We see the final form of the splitting with n time steps showcases the same odd-even structure from Eq. (2.64). However, due to this new partition structure, the approximation error in terms of Δt becomes

$$e^{\tau\hat{\mathcal{L}}} = e^{\tau\hat{\mathcal{L}}}_{\text{approx}} + \mathcal{O}(\Delta t^2)\,. \tag{3.35}$$

Notice how the approximation error is now of second-order, contrary to the third-order of Eq. (2.64). The decrease in the order of the scheme is due to the composition of n time steps, where each of them introduces an error scaling as $\mathcal{O}(\Delta t^3)$, so that

$$n\mathcal{O}(\Delta t^3) = \frac{\tau}{\Delta t}\mathcal{O}(\Delta t^3) = \mathcal{O}(\Delta t^2)\,. \tag{3.36}$$

Alternatively, using the relations between Δt, n and m in Eqs. (3.31) and (3.34), we can express the error scaling in terms of these other parameters

$$\mathcal{O}(\Delta t^2) = \mathcal{O}(n^{-2}) = \mathcal{O}(m^{-2})\,. \tag{3.37}$$

Finally, in order to arrive at the final expression of the cost function, we lay out the assumptions used:

- To achieve translational symmetry, we assume the operators $\hat{\mathcal{D}}^{[l,l+1]}$ are the same for all $l \in [1, N]$.
- We use the splitting for n time steps to simulate the evolution up to time τ with m *distinct* layers. We choose all the layers to be different, contrary to the repeating structure from the original Trotter splitting, to increase the degrees of freedom of the optimization, in hopes of achieving a higher expressivity.
- As mentioned in § 3.1, we focus on the dissipative dynamics, omitting the Hamiltonian terms of the Lindblad superoperator $\mathcal{L}$.
- We restrict our optimization to the *real* Stiefel manifold to avoid problems with non-holomorphic functions defined on complex manifolds.

As explained in Def. (2.4.2), composing superoperators is achieved through the usual matrix multiplication, for which we overload the symbol $\cdot$. Then, by labeling the isometry on the α-th layer as X_α, we can express the cost function as

$$\|e^{\tau\hat{\mathcal{L}}} - \prod_{\alpha}^{m} \left((X_\alpha)^{\mathrm{S_P}} \right)^{L_G} \| \,, \tag{3.38}$$

where $\mathrm{S_P}$ is the isometry-superoperator map from Eq. (3.28) and L_G is the local-to-global transformation we introduced in Eq. (3.19). We can define the vector of isometries

$$\mathbf{X} := (X_1, \dots, X_m)\,, \tag{3.39}$$

and with it the superoperator $\mathcal{S}(\mathbf{X}) \in \mathbb{R}^{d^{2N} \times d^{2N}}$ acting on the full system, so the cost function takes the form

$$f(\mathbf{X}) = \|e^{\tau\hat{\mathcal{L}}} - \mathcal{S}(\mathbf{X})\| \,. \tag{3.40}$$

We will also refer to this cost function as the *approximation error* of the dissipative dynamics.

4. A Riemannian Approach to Lindbladian Dynamics

Now that we have developed the background theory and used it to formulate a real-valued cost function defined on the Stiefel manifold, it is time to tackle the optimization problem. We start this section by outlining the algorithm we implement to this end, shown in Alg. (6), followed by an in-detail explanation of each of its components. Let us dive in!

Algorithm 6 A Riemannian Approach to Lindbladian Dynamics

```
Input:
$L_k$: List of Lindblad operators acting on nearest-neighbors
$\tau$: Final time of Lindblad dissipative evolution
$n$: Number of time steps for Trotter splitting
$N$: Number of sites in the LPDO structure
$d$: Physical dimension per site in the LPDO
R: Desired rank of Kraus tensor acting on nearest-neighbors in the LPDO tensors
$I$: Number of iterations for trust-region optimization
Output:
$f_\beta$: Cost function history, evaluated after every iteration $\beta$ of the optimization
$\tilde{\mathbf{X}}$: List of isometries minimizing approximation error after $I$ iterations
Require: All $L_k \in \mathbb{R}^{d^2\times d^2}$
 1: $e^{\tau\hat{\mathcal{L}}} \leftarrow$ Create reference superoperator from $\{L_k\}_k$ and Eq. (2.63). ▷ Assume translational
    symmetry for all $N$ sites.
 2: $\{\Lambda_\alpha\}_\alpha \leftarrow$ Create $m = 2n+1$ superoperators using the Trotter splitting from Eq. (2.64)
    and the decomposition in Eq. (3.33).
 3: $\mathbf{X}^0 \leftarrow$ Generate ansatz of isometries from superoperators $\{\Lambda_\alpha\}_\alpha$.
 4: $f(\mathbf{X}^\beta, e^{\tau\hat{\mathcal{L}}}) \leftarrow$ Define cost function taking as input the exact superoperator $e^{\tau\hat{\mathcal{L}}}$ and list of
    isometries $\mathbf{X}^\beta$.
 5: grad $f \leftarrow$ Define Riemannian gradient from cost function.
 6: $R_X(Z) \leftarrow$ Define retraction from tangent space to Stiefel manifold.
 7: Hess $f \leftarrow$ Define Riemannian Hessian from cost function and Riemannian gradient.
 8: for $\beta \leftarrow 0$ to $I - 1$ do
 9:     $\mathbf{X}^{\beta+1} \leftarrow$ Perform an iteration of the trust-region algorithm with the conjugate gradient
10:             algorithm as inner iteration using $f$, grad $f$, Hess $f$, $R_X$, and $\mathbf{X}^\beta$.
11: end for
```

The entirety of this work has been implemented in `Python` [63] and can be found opensourced on GitHub as a package called `OpenTN` [64]. As a backend for manipulating the tensors and performing operations on them, we have used `NumPy` [65] and `JAX` [66]. The latter has also been utilized due to its *automatic differentiation* capabilities, as we will elaborate on the following sections.

4.1. Creating the ansatz: compressed factorization

As for many other gradient-based optimization algorithms, the initial optimization point, i.e. the *ansatz*, can play an important role in the number of iterations needed in the optimization scheme. Sometimes, it can even determine whether the algorithm reaches a local minimum or not. In this work, we choose to start at the even-odd splitting given by the second-order Trotter from Eq. (2.64). We have seen that for n time steps this ansatz results in m layers made up of local superoperators

$$\Lambda_\alpha \in \mathbb{R}^{d^4 \times d^4}, \quad \forall \alpha \in [m]\,. \tag{4.1}$$

The creation of this local ansatz of superoperators is implemented within our codebase using the function `get_general_trotter_local_ansatz`.

```python
from opentn.optimization import get_general_trotter_local_ansatz
# Lk is a list of Lindbladian operators acting on each pair of neighboring sites
superop_list = get_general_trotter_local_ansatz(lindbladians=Lk, tau=tau, n=n)
```

We can then convert each superoperator into its isometry representation using Alg. (5) through the $\mathrm{S_T}$ map, which builds on top of the superoperator-Kraus transformation from Alg. (1). In the latter, a crucial step is the Cholesky decomposition of the Choi matrix as shown in Eq. (2.56), and in particular, the degree of freedom present in the choice of the Kraus rank dimension R used in the factorization

$$\mathcal{C}_\mathrm{R} = \sum_{q=1}^{\mathrm{R}} E^q_{s^{out},s^{in}} (E^q_{s^{out},s^{in}})^\dagger\,, \tag{4.2}$$

with $s^{out}, s^{in} \in d^2$, and $\mathcal{C}_\mathrm{R}$ the rank-R approximation of $\mathcal{C}$. The obvious choice is to set $\mathrm{R} = \mathrm{rank}(\mathcal{C})$ so that $\mathcal{C}_\mathrm{R} = \mathcal{C}$. We call this the *natural* Choi rank of the channel. However, we can arbitrarily increase this number up to $\mathrm{R}_{max} = d^4$ or decrease it down to $\mathrm{R}_{min} = 1$. The former choice's advantage is that it improves the expressivity of each layer by increasing the number of parameters at the cost of increasing the complexity of the optimization. On the other hand, the latter choice decreases the parameter space of the optimization, decreasing its computational cost at the expense of a lower maximum possible accuracy. This second strategy can be used as well to control the growth of the Kraus dimension in the LPDO structure, where the indices r_l get contracted with the channel index q of dimension R, as shown in Eq. (2.58). In practice, we achieve the Cholesky factorization through the truncated SVD decomposition

$$\mathcal{C}_\mathrm{R} = VDV^\dagger = V\sqrt{D}\sqrt{D}V^\dagger = V'V'^\dagger\,, \tag{4.3}$$

where V and V' are $d^4 \times \mathrm{R}$ isometries, D is a $\mathrm{R} \times \mathrm{R}$ diagonal matrix, and $\sqrt{D}$ is its *element-wise* square root. The SVD scheme is preferred in this work over the direct Cholesky decomposition implemented in `NumPy`, as the latter suffers from numerical instabilities when the Choi matrix has negative eigenvalues that are zero up to numerical precision. We implement this procedure through the function `factorize_psd_truncated`.

```python
from opentn.transformations import factorize_psd_truncated
# C is any PSD operator and x its Cholesky factor
x = factorize_psd_truncated(C, rank=R)
```

Once we choose a rank R, we apply the S_T transformation on the list of superoperators $\{\Lambda_\alpha\}_{\alpha\in[m]}$ to obtain a list of isometries

$$\mathbf{X}^0 := (X_1^0, \dots, X_m^0), \quad X_\alpha \in \mathbb{R}^{n\times p}, \quad \forall \alpha \in [m], \tag{4.4}$$

where we have defined $n = Rd^2$ and $p = d^2$. The superscript 0 indicates that this corresponds to the *zero*-th iteration of the optimization scheme. This step is implemented in our code by the `super2ortho` function.

```python
from opentn.transformations import super2ortho
# iterate over the m superoperators
x_list = [super2ortho(superop, rank=R) for superop in superop_list]
```

4.2. Parametrizing the tangent space

Remark. As previously done in § 2.7, we will employ the symbols Z, W to label tangent vectors of the tangent space $T_X\mathrm{St}(n,p)$, contrary to η_x, ξ_x which were used in the context of a general manifold $\mathcal{M}$.

As briefly mentioned in § 2.8, the choice of the Riemannian metric, inducing an inner product on the tangent space, determines the form of the Riemannian elements. Thus, it might have an impact in the performance of the optimization algorithm. Therefore, in this section, we discuss the use of different tangent space metrics. To this end, let us start by following the procedure outlined in [67] to derive *parametrized* definitions for the tangent and normal spaces of the Stiefel manifold that will come in handy both for the metric discussions and the numerical implementations. We start by looking at the projection operator $\pi : \mathbb{R}^{n\times p} \to \mathrm{St}(n,p)$. This operation acts on an arbitrary rank-p matrix $Y \in \mathbb{R}^{n\times p}$ and projects it onto the closest point on $\mathrm{St}(n,p)$

$$\pi(Y) = \arg\min_{X\in\mathrm{St}(n,p)} \|Y - X\|^2, \tag{4.5}$$

where $\|.\|$ is the Frobenius norm induced by the Euclidean inner product from Eq. (2.84). Let us now look at the perturbed point $\pi(X + \epsilon Y)$, for $X \in \mathrm{St}(n,p)$, $Y \in \mathbb{R}^{n\times p}$, and $\epsilon \in \mathbb{R}$ a scalar. Since the Stiefel manifold does not fill the entire $\mathbb{R}^{n\times p}$ space, there are some directions Y that satisfy

$$\pi(X + \epsilon Y) = X + \mathcal{O}(\epsilon^2). \tag{4.6}$$

In other words, moving in the direction of Y does not take $\pi(X + \epsilon Y)$ away from X. Looking at Fig. (2.5a), we recognize the collection of such directions Y as the normal space! Using this practical definition, the tangent space can be defined as the orthogonal complement of the normal space.

Remark. For the orthogonality condition we use again the Euclidean inner product $\langle .,. \rangle_e$ inducing the Frobenius norm.

Now, to find the concrete parametrization of these spaces, let us first define the column orthogonal complement of the isometry $X \in \mathrm{St}(n,p)$ as $X_\perp \in \mathbb{R}^{n\times(n-p)}$, satisfying $[XX_\perp]^T[XX_\perp] = \mathbb{1}_n$, such that

$$X^T X_\perp = O_{p\times(n-p)} \tag{4.7}$$

$$X_\perp^T X = O_{(n-p)\times p} \tag{4.8}$$

$$X_\perp^T X_\perp = \mathbb{1}_{n-p} \tag{4.9}$$

$$XX^T + X_\perp X_\perp^T = \mathbb{1}_n\,, \tag{4.10}$$

where $\mathbb{1}_n$ is the $n \times n$ identity matrix and $O_{n\times m}$ is the $n \times m$ matrix with zero in all its elements. The side-by-side stacking of matrices $[AB]$ is a block-matrix yielding a *row vector* with two matrix elements (A, B). Analogously, the *column vector* $[AB]^T$ arises from the vertical stacking of the matrices. We can use these relations to define the decomposition of an arbitrary matrix $Y \in \mathbb{R}^{n\times p}$

$$Y = XA + X_\perp B + XC \tag{4.11}$$

in terms of a skew-symmetric matrix $A \in \mathbb{R}^{p\times p}$, a symmetric matrix $C \in \mathbb{R}^{p\times p}$, and an arbitrary matrix $B \in \mathbb{R}^{(n-p)\times p}$. We provide a proof for this decomposition in App. (A.1). This important decomposition is used in Lemma 8 of [67] to prove the relation

$$\pi(X + tY) = X + t(XA + X_\perp B) + \mathcal{O}(t^2)\,. \tag{4.12}$$

It is then straightforward to see that using $Y = XC$ we get $\pi(X + tXC) = X + \mathcal{O}(t^2)$, and for a small enough $|t| > 0$, this yields

$$\pi(X + tXC) = X\,. \tag{4.13}$$

We recognize this expression as the definition of the normal space we presented in Eq. (4.6), so we can provide an alternative *parametrized* definition for the normal space of the Stiefel manifold.

Definition 4.2.1 (Normal space - revisited)**.** Given an element $X \in \mathrm{St}(n,p)$ of the Stiefel manifold, the normal space $N_X\mathrm{St}(n,p)$ at X is

$$N_x\mathrm{St}(n,p) = \{N \in \mathbb{R}^{n\times p} : N = XC, C \in \mathbb{R}^{p\times p}, C = C^T\}\,. \tag{4.14}$$

This is an equivalent but more explicit definition than the one given in Eq. (2.116).

Inspired by Eq. (4.12), we can infer the form of the tangent space vectors to be $XA + X_\perp B$, which yields the parametrized definition of the tangent space.

Definition 4.2.2 (Tangent space - revisited)**.** Given an element $X \in \mathrm{St}(n,p)$ of the Stiefel manifold, the tangent space $T_X\mathrm{St}(n,p)$ at X is

$$T_X\mathrm{St}(n,p) = \{Z \in \mathbb{R}^{n\times p} : Z = XA + X_\perp B, A \in \mathbb{R}^{p\times p}, A + A^T = 0, B \in \mathbb{R}^{(n-p)\times p}\}\,. \tag{4.15}$$

If we compare Eqs. (2.97) and (4.15), we realize the product $X^T Z$ in the former is exactly the skew-symmetric matrix A of the latter, for $Z \in T_X\mathrm{St}(n,p)$.

It is straightforward to verify that $XA + X_\perp B$ is orthogonal to XC under the Euclidean inner product

$$\langle XA + X_\perp B, XC\rangle_e = \mathrm{tr}\{(XA + X_\perp B)^T XC\} = 0\,, \tag{4.16}$$

where we used $\mathrm{tr}(A^T C) = 0$. This relation helps reconcile the parametrized definition of the tangent space with the original description from Def. (2.7.8). From the revisited definitions of the tangent and normal space we can easily verify that at $X \in \mathrm{St}(n,p)$

$$\pi_{T,X}(Y) = (I - XX^T)Y + X\mathrm{skew}(X^T Y)$$

is the projection of an arbitrary matrix $Y \in \mathbb{R}^{n\times p}$ onto the tangent space, as introduced in Eq. (2.120). The first term comes from using Eq. (4.10) to obtain $X_\perp X_\perp^T = (I - XX^T)$ followed by the orthogonal condition in Eq. (4.8)

$$(I - XX^T)Y = X_\perp X_\perp^T Y = X_\perp B\,. \tag{4.17}$$

The second term uses Eq. (4.7) to obtain $X^T Y = A + C$, followed by the skew operator to *pick* the skew-symmetric factor A

$$\mathrm{skew}(X^T Y) = A\,, \tag{4.18}$$

which we multiply by X to get XA. Similarly, we derive the projection onto the normal space

$$\pi_{N,X}(Y) = X\,\mathrm{sym}(X^T Y) = \frac{X}{2}(Y^T X + X^T Y)\,, \tag{4.19}$$

that picks the symmetric factor

$$\mathrm{sym}(X^T Y) = C\,, \tag{4.20}$$

and multiplies it by X, yielding the parametrization in Eq. (4.14). We can simplify Eq. (2.120) to obtain a relation between $\pi_{N,X}$ and $\pi_{T,X}$

$$\begin{aligned}\pi_{T,X}(Y) &= Y - \frac{X}{2}(Y^T X + X^T Y)\\ &= Y - \pi_{N,X}(Y)\,.\end{aligned} \tag{4.21}$$

We recognize this relation as a reordering of the terms in Eq. (2.117). Our implementation uses the previously introduced `factorized_psd_truncated` function to compute $X_\perp$.

```python
from opentn.transformations import factorize_psd_truncated
# Projection onto the column space of X
P_x = x @ x.T
# Projection onto the orthogonal column space of X
P_x_comp = np.eye(n) - P_x
# Factorize P_x_comp = x_perp @ x_perp.T
x_perp = factorize_psd_truncated(P_x_comp, rank=n-p)
```

As a byproduct of the parametrized formulation, we can easily verify the degrees of freedom of $T_X\mathrm{St}(n,p)$ to be

$$np - \frac{p(p+1)}{2} = \frac{p(p-1)}{2} + p(n-p)\,. \tag{4.22}$$

The arbitrary matrix B contributes all its $p(n-p)$ parameters, while the skew-symmetric matrix A is completely defined by its

$$1+2+\ldots+(p-1)=\frac{p(p-1)}{2}$$

upper diagonal elements. We realize the left-hand side of Eq. (4.22) are the degrees of freedom of the normal space subtracted from the full space dimension np, confirming the relation in Eq. (2.117). A function returning the tuple (A_{params}, B_{params}) with the parameters of A and B from a tangent vector Z is implemented in our codebase as `parametrization_from_tangent`.

```
from opentn.stiefel import parametrization_from_tangent
parametrization_from_tangent(x:np.ndarray, z:np.ndarray, stack:bool=False, vectorized:bool=False)
```

4.3. Choosing a Riemannian metric

Let us employ the parametrization and dimension counting tools we have just described in the discussion of the Riemannian metric choice. The Euclidean metric $\langle .,. \rangle_e$ we have used thus far seems like the natural choice for a submanifold of the Euclidean space, and it is in fact the straightforward choice to define concepts like orthogonality in Eq. (4.16). However, it turns out this inner product does not weight the degrees of freedom of the Stiefel tangent space equally. To see this, let us look at the Euclidean inner product of $Z \in T_X\mathrm{St}(n,p)$

$$\begin{aligned}\langle Z,Z\rangle_e &= \operatorname{tr} A^TA + \operatorname{tr} B^TB && (4.23)\\ &= 2\sum_{i<j} a_{ij}^2 + \sum_{i,j} b_{ij}^2\,, && (4.24)\end{aligned}$$

where we have summed only over the upper diagonal for A, in agreement with Eq. (4.22). We can see from this equation that the Euclidean metric counts the degrees of freedom of A twice as much as the ones for B. As shown in [53], an alternative metric is the *canonical metric* defined for $Z, W \in T_X\mathrm{St}(n,p)$ as

$$\langle Z,W\rangle_c = \operatorname{tr}\left\{Z^T(\mathbb{1}-\frac{1}{2}XX^T)W\right\}. \tag{4.25}$$

Using this metric, the independent coordinates of A and B are weighted equally at every $X \in \mathrm{St}(n,p)$

$$\begin{aligned}\langle Z,Z\rangle_c &= \frac{1}{2}\operatorname{tr} A^TA + \operatorname{tr} B^TB && (4.26)\\ &= \sum_{i<j} a_{ij}^2 + \sum_{i,j} b_{ij}^2\,. && (4.27)\end{aligned}$$

Back in Def. (2.8.1), we introduced the concept of an inner product as a bilinear positive form g with a matrix representation G_X at an element X of a manifold. Since the Stiefel manifold is embedded in the Euclidean space, the inner product induced on each tangent space $T_X\mathrm{St}(n,p)$ is given by Eq. (2.111)

$$\langle Z, G_X W\rangle_e = \mathrm{tr}\left(Z^T G_X W\right) .$$

So far we have studied the Euclidean and canonical metric. However, it turns out these are not the only inner products that induce a Riemannian metric on the Stiefel manifold. In [56], a general formulation for the action of the ambient metric G_X at $X \in \mathrm{St}(n,p)$ is given by

$$G_X Z = \alpha_0(\mathbb{1} - XX^T)Z + \alpha_1 XX^T Z . \tag{4.28}$$

For $\alpha_0, \alpha_1 > 0$, this metric induces an inner product in $T_X\mathrm{St}(n,p)$ using the relation in Eq. (2.110), and hence it induces a Riemannian metric in $\mathrm{St}(n,p)$. The inverse matrix is

$$G_X^{-1} Z = \frac{1}{\alpha_0}(\mathbb{1} - XX^T)Z + \frac{1}{\alpha_1}XX^T Z. \tag{4.29}$$

Using Eq. (2.111), we conclude the metrics described thus far are special cases of the general formulation in Eq. (4.28):

- Euclidean metric ($\alpha_0 = \alpha_1 = 1$): $G_X^{[1,1]} = \mathbb{1}$
- Canonical metric ($\alpha_0 = 1$, $\alpha_1 = 1/2$): $G_X^{[1,1/2]} = \mathbb{1} - \frac{1}{2}XX^T$

As anticipated, the specific form of the gradient depends on the metric of choice. Using Eqs. (2.120) and (4.29) we can rewrite the Riemannian gradient expression in Eq. (2.118) as

$$\begin{aligned} \mathrm{grad}\, f(x) &= \pi_T\left(G^{-1}\,\mathrm{grad}\,\overline{f}(x)\right) \\ &= \frac{\mathrm{grad}\,\overline{f}(x)}{\alpha_0} + \frac{\alpha_1^{-1} - 2\alpha_0^{-1}}{2} XX^T\,\mathrm{grad}\,\overline{f}(x) - \frac{X\,\mathrm{grad}\,\overline{f}(x)^T X}{2\alpha_1} . \end{aligned} \tag{4.30}$$

Given a real-valued `function` and a list of isometries `x_list`, we implement this equation in our codebase in `gradient_stiefel_general`.

```python
from opentn.stiefel import gradient_stiefel_general
gradient_stiefel_general(function:callable, x_list:list[np.ndarray], alpha_0:float, alpha_1:float)
```

As shown in Eq. (2.131), the metric choice also affects the Riemannian connection, and with it, the Riemannian Hessian calculation. We can use Eq. (4.28) to obtain the cross-term in Eq. (2.133) for a point $X \in \mathrm{St}(n,p)$ and tangent vectors $Z, W \in T_X\mathrm{St}(n,p)$

$$\mathcal{X}(Z,W) = (\alpha_1 - \alpha_0)(ZW^T + WZ^T)X . \tag{4.31}$$

Then, using Eqs. (2.131–2.133), the general connection for the Stiefel manifold is given by [56]

$$\nabla_Z W = DW[Z] + \frac{1}{2}X(W^T Z + Z^T W) + \frac{\alpha_0 - \alpha_1}{\alpha_0}(\mathbb{1} - XX^T)(WZ^T + ZW^T)X . \tag{4.32}$$

This relation is implemented in our work through the function

```python
from opentn.stiefel import riemannian_connection
riemannian_connection(D_nu:np.ndarray, nu:np.ndarray, eta:np.ndarray, x:np.ndarray, alpha_0:float=1, alpha_1:float=1)
```

Remark. In the previous section § 4.2, we studied the projection onto the tangent and normal spaces. While these projections are metric-independent for the Stiefel manifold, this is in general not the case for every manifold [56].

4.4. Product of isometry manifolds

In Eq. (3.40), we established the main cost function we seek to minimize in this work, namely

$$f(\mathbf{X}) = \|\mathrm{e}^{\tau\hat{\mathcal{L}}} - \mathcal{S}(\mathbf{X})\| \,.$$

Since this function depends on m isometries, its domain is not a single Stiefel manifold, but rather a *product manifold*: $\mathrm{St}(n_1, p_1) \times \mathrm{St}(n_2, p_2) \ldots \times \mathrm{St}(n_m, p_m)$ (see Def. (2.7.5)). We will assume all the individual manifolds are of the same dimensions, and label the Cartesian product as $\mathrm{St}(n, p)^{\times m}$. We can then define the cost function f as the map acting on this product manifold

$$f : \mathrm{St}(n, p)^{\times m} \to \mathbb{R} : f(\mathbf{X}) \to \|\mathrm{e}^{\tau\hat{\mathcal{L}}} - \mathcal{S}(\mathbf{X})\| \,, \tag{4.33}$$

with $\mathbf{X} \in \mathrm{St}(n, p)^{\times m}$. The tangent space of a product manifold is in general the Cartesian product of the individual tangent spaces, but for our finite dimensional tensors it reduces to the *direct sum* of them [68]. We will see what this means in practice for our Riemannian gradient, Hessian and retraction.

4.5. Building the Riemannian gradient

A central element for the trust-region method used in Alg. (6) is the Riemannian gradient of the function f from Eq. (4.33). As mentioned in § 4.5, the tangent space of the product manifold is the direct sum of the individual tangent spaces. In practice, this means we can treat the tangent vectors Z, living on the tangent space $T_X \mathrm{St}(n, p)$ at $X \in \mathrm{St}(n, p)$, as independent of the rest and *concatenate* them to obtain $\boldsymbol{Z} \in T_{\mathbf{X}} \mathrm{St}(n, p)^{\times m}$. Based on the arguments developed in § 4.3, in this work, we will employ the equitative canonical metric. Consequently, the Riemannian gradient for every $X \in \mathbf{X}$ takes the form

$$\operatorname{grad} f(X) = \operatorname{grad} \overline{f}(x) + X \operatorname{grad} \overline{f}(x)^T X \,. \tag{4.34}$$

In this equation, the ambient gradient $\operatorname{grad} \overline{f}(X)$ is equal to the Euclidean gradient, so we can use Def. (2.6.2) to compute it. As aforementioned, to obtain $\operatorname{grad} f(\mathbf{X}) \in T_{\mathbf{X}} \mathrm{St}(n, p)^{\times m}$, we will concatenate the m individual Riemannian gradients, in other words, we will *stack them*. In our implementation, we make use of the automatic differentiation capabilities of `JAX` to this end.

```python
import jax.numpy as jnp
from opentn.import frobenius_norm
# cost function
f_stiefel = lambda x_list: frobenius_norm(model_stiefel_local(x_list, N, d), exp_Lvec)
# list with Euclidean gradients of length m
grads_ambient = jax.grad(f_stiefel)(x_list)
grads_ambient_stacked = jnp.stack(grads_ambient)
print(grads_ambient_stacked.shape)
>> (m, n, p)
```

In this code snippet, we have defined `model_stiefel_local` as the function that takes a list of local Stiefel matrices, transforms them onto superoperator layers, and composes them. By using `jnp.stack` we assume a vertical stacking, although this need not be the case, as long as we stick to a convention. After using the ambient gradients together with Eq. (4.34) to obtain the Riemannian gradients, a further intermediate step is the *parametrization* of the tangent vectors. As we saw in Eq. (4.22), the tangent spaces $T_X\,\mathrm{St}(n,p)$ are parametrized by

$$\frac{p(p-1)}{2} + p(n-p) = \frac{p(n-1)}{2}$$

degrees of freedom. Therefore, we can store and manipulate $mp(n-1)/2$ parameters, instead of mnp. To achieve this, we use Eqs. (4.17) and (4.18) to obtain A and B for each $\operatorname{grad} f(X)$, respectively, retrieving the $p(p-1)/2$ upper triangular elements of A and stacking them on top of all $p(n-p)$ elements of B. For convenience, we reshape all the stacked parametrized gradients into a vector of length $mp(n-1)/2$. We can obtain this vector with the previously introduced `parametrization_from_tangent` as follows.

```python
from opentn.stiefel import parametrization_from_tangent
parametrization_from_tangent(x, z, stack=True, vectorized=True)
```

We encapsulate the whole process into the function `gradient_stiefel_vec` which we convert into a lambda function `grad_stiefel` that calculates the parametrized vector given a list of isometries `x_list`.

```python
from opentn.stiefel import gradient_stiefel_vec
# convert this into a lambda function that takes only the list x
grad_stiefel = lambda x_list: gradient_stiefel_vec(x_list, f_stiefel, metric='canonical')
riemannian_gradient_vector = grad_stiefel(x_list)
print(riemannian_gradient_vector.shape)
>> (m p(n-1)/2, )
```

4.6. An appropriate retraction

Back in Def. (2.8.6) we introduced another of the cornerstone concepts we employ in our optimization algorithm: the retraction R_X. This is used within the trust-region algorithm to define the first-order model in Eq. (2.143) and to generate the next candidate at the β-th step in the optimization algorithm $X^{\beta+1} = R_{X^\beta}(Z)$, with Z the solution of the inner iteration in Eq. (2.146). As discussed in § 4.4, the tangent space $T_{\mathbf{X}}\mathrm{St}(n,p)^{\times m}$ of the product manifold $\mathrm{St}(n,p)^{\times m}$ results from the direct sum of the individual tangent spaces. Therefore, to implement a retraction from $\mathbf{Z} \in T_{\mathbf{X}}\mathrm{St}(n,p)^{\times m}$ onto $\mathbf{X} \in \mathrm{St}(n,p)^{\times m}$, all we need is to apply the retraction to each pair $(X_\alpha \in \mathbf{X}, Z_\alpha \in \mathbf{Z})$ individually. The first-order retraction for the Stiefel manifold used in this work is the *canonical generalized polar decomposition* introduced in [69] for any rectangular matrix $M \in \mathbb{R}^{n\times p}$

$$M = XC\,, \tag{4.35}$$

in terms of $X \in \mathbb{R}^{n\times p}$ an isometry and $C \in \mathbb{R}^{p\times p}$ a positive semidefinite matrix. The calculation of the isometry factor X was discussed in Eq. (2.128). However, an alternative calculation was introduced in [69] that avoids explicitly computing any inverse matrix, thus improving the stability of the procedure. Given the SVD decomposition of M,

$$M = UDV^T \,,$$

the isometry factor X is

$$X = U\,\mathbb{1}_{n,p}\,V^T \,, \tag{4.36}$$

with $\mathbb{1}_{n,p}$ the first p columns of the $n \times n$ identity matrix. In our code, we use the implementation given by `SciPy` [70] to compute the general polar decomposition. For $(X_\alpha \in \mathbf{X}, Z_\alpha \in \mathbf{Z})$ we define the following function corresponding to the retraction $R_{X_\alpha}(Z_\alpha)$:

```python
from scipy.linalg import polar
def polar_decomposition_rectangular(X:np.ndarray, Z:np.ndarray):
    """
    Retraction based on canonical polar decomposition of scipy
    """
    return polar(X+Z)[0]
```

Since, the tangent vectors within the trust-region method are parametrized and reshaped into a vector as discussed in § 4.5, the input to `polar_decomposition_rectangular` needs to be pre-processed as follows:

1. Divide the input vector $\mathbf{Z}_{\text{param}}$ of $mp(n-1)/2$ parameters into m batches.
2. Following the order by which these were stacked, each batch is further split into $p(p-1)/2$ and $p(n-p)$ parameters from where A_α and B_α are reconstructed, respectively.
3. Finally, we obtain
$$Z_\alpha = X_\alpha A_\alpha + X_{\alpha\perp} B_\alpha .$$

Steps (1) and (2) are implemented in `parametrizations_from_vector` while step (3) is implemented in `tangent_from_parametrization`. The retraction is then applied to each pair $R_{X_\alpha}(Z_\alpha)$, and the full procedure is encapsulated in the following function.

```python
from opentn.stiefel import parametrizations_from_vector, tangent_from_parametrization
def retract_stiefel(x_list:list[np.ndarray], zvec_param:np.ndarray):
    """
    Retraction of parametrized tangent vectors to Stiefel manifold
    """
    # get a list of tuples (A_{\alpha}, B_{\alpha}) for each \alpha in [1, m]
    params_list = parametrizations_from_vector(zvec_param, shapes=[op.shape for op in x_list])
    # compute tangent vector z_\alpha from its parametrization (A_{\alpha}, B_{\alpha})
    z_list = [tangent_from_parametrization(x, *params) for x, params in zip(x_list, params_list)]
    return [polar_decomposition_rectangular(x, z) for x,z in zip(x_list, z_list)]
```

4.7. Building the Riemannian Hessian

The last missing piece in our optimization algorithm is the calculation of the Riemannian Hessian. As we saw in Eq. (2.143), this linear mapping from $T_X\mathrm{St}(n,p)$ to $T_X\mathrm{St}(n,p)$ allows us to define a second-order approximation of the cost function f, hence, being a fundamental element of the trust-region algorithm. In Eq. (2.138), we defined the Riemannian Hessian-vector product for $Z \in T_X\mathrm{St}(n,p)$ as

$$\operatorname{Hess} f(X)[Z] = \nabla_Z \operatorname{grad} f(X) \,.$$

We have also discussed the general form of the Riemannian connection of the Stiefel manifold in Eq. (4.32), which for the canonical metric $G_X^{[1,1/2]}$ becomes

$$\nabla_Z W = DW[Z] + \frac{1}{2}X(W^TZ + Z^TW) + \frac{1}{2}(\mathbb{1} - XX^T)(WZ^T + ZW^T)X\,.$$

Together with Eq. (2.138), we obtain the relation

$$\begin{aligned}\operatorname{Hess} f(X)[Z] = D\operatorname{grad} f(X)[Z] &+ \frac{1}{2}X(\operatorname{grad} f(X)^T Z + Z^T \operatorname{grad} f(X)) \\ &+ \frac{1}{2}(\mathbb{1} - XX^T)(\operatorname{grad} f(X)Z^T + Z\operatorname{grad} f(X)^T)X\,.\end{aligned} \tag{4.37}$$

In order to compute the full Riemannian Hessian $\operatorname{Hess} f(X)$, we take inspiration from Eq. (2.76), where each column of the Euclidean Hessian $H_f(x)_k$ is *revealed* by the Hessian-vector product with the basis vectors $e_k \in \mathbb{R}^n$

$$H_f(x)_k = H_f(x)e_k\,.$$

Here, we will follow a similar approach. We generalized the concept of a basis vector on $\mathbb{R}^n$ with the coordinate vector fields or *elementary tangent directions* on the tangent space of a matrix manifold, introduced in Eq. (2.112). Using the canonical metric, the elementary tangent directions are [67]

$$E^A_{ij} = X(E_{ij} - E_{ji}), \quad E_{ij}, E_{ji} \in \mathbb{R}^{p\times p}\,, \tag{4.38}$$

$$E^B_{ij} = X_\perp E_{ij}, \qquad E_{ij} \in \mathbb{R}^{(n-p)\times p}\,, \tag{4.39}$$

where E_{ij} is the unit matrix from Eq. (2.82) and is defined for *appropriate* (i, j) index values. We notice Eq. (4.38) corresponds to the antisymmetric component XA of the tangent vectors, so (i, j) should traverse the $p(p-1)/2$ upper diagonal elements. On the other hand, Eq. (4.39) corresponds to the arbitrary component $X_\perp B$, so the indices (i, j) traverse all $p(n-p)$ values. We define the *flattened* index $\text{idx} \in [p(n-p)/2]$, and the corresponding elementary direction $E^{[\text{idx}]}$, such that

$$E^{[\text{idx}]} \in \begin{cases} E^A_{ij} & \text{if idx} \le \frac{p(p-1)}{2} \\ E^B_{ij} & \text{else} \end{cases}\,. \tag{4.40}$$

The corresponding python function is

```python
from opentn.stiefel import get_elementary_tangent_direction
get_elementary_tangent_direction(idx:int, x:np.ndarray)
```

Thus, for a single isometry X, we obtain the relation

$$\operatorname{Hess} f(X)_{\text{idx}} = \operatorname{Hess} f(X)[E^{[\text{idx}]}]\,. \tag{4.41}$$

The classical directional derivative $D\operatorname{grad} f(X)[Z]$ of the vector valued function $\operatorname{grad} f(X)$ is implemented through the Jacobian-vector product discussed in Def. (2.6.4). Once again, we take advantage of the automatic differentiation capabilities of `JAX`, and in particular we employ the *forward-mode* differentiation function `jvp` [71].

```python
from opentn.stiefel import gradient_stiefel_general
import jax
grad_function = lambda x_list: gradient_stiefel_general(x_list, f_stiefel, alpha_0=1, alpha_1=0.5)
# tangents is a list of m tangent vectors, one for each x in x_list
gradx, Dgradx_z  = jax.jvp(grad_function, (x_list,), (tangents,))
```

Besides returning $D\,\mathrm{grad}\, f(X)[Z]$, this function also returns the evaluation of $\mathrm{grad}\, f(X)$ needed in Eq. (4.37).

Remark. We can think of the computation of $D\,\mathrm{grad}\, f(X)[Z]$ as differentiating the *function* $\mathrm{grad}\, f$, resulting in another function $D\,\mathrm{grad}\, f$, which then takes two inputs (X, Z) and maps them onto the vector $D\,\mathrm{grad}\, f(X)[Z]$.

In order to accommodate this procedure to the product manifold structure, we compute the (idx, α) element of the Riemannian Hessian $\mathrm{Hess}\, f(\mathbf{X})$ using the tangent directions constructed as the m-length list

$$\text{tangents} = \left[O_{n\times p}, \ldots, E^{[idx]}, \ldots, O_{n\times p} \right] , \tag{4.42}$$

where the only non-zero matrix is the α-th element $E^{[\mathrm{idx}]}$. Then, each of the resulting elements in `Dgradx_z` is *projected* using the Riemannian connection in Eq. (4.32) to obtain the corresponding tangent vector in $T_{\mathbf{X}}\mathrm{St}(n,p)$. We use `parametrization_from_tangent` to extract and store the parametrization on $\mathrm{Hess}\, f(\mathbf{X})_{\mathrm{idx},\alpha}$. To obtain the full Riemannian Hessian, we repeat the process for all $\alpha \in [m]$ and $\mathrm{idx} \in [p(n-1)/2]$, and reshape the Riemannian Hessian into a $mp(n-1)/2 \times mp(n-1)/2$ dimensional matrix. The full procedure is implemented in the function `riemannian_hessian_vec` and we convert it into a lambda function `hessian_stiefel` that calculates the Hessian given a list of isometries `x_list`.

```python
from opentn.stiefel import riemannian_hessian_vec
hessian_stiefel = lambda x_list: riemannian_hessian_vec(x_list, f_stiefel, metric='canonical')
hessian_stiefel_matrix = hessian_stiefel(x_list)
print(hessian_stiefel_matrix.shape)
>> (mp(n-1)/2, mp(n-1)/2)
```

4.8. Optimization algorithm: trust-region

We have now put together all the building blocks we need for our optimization algorithm — namely the cost function `f_stiefel`, the retraction operator `retract_stiefel`, the gradient `grad_stiefel`, the Hessian `hessian_stiefel`, and the initial guess `x_list`. All that is left to do is to make use of these elements within the trust-region algorithm, together with the conjugate gradient as inner iteration, as discussed in § 2.9. We have wrapped the whole procedure into a single function `riemannian_trust_region_optimize`.

```python
from opentn.trust_region_rcopt import riemannian_trust_region_optimize
x_list_opt, f_opt, radius_opt = riemannian_trust_region_optimize(
cost_function = f_stiefel,
retraction = retract_stiefel,
gradient = grad_stiefel,
hessian = hessian_stiefel,
ansatz = x_list,
iter_num = I
radius_init = None
)
```

This returns:

- `x_list_opt` : the isometry history, i.e. a list of lists of isometries used at every iteration. The β-th element in the outer list corresponds to $\mathbf{X}^{\beta}$ for $\beta \in [I]$, with I the number of iterations in the optimization.
- `f_opt` : the optimization history, i.e. the cost function evaluated after every iteration β.
- `radius_opt` is the last trust-region radius Δ used in the optimization.

The last value $\tilde{\mathbf{X}} :=$ `x_list_opt[-1]` is the list of isometries optimizing our Stiefel-valued cost function after I iterations. These isometries can now be used to simulate the dissipative dynamics of an LPDO tensor network as explained in Alg. (3) or, together with `radius_opt`, they can be used to start the optimization anew to try to further minimize the cost function. In the next section, we will see this algorithm in action!

5. Numerical Simulations

In this chapter, we study the performance of the *Riemannian Approach to Lindbladian Dynamics* scheme introduced in Alg. (6). As a testbed, we focus on two models of nearest-neighbor dissipative interactions. Namely, at all neighboring sites $[l, l+1]$ of an N-sites quantum system we consider the Lindblad operators given by the models

1. Kitaev wire:
$$L_1 = \frac{\sqrt{\gamma}}{4}(a^\dagger \otimes \mathbb{1} + \mathbb{1} \otimes a)\,, \tag{5.1}$$
where $a^\dagger$ and a are the two-level quantum harmonic oscillator creation and annihilation operators, respectively, and $\mathbb{1}$ is the 2×2 identity matrix, as introduced in [9].

2. Pseudo Sparse Pauli-Lindblad (PSPL):
$$L_k = \sqrt{\gamma}(P_k - Q_k), \quad k \in [\kappa]\,, \tag{5.2}$$
where P_k, Q_k are elements of the Pauli-2 group, such that the total number of Lindblad operators $\kappa \ll 4^2 - 1$, i.e. the Lindblad terms are *sparse*. Compared to the original sparse Pauli-Lindblad model introduced in [10], here we have the difference of two Pauli operators instead of a single Pauli. In particular, in this work, we choose to use two Lindblad operators with weight-one Pauli terms
$$\begin{aligned} L_1 &= \sqrt{\gamma}(X \otimes \mathbb{1} - \mathbb{1} \otimes X)\,, \\ L_2 &= \sqrt{\gamma}(Z \otimes \mathbb{1} - \mathbb{1} \otimes Z)\,. \end{aligned} \tag{5.3}$$

For both models, the prefactor γ corresponds to the noise strength, which we normalize to $\gamma = 1$ over all simulations. Instead, we vary the final evolution time τ over the experiments, since this is equivalent to increasing the noise strength, so we can verify the validity of our results across different noise regimes. We have implicitly assumed the local Hilbert space dimension to be $d = 2$, i.e. we work with a system of qubits. We perform the simulations using the first non-trivial even number of sites $N = 4$ yielding an effective Hilbert space dimension $d^{2N} = 2^8$, as this keeps the computation tractable. In the following sections, we test our algorithm under the two aforementioned noise models, while exploring the influence of different simulation parameters and the meta-parameters we introduced in Chapter 4. Further, we compare the performance of our algorithm against the second-order Trotter splitting [6] and a structure-preserving scheme of arbitrary order introduced in [72]. To end this chapter, we leverage the translational invariance and PBC assumed in our ansatz to test the optimized isometries when embedding them into larger systems.

5.1. The importance of a good parametrization

In § 4.2, we discussed how to parametrize the tangent spaces of the Stiefel manifold. And later in § 4.3, we used these ideas to derive expressions that showcase the influence of the Riemannian metric in the calculation of the gradient and Hessian. We now explore how the choice of the metric affects the performance of the Riemannian optimization in practice.

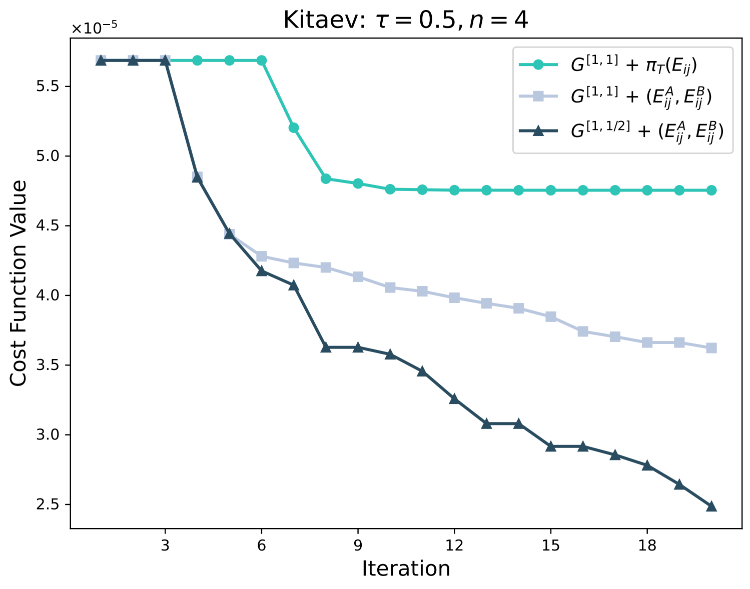
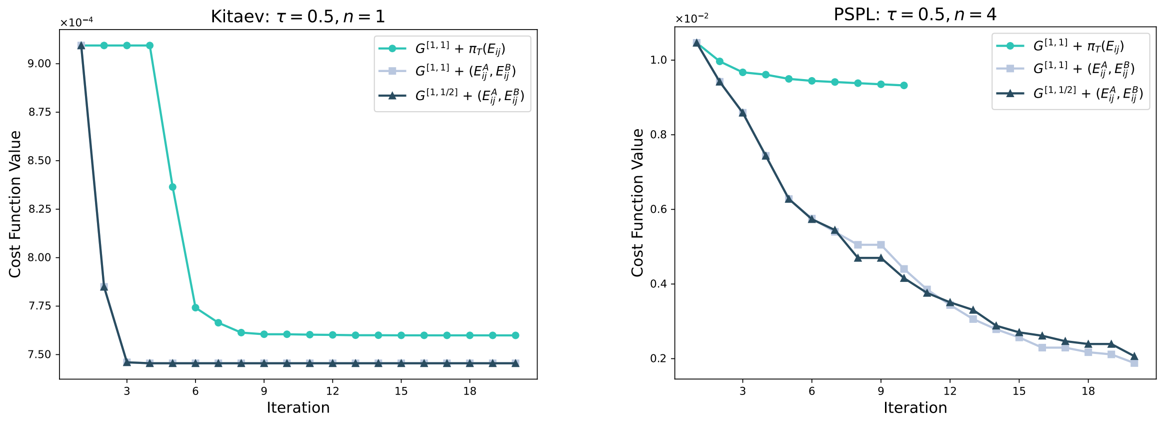


Figure 5.1.: Optimization of cost function for 20 iterations under the Kitaev model for $N = 4$ sites, evolution time of $\tau = 0.5$ and $n = 4$ time steps. Comparison of three different combinations of metric G and elementary tangent directions $E_{i,j}$ (see text), with the *dark blue* curve showing the choice used in this work.

To this end, we will carry our optimization scheme for three different combinations of metric and elementary tangent directions:

- Euclidean metric $G^{[1,1]}$ and a suboptimal elementary tangent direction $\pi_T(E_{ij})$, i.e. the unit matrix E_{ij} projected onto the tangent space.
- Euclidean metric $G^{[1,1]}$ and the elementary tangent directions (E^A_{ij}, E^B_{ij}) introduced in Eqs. (4.38) and (4.39).
- Canonical metric $G^{[1,1/2]}$ and elementary tangent directions (E^A_{ij}, E^B_{ij}).

Fig. (5.1) shows the value of the cost function $f(\mathbf{X}^\beta)$ for 20 iterations under the Kitaev model with $n = 4$ time steps and evolution time of $\tau = 0.5$. This comparison reaffirms the choice of the canonical metric for the Riemannian metric of the Stiefel manifold, as it clearly outperforms the optimization using the Euclidean counterpart. This difference, as argued in § 4.3, can be attributed to the correct weight assigned by the canonical metric to each degree of freedom parametrizing the tangent space vectors. On the contrary, the Euclidean metric assigns twice as much weight to the antisymmetric components, as shown in Eq. (4.24). In addition, the under-performance of the elementary directions $\pi_T(E_{ij})$, shown by the turquoise curve, confirms the importance of choosing properly how to traverse the tangent space when creating the Riemannian Hessian using Eq. (4.41). In Fig. (5.1), we also see that using this elementary direction results in a fast convergence to a suboptimal local minimum.



(a) Kitaev

(b) PSPL

Figure 5.2.: Optimization of cost function for 20 iterations and final evolution time $\tau = 0.5$. (a) Shows the optimization for the Kitaev model with one time step. (b) Shows the optimization for the PSPL model with $n = 4$ time steps. In this panel, the curve with $\pi_T(E_{ij})$ is only shown for 10 time steps as it reaches a saddle point early on.

However, it turns out the canonical metric is not always the obvious choice, as shown in Fig. (5.2). The plot in Fig. (5.2a) shows the optimization of the cost function under the Kitaev model and final evolution time $\tau = 0.5$ with a single time step $n = 1$, while Fig. (5.2b), uses the same final time, under the PSPL model with $n = 4$ time steps. Contrary to the results from Fig. (5.1), the comparisons in Fig. (5.2a) show that the Euclidean and canonical metrics yield very similar results when using the same elementary tangent directions. Moreover, at some intervals of the iterations, the Euclidean metric even outperforms the canonical metric, as shown in Fig. (5.2b).

5.2. Rank up

Another fundamental element in our optimization scheme was introduced in § 4.1: the rank R of the ansatz. This quantity arises in the Cholesky decomposition from Eq. (4.2) and determines the column dimension of the Stiefel manifold $p = \mathrm{R}d^2$. In this section, we will examine the conjectures presented in § 4.1, regarding the impact of the ansatz's rank on the performance of the Riemannian algorithm. To this end, we carry out the optimization scheme under the Kitaev and PSPL noise models, while choosing the ansatz's rank R as

$$R \in \{1, \dots, R_{\mathcal{C}}, \dots, d^4\}, \tag{5.4}$$

where $R_{\mathcal{C}} = \text{rank}(\mathcal{C})$ is the natural Choi rank, as introduced in Eq. (4.2).

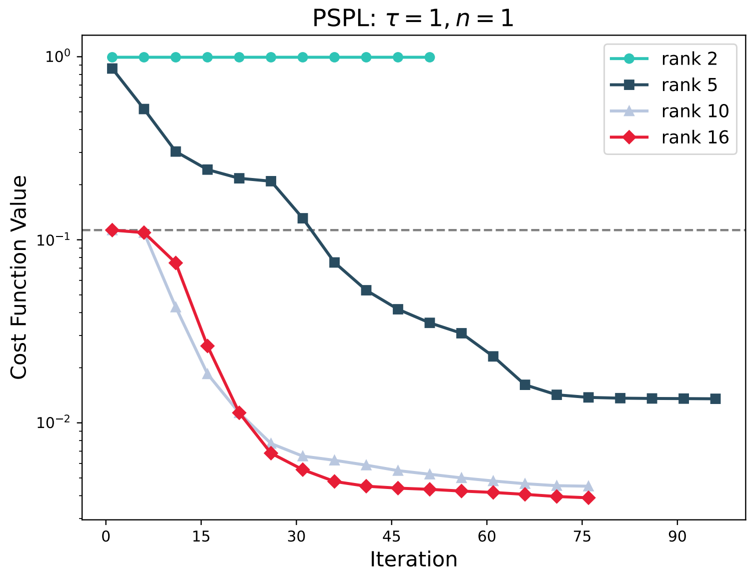


Figure 5.3.: Riemannian optimization with PSPL Lindblad model for a final time of $\tau = 1$ and $n = 1$ time steps. The four curves correspond to the use of different ranks for the ansatz. The gray dashed line marks the value of the Trotter splitting error.

We start by using Alg. (1) to transform the local superoperator arising from the PSPL Lindblad operators in Eq. (5.3), to obtain the natural Choi rank of the channel

$$R_{\mathcal{C}}^{\text{PSPL}} = 10 .$$

In other words, the PSPL quantum channel can be described exactly with 10 Kraus operators. We then perform the Riemannian optimization for a final time of $\tau = 1$ with $n = 1$ time steps and visualize the optimization trajectory in Fig. (5.3) using four different ranks

$$R \in \{2, 5, 10, 16\} .$$

As expected, this comparison illustrates how the algorithm's performance improves with increasing rank. The highest ranks $R \in \{10, 16\}$ start the optimization at the Trotter splitting value, marked by the *gray* dashed line, and exhibit a continuous decrease with the iterations. Due to the increased parameter space, the optimization with $R = 16$ attains the lowest cost function value out of all. However, a more surprising result is yielded by the curve with $R = 5 < R_{\mathcal{C}}^{\text{PSPL}}$. Since this ansatz corresponds to a *compressed* representation of the channel, it initially performs worse than the higher rank approximations, as we expected. Yet, after 30 iterations it already reaches the Trotter splitting value, and after 100 iterations, the cost function sees an improvement of nearly one order of magnitude with respect to the Trotter splitting and two orders of magnitude with respect to its ansatz value. This demonstrates that using this compressed representation not only enables a decrease in the approximation error of the dissipative dynamics but also helps control the exponential increase in the Kraus dimensions of the LPDO tensors after each dissipative step. This alleviates the computational cost associated with the noisy evolution of the LPDO structure that we discussed in § 2.5.

The other end of the spectrum is shown by the *turquoise* curve with $\mathrm{R} = 2$, where the optimization yields a considerably less drastic improvement. To understand this behavior, we can look at the Choi rank of the superoperator acting on all N sites arising from the rank-R ansatz, $\mathcal{S}^{\mathrm{R}}(\mathbf{X})$. As explained in detail in App. (A.2), the Choi rank of $\mathcal{S}^{\mathrm{R}}(\mathbf{X})$ is upper bounded by

$$\mathrm{R}_N := \mathrm{rank}\left\{ \left(\mathcal{S}^{\mathrm{R}}(\mathbf{X})\right)^{R_r} \right\} \leq \mathrm{R}^{2m} \,. \tag{5.5}$$

This bound reaffirms our choice of not using $\mathrm{R} = 1$, as this would result in a single Kraus operator for the full superoperator. To make this expression more tangible, we calculate numerically the Choi rank of the exact full superoperator $\mathrm{e}^{\tau\hat{\mathcal{L}}}$ and compare it to the rank of $\mathcal{S}^{\mathrm{R}}(\mathbf{X})$ for each of the ansatz ranks in our simulation.

R	R_N	$\mathrm{R}_N^{\mathrm{opt}}$
Exact	2^8	-
2	53	53
5	2^8	2^8
10	2^8	2^8
16	2^8	2^8

Table 5.1.: Comparison of full Choi ranks for different approximations using the PSPL model before and after optimization.

Table (5.1) shows both the initial rank R_N and the rank after the optimization $\mathrm{R}_N^{\mathrm{opt}}$. From this, we observe that unlike the higher-rank approximations, the rank-two ansatz reaches only $\approx 1/4$ of the full rank of $\mathrm{e}^{\tau\hat{\mathcal{L}}}$, even after the attempt to optimize it. This result agrees with Eq. (5.5), and provides an explanation for the limited performance shown in Fig. (5.3).

Contrary to what we initially expected based on Fig. (5.3), increasing the rank of the ansätze does not always lead to considerably better results. An example of this is the simulation under the Kitaev model shown in Fig. (5.4). For this computation, we use $\tau = 0.5$, $n = 4$, and choose R from $\{2, 4, 8, 16\}$. A first thing to notice is the natural Choi rank, which for this model is

$$\mathrm{R}_C^{\mathrm{Kitaev}} = 2\,.$$

For this reason, all the ansätze in this simulation start at the Trotter splitting value. While we notice a consistent decrease in the cost function for all the curves in Fig. (5.4), it is also

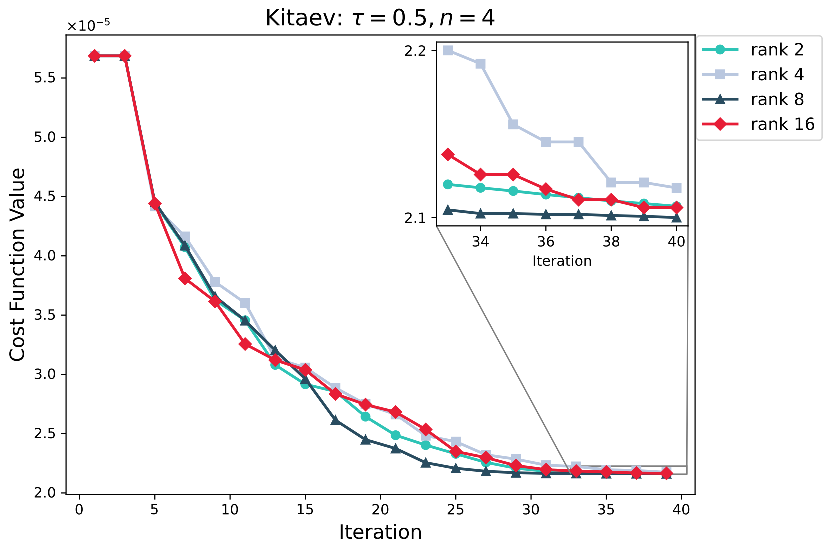


Figure 5.4.: Riemannian optimization with Kitaev Lindblad model for a final time of $\tau = 0.5$ and $n = 4$ time steps. The four curves correspond to the use of different ranks for the ansatz. The inset shows a zoom-in version of the last 10 iterations to emphasize the difference between the ansätze.

noteworthy that all of them reach a plateau at only 40 iterations, and none of them manage to attain a cost function lower than 30% of the original value. Analogously to the computation for the PSPL model, we calculate the Choi rank of $\mathcal{S}^{\mathrm{R}}(\mathbf{X})$ before and after the optimization, and show the results in Table (5.2).

R	R_N	$\mathrm{R}_N^{\mathrm{opt}}$
Exact	45	-
2	45	117
4	45	205
8	45	94
16	45	202

Table 5.2.: Comparison of full Choi ranks for different approximations using the Kitaev model before and after optimization.

This comparison highlights a couple of things. First, the exact superoperator can be represented by a relatively low rank, $\mathrm{R}_N = 45$, which is already achieved by all $\mathcal{S}^{\mathrm{R}}(\mathbf{X})$ even before the optimization. This can serve as an explanation for the low approximation error at which we start the optimization scheme, hence making it harder to improve upon. Second, in an attempt to improve on this approximation, the rank of all the ansätze R_N^{opt} increases considerably. This would result in a rapid increase of the Kraus dimensions of the LPDO structure, without necessarily achieving a substantial decrease in the approximation error of the dissipative dynamics.

5.3. Expressivity vs. iterations

In the previous section, we carried out the Riemannian optimization using $n = 1$ and $n = 4$ as the number of time steps for the PSPL and Kitaev models, respectively. However, an important question to ask when working with iterative optimization methods is *"How many iterations do we need for the algorithm to converge to a minimum?"*. In this section, we will try to address this question, by looking at the behavior of the cost function when increasing the number of time steps. This dependence is of particular interest to us, as the degrees of freedom in the optimization grow linearly with the number of time steps n, i.e. the number of time steps increases the expressivity.

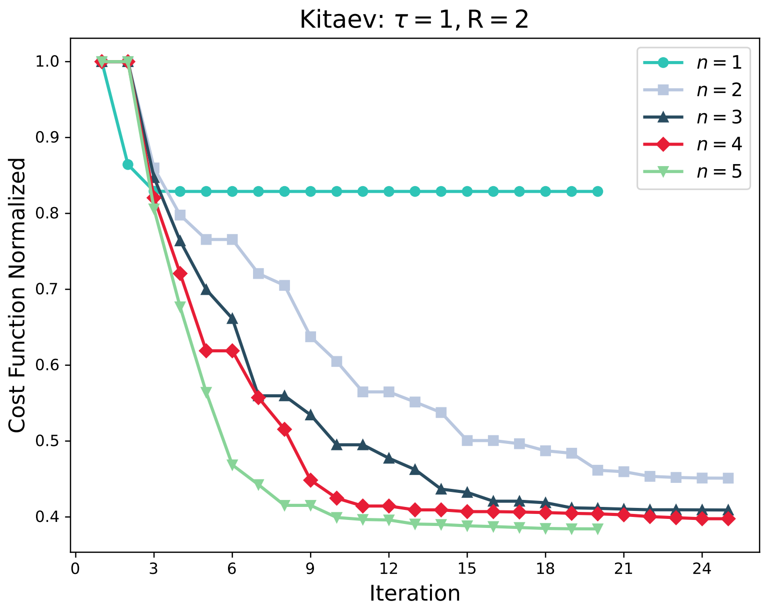


Figure 5.5.: Normalized approximation error as a function of the number of iterations for the Kitaev model, with $\tau = 1$ and $R = 2$, while varying the number of time steps n.

First, we look at the optimization with the Kitaev model, using $\tau = 1$ and $R = 2$ (its natural Choi rank), and vary the number of time steps n. In order to compare the decrease in the cost function for each value of n, we look at the *normalized* cost function trajectory

$$f_{\text{Norm}}(\mathbf{X}^{\beta}) = \frac{f(\mathbf{X}^{\beta})}{f(\mathbf{X}^{0})} \, . \tag{5.6}$$

The results shown in Fig. (5.5) provide insight to answer the question we posed at the beginning of this section. First, we notice that by increasing the number of time steps, thus increasing the expressivity, we can achieve a lower relative approximation error. We also observe that all the curves converge after less than 30 iterations. The curve with $n = 1$ shows the least improvement, as it reaches a plateau after only 5 iterations. While the curves for $n = 3, 4, 5$ all converge around the same point, surprisingly, the optimization using $n = 2$ takes the longest to settle.

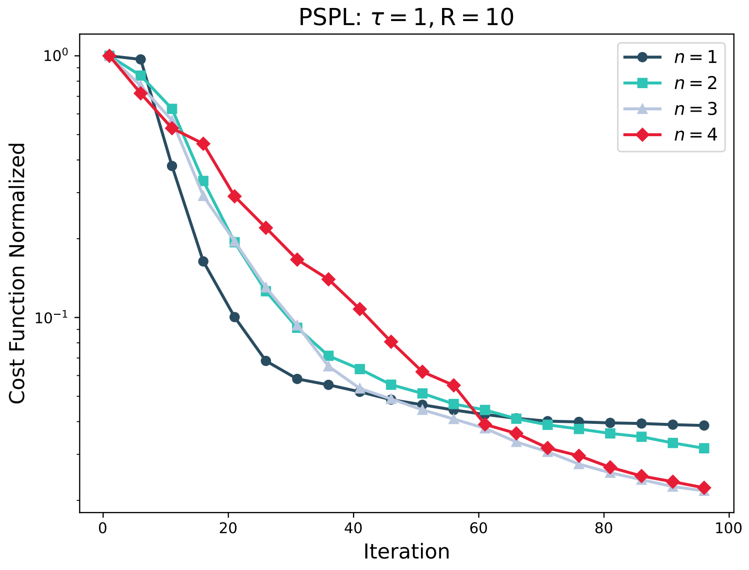


Figure 5.6.: Normalized approximation error as a function of the number of iterations for the PSPL model, with $\tau = 1$ and $R = 10$, while varying the number of time steps n.

We repeat an analogous simulation using the PSPL model, $\tau = 1$, and $R = 10$. Compared to the Kitaev scenario, the trajectories in Fig. (5.6) show that only the optimization using the lowest number of time steps $n = 1$ converges under 100 iterations. Even after decreasing the cost function over an order of magnitude, the curves for $n = 3$ and $n = 4$ still depict a steep descent towards the final iterations. We can attribute this behavior to the big parameter space induced by the high rank value $R = 10$, compared to the Kitaev rank $R = 2$, causing an increase in the complexity of the optimization. Similarly, we observe how the red curve corresponding to $n = 4$ has the worse performance up to the 60-th iteration, due to the increase in the parameter space, thus necessitating more iterations to find a minimum.

5.4. Riemannian to the test

Now that we have developed an understanding of how the input parameters in Alg. (6) affect the performance of the optimization, let us test our algorithm against other known schemes.

As shown in Eq. (3.35), the Trotter splitting of the dissipative dynamics using n time steps yields an error that scales with $\mathcal{O}(\Delta t^2)$. To compare the scaling of the approximation error

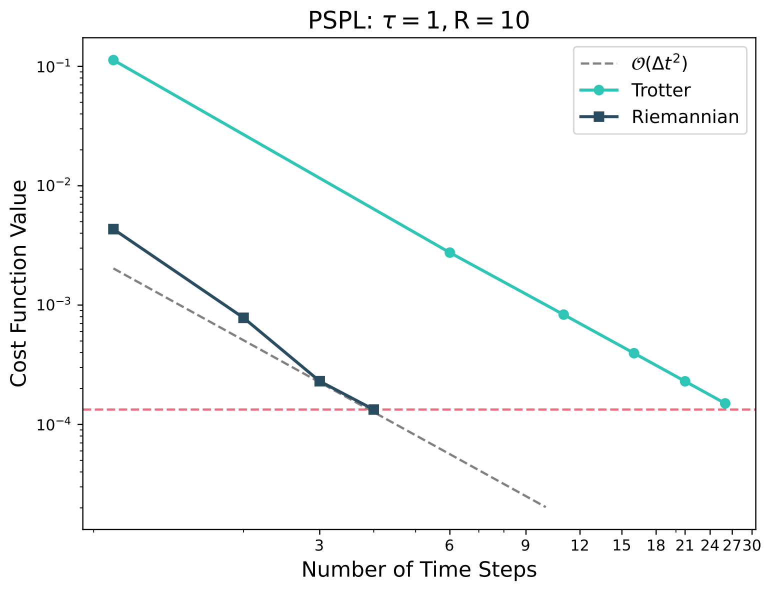


Figure 5.7.: Comparison of the error scaling between the Trotter splitting and Riemannian optimization using the PSPL noise model, final time of $\tau = 1$, and ansatz rank of $\mathrm{R} = 10$. The red horizontal line is used to emphasize the number of time steps needed by the Trotter splitting to achieve the same accuracy as the Riemannian optimized method.

achieved by our method, we perform the Riemannian optimization using the PSPL model, for a final time of $\tau = 1$, ansatz rank $\mathrm{R} = 10$, and plot it against the number of time steps n. We use the relation in Eq. (3.37) to plot a curve following the second-order scaling along the Riemannian trajectory. The results in Fig. (5.7) highlight the significant improvement attained by our optimization scheme, decreasing the error for every n by more than one order of magnitude, compared to the Trotter splitting. For instance, as emphasized by the red horizontal line, the Trotter splitting would need $n \approx 27$ to achieve the same accuracy obtained by only $n = 4$ using our method. For the higher number of time steps $n = 3$ and $n = 4$, it might appear as though the order of the scheme increases. However, since its trajectory remains parallel to the second-order curve, we realize it is actually the prefactor that improves, without changing the second-order scaling. We attribute this behavior to the increased expressivity of the ansatz, since the curve with $n = 4$ corresponds to $m = 9$ layers of isometries, with $mp(n-1)/2 = 702$ degrees of freedom we can optimize over, compared to the 234 parameters available in the optimization using $n = 1$.

Next, we benchmark the performance of our algorithm against the *structure-preserving* (SP) scheme introduced by Y. Cao and J. Lu in [72]. There, the authors introduce a family of *unnormalized* CP schemes of arbitrary order. To achieve a fair comparison against the SP methods, we use the normalized version of the schemes and compute the composite superoperator up to fourth order, due to the computational cost of higher-order schemes. To

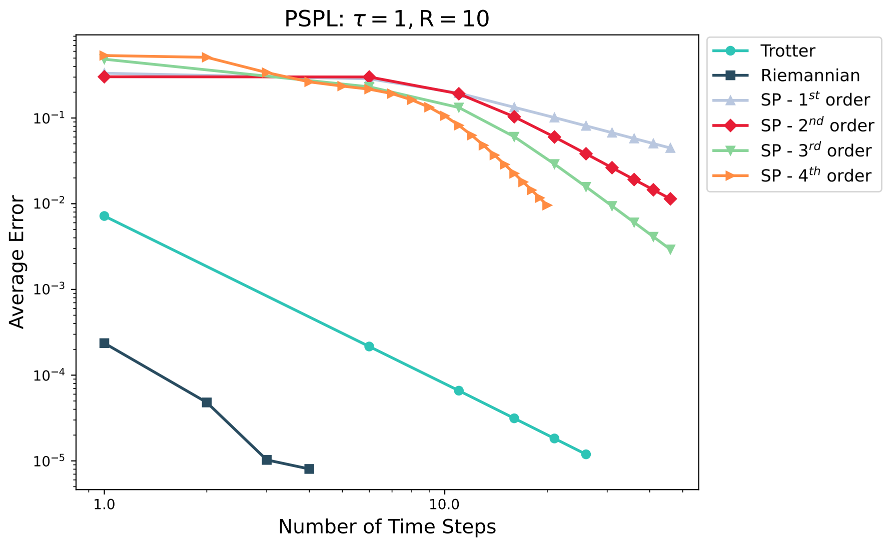


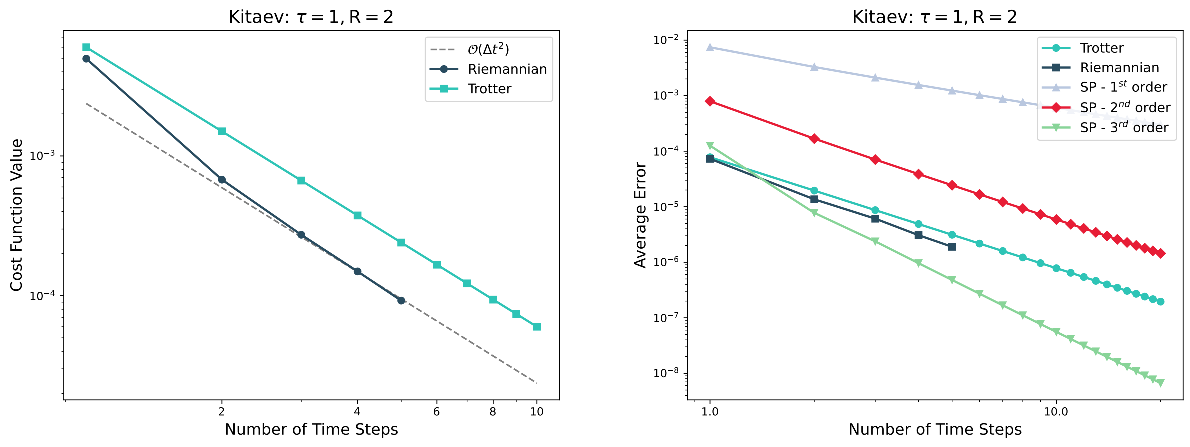
Figure 5.8.: Average error as a function of the number of time steps for different CPTP methods under the PSPL noise model, final time of $\tau = 1$, and ansatz rank of $R = 10$. We use the normalized version of the structure-preserving schemes of order $1, 2, 3$, and 4.

this end, we compute the average error as

$$\mathbb{E}\left(\|e^{\tau\hat{\mathcal{L}}}(\rho_0) - \mathcal{S}_n(\rho_0)\|\right), \tag{5.7}$$

where $\mathcal{S}_n$ is the superoperator from n time steps of either the normalized SP schemes, the Trotter splitting, or our Riemannian algorithm. The expectation value is taken over 500 randomly generated density matrices ρ_0. Once again, we run our simulations using the PSPL model, with a final time of $\tau = 1$ and ansatz rank of $R = 10$. The results in Fig. (5.8) showcase that while the structure-preserving schemes are capable of reaching approximations of arbitrarily high order, the error prefactor is larger than those for the Trotter and Riemannian methods. Moreover, the error order of these methods is only observed for higher number of time steps. Both of these behaviors are expected and explained in detail in [72].

To end this section, we repeat the above comparison using the Kitaev model, final evolution time $\tau = 1$, and ansatz rank $R = 2$. In Fig. (5.9a), we compare the Riemannian optimization against the Trotter splitting. From this, we observe how the Riemannian curve has an initial steep descent, corresponding to an approximation order higher than two. After $n = 3$, the decrease of the cost function stabilizes and accurately follows the second-order curve. As already demonstrated in previous sections, the improvement of the Riemannian optimization in this setting is not as significant as for the PSPL model. This is emphasized by Fig. (5.9b), where we compare the Riemannian optimization against the aforementioned SP schemes. We observe that the three SP schemes employed also suffer from a high error prefactor, as previously discussed. However, due to the higher approximation order, the order-three SP scheme quickly catches up, and for values of n close to 10 it attains an improvement of orders



(a) Comparison with Trotter

(b) Comparison with SP schemes

Figure 5.9.: Benchmark of the Riemannian optimization scheme against the Trotter and structure-preserving schemes under the Kitaev model, with $\tau = 1$, and $\mathrm{R} = 2$, plotted against increasing number of time steps. Panel (a) showcases the approximation order of the Riemannian and Trotter schemes. In panel (b), we use the normalized version of the structure-preserving schemes of order $1, 2$, and 3.

of magnitude compared to the Trotter splitting. For additional insight into the comparison between the superoperators achieved by each scheme, in App. (A.3), we compare the Choi ranks R_N for each of the approximation methods discussed in this section using both noise models.

5.5. Increasing the system size

In § 3.3, we discussed the assumptions we made when defining our cost function, one of which was the translational invariance of our ansatz. Additionally, we assumed all odd layers to follow periodic boundary conditions, as shown in Eq. (3.33). Due to these assumptions, although we performed the optimizations for systems of size $N = 4$, it is possible to employ the optimized isometries within larger systems. In this section, we will use the optimized layers from $N = 4$ and verify how well they approximate the dissipative dynamics of a system of $N = 6$ sites with an effective dimension of $d^{2N} = 2^{12}$.

For this comparison, we use the optimized isometries obtained in § 5.4 under the Kitaev and PSPL models, using $\tau = 1$ and their natural ranks $\mathrm{R} = 2$ and $\mathrm{R} = 10$, respectively. Fig. (5.10) shows the approximation error against the number of time steps n for the original $N = 4$ and the increased size $N = 6$. First, in Fig. (5.10a) we examine the Kitaev model comparison, from which we observe a mirroring behavior between the dissipative evolution of both system sizes. In other words, although the Trotter error increases with system size, so does the Riemannian error, yielding very similar curves. This behavior is emphasized by the

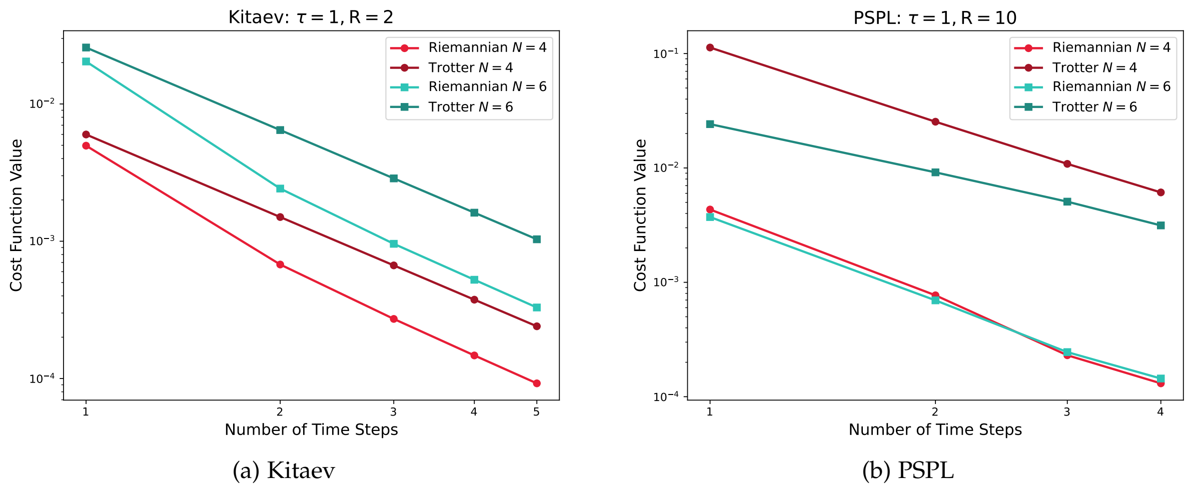
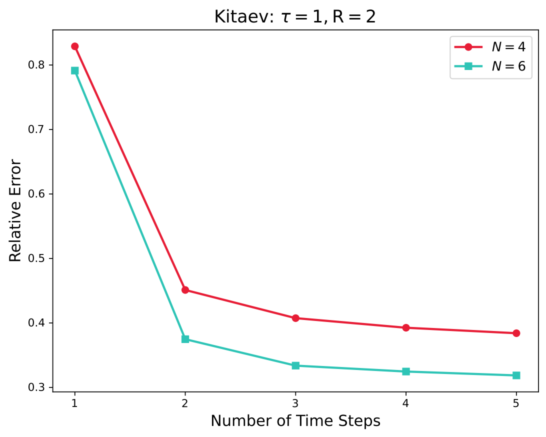


Figure 5.10.: Comparison of the cost function from Trotter and Riemannian schemes against the number of time steps n, with evolution time $\tau = 1$, for system sizes $N = 4$ and $N = 6$. The layers for the Riemannian scheme with $N = 6$ use the same isometries obtained with $N = 4$. Panel (a) shows the comparison for the Kitaev model with $\mathrm{R} = 2$, while panel (b) shows the comparison using the PSPL model with $\mathrm{R} = 10$.

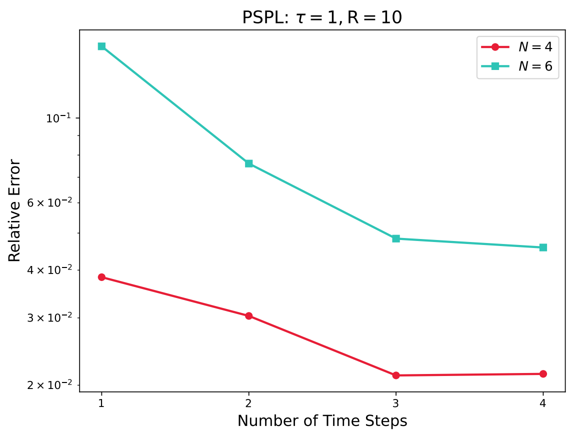
relative error between the Trotter and the Riemannian cost function values

$$f_{\text{relative}}(\tilde{\mathbf{X}}) = \frac{f_{\text{Riemannian}}(\tilde{\mathbf{X}})}{f_{\text{Trotter}}(\tilde{\mathbf{X}})}, \tag{5.8}$$

shown in Fig. (5.11a) for the Kitaev model. The results of the relative error plot are two-fold. First, it shows that the error achieved by the optimized isometries used within a bigger system faithfully follows the original approximation curve. Second, and more surprisingly, it shows that relative to the Trotter error, the Riemannian optimized isometries under the Kitaev noise work better for bigger systems. Similarly, Fig. (5.10b) shows the comparison using the PSPL model, yielding somewhat different results. In this, we observe how the Trotter and Riemannian errors are lower for the bigger system, being particularly true for the former scheme. However, for both methods, we observe that as the number of time steps increases, the slope of the error decreases for $N = 6$. From this, we can infer that for higher n, the error for the system with $N = 4$ may be lower. Due to this behavior, the relative error for the bigger system is worse than for the smaller one, as illustrated by Fig. (5.11b). Nevertheless, the trajectory for both system sizes follows a similar trend, indicating an improvement of orders of magnitude regardless of the system size.



(a) Kitaev



(b) PSPL

Figure 5.11.: Relative error between Trotter and Riemannian schemes against the number of time steps n, for system sizes $N = 4$ and $N = 6$, using an evolution time $\tau = 1$. Panel (a) shows the simulation using the Kitaev model with $\mathrm{R} = 2$, while panel (b) shows the simulation using the PSPL model with $\mathrm{R} = 10$.

6. Conclusion and Outlook

In this work, we have formulated the approximation error of the nearest-neighbor Lindbladian dynamics as an optimization problem defined on the Stiefel manifold. To this end, we have delved into the intricacies of the dissipative dynamics within the LPDO ansatz and devised transformations that allowed us to focus on the two-site Kraus channels. To find a solution to our optimization problem, we have employed the trust-region algorithm with the conjugate gradient as the inner iteration, requiring us to arrive at appropriate formulations for the Riemannian gradient, Riemannian Hessian, and retraction.

In our numerical experiments, we have studied the influence of the input parameters on the performance of the Riemannian algorithm. We have started by testing different choices of metrics and parametrizations for the Hessian construction. The results have shown that the canonical metric, weighting all degrees of freedom equally, performs in most cases at least as well as the embedded Euclidean metric. In some instances, it can even outperform the latter metric significantly, vouching for its use when working on the Stiefel manifold.

Furthermore, our simulations have also demonstrated the impact of the ansatz rank, which we have seen to vary based on the nature of the noise models. For instance, for noise channels with low natural Choi rank, like the Kitaev model with $R = 2$, the increase in rank, along with its associated computational cost, yields minimal benefit. On the other hand, noise channels with higher Choi rank, like the PSPL model with $R = 10$, showcase a direct correlation between the increase in the rank and a better approximation error. This allows us to trade computation time for accuracy. Interestingly, these simulations have also shown us that it is possible to use a compressed representation of the original quantum channel, and still achieve an improvement in the cost function of almost an order of magnitude compared to standard methods. This would allow us to alleviate the cost incurred by the increase of the Kraus indices of the LPDO structure after every dissipative step.

In addition, we have analyzed the behavior of the optimization trajectory when increasing the number of time steps, and hence the expressivity of the ansatz. The results have, once again, demonstrated a dependence on the noise model under study, with the high-rank channel necessitating considerably more iterations to converge, even for a low number of time steps, e.g. $n = 2$. This difference translates directly into the performance of the Riemannian algorithm. For the high-rank channel, the Riemannian algorithm has significantly outperformed the second-order Trotter and higher-order structure-preserving schemes by orders of magnitude. Conversely, the Riemannian algorithm used with the lower-rank channel has achieved a less significant improvement compared to the Trotter scheme and has been outperformed by the structure-preserving schemes of order higher than two. It is, however, worth highlighting that even in scenarios where the structure-preserving schemes outperform the Trotter and Riemannian approximations, they cannot be implemented directly on the

LPDO ansatz, as they do not preserve the local structure of the tensors.

Furthermore, we have leveraged the translational invariance and periodic boundary conditions inherent in our ansatz to test the approximation error yielded by isometries optimized in smaller systems when embedding them into larger systems. The results have shown good agreement among the curves of different system sizes. While the exact behavior varies based on the noise model, the overall trend demonstrates an improvement by the Riemannian approach over the Trotter splitting. This result is particularly valuable as it allows us to achieve an improvement in the dissipative evolution error of larger system sizes by optimizing computationally tractable systems. This is especially relevant as the dimension of the Lindblad superoperators grows as d^{2N} with the system size N.

These results have demonstrated that, under the correct noise models, our algorithm can improve both the approximation error of dissipative dynamics and the computational cost of each time step in the LPDO structure. A potential area for improvement is the wall time needed by our algorithm, with the most computationally expensive step being the construction of the Riemannian Hessian. To address this, we can implement a more efficient Hessian-vector product computation that does not need to construct the full Hessian. We leave this task for future work. Conversely, one might argue that the wall time of our algorithm should not be considered a strict constraint. This is especially true for a time-independent noise model, where our scheme can be seen as a single pre-processing step, entailing a *non-scaling* overhead.

It remains an open task to study the effect of the specific ansatz in the optimization algorithm, as discussed in [60], since this might influence the complexity of the optimization landscape, and therefore, affect the convergence point and speed. A similar open issue is to develop a rigorous mathematical understanding of the dependence of the algorithm's performance on the properties of the noise model. In this light, studying other physically meaningful noise models with high rank would provide further insight.

Our numerical implementation of the Riemannian gradient and Hessian is based on an automatic differentiation software framework, providing us great flexibility when defining the cost function. Therefore, generalizing the ansatz to a non-translational invariant ansatz would be relatively straightforward. Finally, a further generalization of this work could encompass the extension to the complex Stiefel manifold. This would require a recounting of the tangent space dimensions, and the correct formulation of the non-holomorphic complex derivatives, as outlined in [71].

A. Appendix

In this appendix, we include detailed calculations and proofs that were not explicitly given in any of the references, and that would be too lengthy to include in the main explanation.

A.1. Decomposition of an arbitrary element of $\mathbb{R}^{n\times p}$

In [67], the decomposition of an arbitrary matrix $Y \in \mathbb{R}^{n\times p}$ as shown in Eq. (4.11) is essential for deriving the tangent and normal space projectors. However, the proof for this decomposition is not provided there. Therefore, we will present a proof of this decomposition here.

Lemma: Given $X \in \mathrm{St}(n,p)$, and $X_\perp$ such that $[XX_\perp]^T[XX_\perp] = \mathbb{1}_n$, an arbitrary matrix $Y \in \mathbb{R}^{n\times p}$ can be decomposed as

$$Y = XA + X_\perp B + XC\,,$$

in terms of a skew-symmetric matrix $A \in \mathbb{R}^{p\times p}$, a symmetric matrix $C \in \mathbb{R}^{p\times p}$, and an arbitrary matrix $B \in \mathbb{R}^{(n-p)\times p}$.

Proof. An arbitrary square matrix $M \in \mathbb{R}^{p\times p}$ can be written as the sum [73]

$$M = \mathrm{skew}(M) + \mathrm{sym}(M) = A + C\,, \tag{A.1}$$

with $A = \mathrm{skew}(M)$ defined in Eq. (2.121) and

$$C = \mathrm{sym}(M) = \frac{1}{2}(M + M^T)\,. \tag{A.2}$$

We can then look at a matrix $Y' \in \mathbb{R}^{n\times p}$ and *block partition* it as a column vector of the form

$$Y' = \begin{pmatrix} M \\ B \end{pmatrix}\,,$$

with $M \in \mathbb{R}^{p\times p}$ and an arbitrary matrix $B \in \mathbb{R}^{(n-p)\times p}$. Then, we can decompose it using Eq. (A.1)

$$Y' = \begin{pmatrix} \mathbb{1}_p \\ O_{(n-p)\times p} \end{pmatrix} A + \begin{pmatrix} \mathbb{1}_p \\ O_{(n-p)\times p} \end{pmatrix} C + \begin{pmatrix} O_p \\ \mathbb{1}_{n-p} \end{pmatrix} B\,.$$

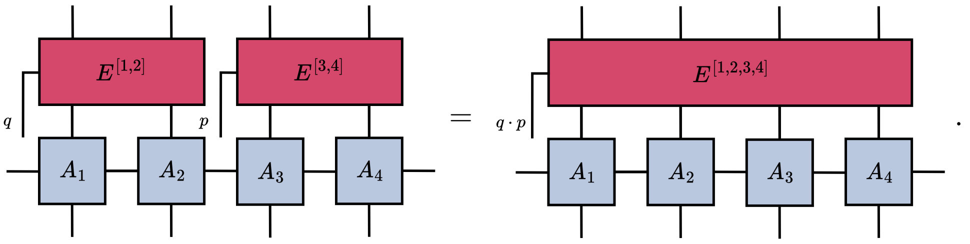
Finally, by multiplying the row vector $(X, X_\perp)$ from the left, which is a *comformable* partition yielding well-defined operations [74], we obtain

$$\begin{aligned}(X, X_\perp)\, Y' &= (X, X_\perp) \left[\begin{pmatrix} \mathbb{1}_p \\ O_{(n-p)\times p} \end{pmatrix} A + \begin{pmatrix} \mathbb{1}_p \\ O_{(n-p)\times p} \end{pmatrix} C + \begin{pmatrix} O_p \\ \mathbb{1}_{n-p} \end{pmatrix} B \right] \\ Y &= XA + X_\perp B + XC\,,\end{aligned}$$

where we have defined $Y := (X, X_\perp)\, Y'$ to finish the proof. $\square$

A.2. Rank of Kraus channels

Here, we derive an expression for the Rank R_N of the Kraus channel $\mathbf{E}$ acting on all N sites of an open quantum system in its LPDO form in terms of the ranks of the nearest-neighbor Kraus channels. First, we look at the action of each layer of local Kraus channels on the LPDO structure. For $N = 4$, the diagram representation is



(A.3)

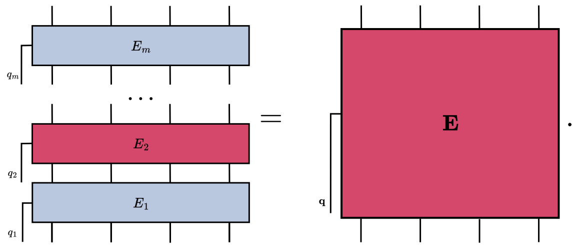
We notice that the single Kraus channel acting on all sites is obtained by merging the individual channel indices, hence, multiplying their ranks

$$\mathrm{rank}(E^{[1,2]} \otimes E^{[3,4]}) = \mathrm{rank}(E^{[1,2]})\, \mathrm{rank}(E^{[3,4]})\,. \tag{A.4}$$

Generalizing for any N, yields the expression

$$\mathrm{rank}\left(\bigotimes_{l=1}^{N/2} E^{[l,l+1]} \right) = \prod_{l=1}^{N/2} \mathrm{rank}(E^{[l,l+1]})\,. \tag{A.5}$$

Next, we compose the $m = 2n + 1$ layers of Kraus channels coming from a splitting using n time steps



(A.6)

At this point, we might be tempted to assume the total Kraus rank is obtained by multiplying the rank of each layer. However, it turns out this is only an upper bound of the actual value

$$\mathrm{R}_N := \mathrm{rank}(\mathbf{E}) \leq \prod_{\alpha=1}^{m} \mathrm{rank}(E_\alpha)\,. \tag{A.7}$$

In this work, we assume all two-local Kraus channels in layer α are the same, and thus have the same rank, i.e. they are translational invariant. Further, we choose the rank of our ansatz homogeneously across all layers, so if we let the ansatz rank be R, Eq. (A.7) becomes

$$\mathrm{R}_N = \mathrm{rank}(\mathbf{E}) \leq \mathrm{R}^{\frac{mN}{2}}\,. \tag{A.8}$$

A.3. Comparison of ranks for different approximation schemes

In this section, we present additional data demonstrating the increase in the Choi rank R_N of the superoperator acting on all $N = 4$ sites as the number of time steps n increases. We provide a comparison of the ranks obtained using the Trotter, Riemannian, and SP schemes, which were introduced in § 5.4. The comparison for the Kitaev model with $\tau = 1$ and R = 2 is shown in Table (A.1).

n	Trotter	Riemannian	SP 1st order	SP 2nd order	SP 3rd order
1	36	46	5	19	29
2	45	68	19	45	45
3	45	87	45	45	45
4	45	130	45	45	45
5	45	160	45	45	45

Table A.1.: Comparison of full Choi ranks for different approximation schemes using the Kitaev model for increasing number of time steps.

Analogously, in Table (A.2), we show the comparison of the ranks when using the PSPL model with $\tau = 1$ and R = 10.

n	Trotter	Riemannian	SP 1st order	SP 2nd order	SP 3rd order	SP 4th order
1	256	256	7	38	96	190
2	256	256	38	241	256	256
3	256	256	145	256	256	256
4	256	256	241	256	256	256
5	256	-	256	256	256	256

Table A.2.: Comparison of full Choi ranks for different approximation schemes using the PSPL model for increasing number of time steps.

List of Figures

List of Tables

Bibliography

[1] H.-P. Breuer and F. Petruccione. "The theory of open quantum systems". Oxford: Oxford Univ. Press, 2010. ISBN: 9780198520634.

[2] D. Manzano. "A short introduction to the lindblad master equation". In: *Aip advances* 10.2 (Feb. 2020). ISSN: 2158-3226.

[3] M. B. Hastings. "Solving gapped hamiltonians locally". In: *Physical review b* 73.8 (Feb. 2006). ISSN: 1550-235X.

[4] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac. "Matrix product density operators: simulation of finite-temperature and dissipative systems". In: *Physical review letters* 93.20 (Nov. 2004). ISSN: 1079-7114.

[5] G. D. l. Cuevas, N. Schuch, D. Pérez-García, and J. Ignacio Cirac. "Purifications of multipartite states: limitations and constructive methods". In: *New journal of physics* 15.12 (Dec. 2013), p. 123021. ISSN: 1367-2630.

[6] A. H. Werner, D. Jaschke, P. Silvi, M. Kliesch, T. Calarco, J. Eisert, and S. Montangero. "Positive tensor network approach for simulating open quantum many-body systems". In: *Physical review letters* 116.23 (June 2016). ISSN: 1079-7114.

[7] M. Suzuki. "General theory of fractal path integrals with applications to many-body theories and statistical physics". In: *Journal of mathematical physics* 32.2 (Feb. 1991), pp. 400–407. ISSN: 0022-2488.

[8] P.-A. Absil, R. Mahony, and R. Sepulchre. "Optimization algorithms on matrix manifolds". Princeton University Press, 2008. ISBN: 9780691132983.

[9] S. Diehl, E. Rico, M. A. Baranov, and P. Zoller. "Topology by dissipation in atomic quantum wires". In: *Nature physics* 7.12 (Oct. 2011), pp. 971–977. ISSN: 1745-2481.

[10] E. van den Berg, Z. K. Minev, A. Kandala, and K. Temme. "Probabilistic error cancellation with sparse pauli–lindblad models on noisy quantum processors". In: *Nature physics* 19.8 (May 2023), pp. 1116–1121. ISSN: 1745-2481.

[11] C. J. Wood, J. D. Biamonte, and D. G. Cory. "Tensor networks and graphical calculus for open quantum systems". 2015.

[12] G. Lindblad. "On the generators of quantum dynamical semigroups". In: *Communications in mathematical physics* 48.2 (1976), pp. 119–130.

[13] J. Johansson, P. Nation, and F. Nori. "Qutip 2: a python framework for the dynamics of open quantum systems". In: *Computer physics communications* 184.4 (Apr. 2013), pp. 1234–1240. ISSN: 0010-4655.

[14] "Qutip: quantum toolbox in python".

[15] M. Am-Shallem, A. Levy, I. Schaefer, and R. Kosloff. "Three approaches for representing lindblad dynamics by a matrix-vector notation". 2015.

[16] G. T. Landi. "Quantum information and quantum noise". en. graduate course in Quantum Information and Quantum noise. 2018.

[17] H. D. Macedo and J. N. Oliveira. "Typing linear algebra: a biproduct-oriented approach". 2013.

[18] M. M. Wolf. "Mathematical introduction to quantum information processing". en. Graue Literatur. 2022.

[19] A. Ekert. "Introduction to quantum information science". Oxford, UK: Oxford University Press, 2006. ISBN: 978-0199215706.

[20] A. Zalcman. "What are "completely positive" and "cptp" quantum maps?" Sept. 2023.

[21] J. M. Howie. "Fundamentals of semigroup theory". Oxford: Clarendon Press, 1995. ISBN: 9780198511946.

[22] Y. Pang, T. Hao, A. Dugad, Y. Zhou, and E. Solomonik. "Efficient 2d tensor network simulation of quantum systems". 2020.

[23] S. Patra, S. S. Jahromi, S. Singh, and R. Orus. "Efficient tensor network simulation of ibm's largest quantum processors". 2023.

[24] J. Tindall, M. Fishman, E. M. Stoudenmire, and D. Sels. "Efficient tensor network simulation of ibm's eagle kicked ising experiment". In: *Prx quantum* 5 (1 Jan. 2024), p. 010308.

[25] J. Biamonte. "Lectures on quantum tensor networks". 2020.

[26] R. Orús. "A practical introduction to tensor networks: matrix product states and projected entangled pair states". In: *Annals of physics* 349 (Oct. 2014), pp. 117–158. ISSN: 0003-4916.

[27] J. C. Bridgeman and C. T. Chubb. "Hand-waving and interpretive dance: an introductory course on tensor networks". In: *Journal of physics a: mathematical and theoretical* 50.22 (May 2017), p. 223001. ISSN: 1751-8121.

[28] J. Biamonte and V. Bergholm. "Tensor networks in a nutshell". 2017.

[29] J. Hauschild and F. Pollmann. "Efficient numerical simulations with tensor networks: tensor network python (tenpy)". In: *Scipost physics lecture notes* (Oct. 2018).

[30] K. Kraus. "States, effects, and operations". Springer, Oct. 1983.

[31] M. A. Nielsen and I. L. Chuang. "Quantum computation and quantum information". Cambridge University Press, Dec. 2010. ISBN: 9781139495486.

[32] F. Verstraete and H. Verschelde. "On quantum channels". 2003.

[33] M.-D. Choi. "Completely positive linear maps on complex matrices". In: *Linear algebra and its applications* 10.3 (1975), pp. 285–290. ISSN: 0024-3795.

[34] A. Jamiołkowski. "Linear transformations which preserve trace and positive semidefiniteness of operators". In: *Reports on mathematical physics* 3.4 (1972), pp. 275–278. ISSN: 0034-4877.

[35] D. Dereniowski and M. Kubale. "Cholesky factorization of matrices in parallel and ranking of graphs". In: *Parallel processing and applied mathematics*. Ed. by R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Waśniewski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 985–992. ISBN: 978-3-540-24669-5.

[36] M. Fannes, B. Nachtergaele, and R. F. Werner. "Finitely correlated states on quantum spin chains". In: *Communications in mathematical physics* 144.3 (Mar. 1992), pp. 443–490.

[37] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. "Matrix product state representations". 2007.

[38] M. Zwolak and G. Vidal. "Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm". In: *Physical review letters* 93.20 (Nov. 2004). ISSN: 1079-7114.

[39] M. Kliesch, D. Gross, and J. Eisert. "Matrix-product operators and states: np-hardness and undecidability". In: *Phys. rev. lett.* 113 (16 Oct. 2014), p. 160503.

[40] H. Pichler, J. Schachenmayer, A. J. Daley, and P. Zoller. "Heating dynamics of bosonic atoms in a noisy optical lattice". In: *Phys. rev. a* 87 (3 Mar. 2013), p. 033606.

[41] D. Jaschke, S. Montangero, and L. D. Carr. "One-dimensional many-body entangled open quantum systems with tensor network methods". In: *Quantum science and technology* 4.1 (Nov. 2018), p. 013001. ISSN: 2058-9565.

[42] G. Vidal. "Efficient classical simulation of slightly entangled quantum computations". In: *Physical review letters* 91.14 (Oct. 2003). ISSN: 1079-7114.

[43] M. Kliesch, T. Barthel, C. Gogolin, M. Kastoryano, and J. Eisert. "Dissipative quantum church-turing theorem". In: *Phys. rev. lett.* 107 (12 Sept. 2011), p. 120501.

[44] S. Cheng, C. Cao, C. Zhang, Y. Liu, S.-Y. Hou, P. Xu, and B. Zeng. "Simulating noisy quantum circuits with matrix product density operators". In: *Physical review research* 3.2 (Apr. 2021). ISSN: 2643-1564.

[45] A. Müller, T. Ayral, and C. Bertrand. "Enabling large-depth simulation of noisy quantum circuits with positive tensor networks". 2024.

[46] N. Boumal. "An introduction to optimization on smooth manifolds". Cambridge University Press, 2023.

[47] J. Toulouse, R. Assaraf, and C. J. Umrigar. "Introduction to the variational and diffusion monte carlo methods". 2015.

[48] G. Evenbly and G. Vidal. "Algorithms for entanglement renormalization: boundaries, impurities and interfaces". In: *Journal of statistical physics* 157.4–5 (Apr. 2014), pp. 931–978. ISSN: 1572-9613.

[49] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo. “Neural-network quantum state tomography”. In: *Nature physics* 14.5 (Feb. 2018), pp. 447–450. ISSN: 1745-2481.

[50] G. Torlai, C. J. Wood, A. Acharya, G. Carleo, J. Carrasquilla, and L. Aolita. “Quantum process tomography with unsupervised learning and tensor networks”. 2020.

[51] I. Luchnikov, A. Ryzhov, S. Filippov, and H. Ouerdane. “Qgopt: riemannian optimization for quantum technologies”. In: *Scipost physics* 10.3 (Mar. 2021). ISSN: 2542-4653.

[52] J. H. Hubbard and B. H. West. “Differential equations: a dynamical systems approach”. Springer Nature, 1995.

[53] A. Edelman, T. A. Arias, and S. T. Smith. “The geometry of algorithms with orthogonality constraints”. 1998.

[54] N. Bourbaki. “Elements of mathematics. algebra 2 : chapters 4-7”. 1990. ISBN: 9783540642435.

[55] E. Meinrenken. “Differentiable manifolds lecture notes, ...”

[56] D. Nguyen. “Operator-valued formulas for riemannian gradient and hessian and families of tractable metrics”. 2021.

[57] S. Ruder. “An overview of gradient descent optimization algorithms”. 2017.

[58] R. Zhang. “Newton retraction as approximate geodesics on submanifolds”. 2020.

[59] R. Brieger, I. Roth, and M. Kliesch. “Compressive gate set tomography”. In: *Prx quantum* 4.1 (Mar. 2023). ISSN: 2691-3399.

[60] A. Kotil, R. Banerjee, Q. Huang, and C. B. Mendl. “Riemannian quantum circuit optimization for hamiltonian simulation”. 2023.

[61] H. Yoshida. “Construction of higher order symplectic integrators”. In: *Physics letters a* 150.5 (1990), pp. 262–268. ISSN: 0375-9601.

[62] S. Blanes and P. Moan. “Practical symplectic partitioned runge–kutta and runge–kutta–nyström methods”. In: *Journal of computational and applied mathematics* 142.2 (2002), pp. 313–330. ISSN: 0377-0427.

[63] Python Software Foundation. “Python 3.11”. [Computer software]. 2022.

[64] E. Godinez. “Opentn: a riemannian approach to the lindbladian dynamics of a locally purified tensor network”. [Online]. 2023.

[65] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.

[66] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. “JAX: composable transformations of Python+NumPy programs”. Version 0.3.13. 2018.

[67] J. Manton. "Optimization algorithms exploiting unitary constraints". In: *Ieee transactions on signal processing* 50.3 (2002), pp. 635–650.

[68] M. Hauru, M. Van Damme, and J. Haegeman. "Riemannian optimization of isometric tensor networks". In: *Scipost physics* 10.2 (Feb. 2021). ISSN: 2542-4653.

[69] N. J. Higham, C. Mehl, and F. Tisseur. "The canonical generalized polar decomposition". In: *Siam journal on matrix analysis and applications* 31.4 (2010), pp. 2163–2180.

[70] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature methods* 17 (2020), pp. 261–272.

[71] @alexbw and @mattjj. "The autodiff cookbook — jax documentation".

[72] Y. Cao and J. Lu. "Structure-preserving numerical schemes for lindblad equations". 2021.

[73] R. A. Horn and C. R. Johnson. "Matrix analysis". Cambridge University Press, 2012. ISBN: 9781139788885.

[74] H. Anton. "Elementary linear algebra". 7th. ...provided the sizes of the submatrices of A and B are such that the indicated operations can be performed. New York: John Wiley, 1994, p. 36. ISBN: 0-471-58742-7.