Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Quantum Error Correction via Riemannian Optimisation

Master Candidate

Miguel Ángel Casanova Medina

Student ID : 2043187

Supervisor

Prof. Francesco Ticozzi

University of Padova

Co-supervisor

Prof. Kentaro Ohki

Kyoto University

Academic Year
2022/2023

To my parents,
cats and friends

Abstract

A quantum error correction protocol consists of (1) a set of quantum states faithfully representing some logical information (the code), and (2) a recovery map that is able to correct the effect of environmental noise on such states. The aim of the work is to develop an optimisation scheme that is able to find optimally correctable subspace codes for a known quantum noise channel, leveraging Petz recovery maps and nonlinear optimization methods based on the Stiefel manifold. In fact, in the proposed approach, given a candidate subspace to be associated to a code, we fix the recovery map as if the code was perfectly correctable. Therefore, optimisation must only be considered over the set of valid codes, not over the set of recovery operators. Optimisation over the codes is equivalent to optimisation over the complex-valued Stiefel manifold of the corresponding dimensions. In this thesis, which includes a brief mathematical introduction to the relevant elements of quantum mechanics, quantum error correction and Riemannian optimisation, the theoretical basis for the problem is developed and gradient-based local optimisation algorithms are discussed and tested, as well as two different kinds of global optimisers on the Stiefel manifold. The global optimisation algorithms are based on simulated annealing with intermittent diffusion, and a consensus based algorithm. Using these algorithms, correctable codes are found and compared to existing ones for three qubits subjected to bit-flip errors (single and correlated), four qubits undergoing local amplitude damping and five qubits subjected to local depolarising channels, and general single qubit errors.

Sommario

Un protocollo di correzione degli errori quantistici consiste in (1) un insieme di stati quantistici che rappresentano alcune informazioni logiche (il codice) e (2) una mappa di recupero in grado di correggere l'effetto dell'interazione dell'ambiente esterno con tali stati. Lo scopo di questo lavoro è sviluppare uno schema di ottimizzazione in grado di trovare codici ottimamente correggibili per un canale di rumore quantistico noto a priori, sfruttando mappe di recupero di Petz e metodi di ottimizzazione non lineari basati sulla varietà di Stiefel. Infatti, nell'approccio proposto, dato un sottospazio candidato ad essere associato ad un codice, fissiamo la mappa di recupero come se il codice fosse perfettamente correggibile. Pertanto, l'ottimizzazione deve essere considerata solo sull'insieme di codici validi, non sull'insieme di operatori di recupero. L'ottimizzazione sui codici equivale all'ottimizzazione sulla varietà di Stiefel a valori complessi delle dimensioni corrispondenti. In questa tesi, che include una breve introduzione matematica agli elementi rilevanti della meccanica quantistica, alla correzione degli errori quantistici e all'ottimizzazione riemanniana, vengono sviluppate le basi teoriche del problema e vengono discussi e testati algoritmi di ottimizzazione locale "gradient-based", nonché due diversi tipi di ottimizzatori globali sulla varietà di Stiefel. Gli algoritmi di ottimizzazione globale si basano sul simulated annealing con diffusione intermittente e su un algoritmo basato sui metodi di consenso. Utilizzando questi algoritmi, vengono trovati codici correggibili e confrontati con quelli esistenti tramite tre qubit soggetti a errori di bit-flip (singoli e correlati), quattro qubit sottoposti a smorzamento dell'ampiezza locale e cinque qubit soggetti a canali di depolarizzazione locali ed errori generali a singolo qubit.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

QEC Quantum Error Correction

QM Quantum Mechanics

NISQ Noisy Intermediate-Scale Quantum

CPTP Completely Possitive Trace Preserving

SDP Semidefinite Programming

IDDM Intermittent Diminishing Diffusion on Manifold

CBO Consensus-based Optimisation

SDE Stochastic differential equation

1 Introduction

The field of quantum science and technology is evolving at an impressive speed. IBM's latest quantum computer [12], Osprey, has 433 qubits. Moreover, they have planned Kookaburra, with over 4158 qubits, on their roadmap by 2025. That is about a tenfold increase in the amount of qubits in the span of little more than two years, five times faster than Moore's law. At the same time, ion-photon interfaces capable of generating entanglement every 5ms have already been developed [31]. However, mostly due to difficulties in producing scalable solid-state quantum processors, induced by the interaction with the environment, the industry is still in the Noisy Intermediate-Scale Quantum (NISQ) era. In order to overcome this era, different information protection methods are being designed. Among them are the exploitation of noise-free codes [23], quantum dynamical decoupling (QDD) [33] and quantum error correction (QEC) [14].

The mitigation of noise and errors in quantum systems requires innovative solutions. Particularly, the techniques used in classical information processing are not applicable to quantum systems, due to the no-cloning theorem [36]. Therefore, it not only is of vital importance to the progress of the entire field, but it also is a topic with a challenging nature.

In this work, we aim to explore a new way in which control and optimisation techniques can lead to better error suppression. Subspace codes are the main type of codes used in QEC, but it is known that finding correctable codes even if the system dynamics is perfectly known is an NP-Hard problem [4]. Nonetheless, relying on geometric and probabilistic intuition a numerical approach may be pursued: a subspace can be associated to a projector, which in turn can be expressed in terms of an orthonormal k-frame. Hence, the set of subspace codes is equal to the complex-valued Stiefel manifold. This observation opens the door to the use of Riemannian optimisation algorithms for the search of error correcting codes. As such, a suitable formulation of an optimisation problem equivalent to finding correctable Quantum Error Correction (QEC) codes has been designed, and three different algorithms performing optimisation on the Stiefel manifold, leading to the optimal code, have been implemented.

The work is divided into the following chapters,

- Quantum mechanics: All the algebraic concepts necessary to understand the mathematical structure of Quantum Mechanics (QM) are given first. Then, a brief introduction QM is given, with the aim of introducing all the elements necessary to understand QEC.
- Quantum error correction: The chapter starts with an introduction of the theory of QEC, including a series of fundamental theorems, that will allow us to determine the correctability of a code, and the most common fidelity measures. Then, a review of the existing work on QEC via optimisation is given. The chapter ends with the description of our proposed scheme for code optimisation.
- Riemannian optimisation: First, an introduction to Riemannian optimisation is given, including the necessary concepts from topology and differential geometry. Then, three different optimisation algorithms are described. The first is a local optimisation algorithm that is the adaptation gradient descent to Riemannian manifolds. The other two are global optimisation algorithms, Intermittent Diminishing Diffusion on Manifold (IDDM) and a Consensus-based Optimisation (CBO) algorithm.
- Analysis: The previously described algorithms were tested with three different kinds of noise channels, with the aim of recreating known correctable codes and testing the limitations, both of our methods and of QEC in general.

Then, the conclusions are given, together with suggestions for future work. Finally, the computation of the gradient of the cost function and the MATLAB code of the implemented algorithms are given in two appendixes.

2 Quantum mechanics

The underlying mathematical structure of quantum theories is that of C^*-algebras. In order to provide a full theoretical picture, this chapter starts by introducing the essential algebraic concepts. The relation to matrix algebra is given, as well as the arguments as to why finite dimensional QM can be faithfully represented in these terms. Finally, a brief introduction to QM is given, with a focus on the concept of quantum channels and quantum operations, which are extensively used in the following chapters.

2.1 Algebraic preliminaries

The most basic mathematical structure on which all of the following is based is that of vector space. For this reason, we start from this definition, and then we specialise it progressively, building up to C^*-algebras.

Definition 2.1.1. Let V be a set endowed with an addition operation $+ : V \times V \to V$, and a scalar multiplication $\cdot : \mathbb{F} \times V \to V$, where $\mathbb{F}$ is a field (e.g. $\mathbb{R}$, $\mathbb{C}$, etc), such that the following properties are satisfied for any $u, v, w \in V$ and $\alpha, \beta \in \mathbb{F}$

- Commutativity of the addition: $u + v = v + u$.
- Associativity of the addition: $(u + v) + w = u + (v + w)$.
- Existence of a null element $0 \in V$: $v + 0 = v$.
- Existence of an additive inverse $-v \in V$: $v + (-v) = 0$.
- Associativity of the scalar multiplication: $(\alpha\beta)v = \alpha(\beta v)$.

- Existence of the multiplicative identity $1 \in \mathbb{F}$: $1v = v$.
- Distributivity of the scalar multiplication with addition in V: $\alpha(u + v) = \alpha u + \alpha v$.
- Distributivity of the scalar multiplication with addition in $\mathbb{F}$: $(\alpha + \beta)u = \alpha u + \beta u$.

Such a structure $(V, +, \cdot)$ is called a vector space.

Now the notions of sizes, i.e. norms, and distances, i.e. metrics are introduced.

Definition 2.1.2. Given a vector space X, a map $\| \cdot \| : X \to [0, \infty)$ is called a norm if the following properties are satisfied for any $x, y \in X$ and $\alpha \in \mathbb{F}$

- $\|\alpha x\| = |\alpha| \|x\|$.
- $\|x + y\| \leq \|x\| + \|y\|$.
- $\|x\| = 0 \Rightarrow x = 0$.

A vector space endowed with a norm is called a normed vector space $(X, \|\cdot\|)$.

Definition 2.1.3. A map $d : X \times X \to \mathbb{R}$ is called a metric if it satisfies the following properties for any $x, y, z \in X$

- $d(x, y) \geq 0$.
- $d(x, y) = 0 \Leftrightarrow x = y$.
- $d(y, x) = d(x, y)$.
- $d(x, z) \leq d(x, y) + d(y, z)$.

A vector space endowed with a metric is called a metric space (X, d).

Of course, a metric can be induced from a norm as $d(x, y) = \|x - y\|$. The notion of completeness is given now, which leads to the definitions of Banach and Hilbert spaces.

Definition 2.1.4. Let (X, d) be a metric space. A sequence $\{x_n \in X\}$ is said to be Cauchy if given $\epsilon > 0$ there exists a natural number N_ϵ such that $d(x_m, x_n) < \epsilon$ for all $m, n > N_\epsilon$.

Definition 2.1.5. A metric space (X, d) is said to be complete if every Cauchy sequence converges in X.

Definition 2.1.6. A complete normed vector space is called a Banach space.

Definition 2.1.7. Let $\mathcal{H}$ be a complex vector space equipped with a scalar product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{C}$ satisfying

1. $\langle x, y \rangle = \overline{\langle y, x \rangle}$.
2. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$.
3. $\langle x, x \rangle \geq 0$, $\langle x, x \rangle = 0$ iff $x = 0$.

If it is complete with respect to the norm induced by the scalar product $\|x\|^2 = \langle x, x \rangle$, it is called a Hilbert space.

This definition implies that a Hilbert space is a Banach space where the norm induced by the scalar product.

Definition 2.1.8. Consider two Hilbert spaces $\mathcal{H}_1, \mathcal{H}_2$, and two scalar products $\langle x_1, y_1 \rangle_{\mathcal{H}_1} : \mathcal{H}_1 \times \mathcal{H}_1 \longrightarrow \mathbb{C}$ and $\langle x_2, y_2 \rangle_{\mathcal{H}_2} : \mathcal{H}_2 \times \mathcal{H}_2 \longrightarrow \mathbb{C}$. Now consider two operators $A : \mathcal{H}_1 \longrightarrow \mathcal{H}_2$ and $A^* : \mathcal{H}_2 \longrightarrow \mathcal{H}_1$, such that $\langle x_2, Ay_1 \rangle_{\mathcal{H}_2} = \langle A^* x_2, y_1 \rangle_{\mathcal{H}_1}$, $\forall x_2 \in \mathcal{H}_2,\ y_1 \in \mathcal{H}_1$, then A^* is said to be the Hermitian adjoint of A.

The Hermitian conjugate satisfies the following properties,

- Involutivity: $(A^*)^* = A$.
- Anti-linearity: $(A + B)^* = A^* + B^*$ and $(cA)^* = \overline{c}A^*$.
- Anti-distributivity: $(AB)* = B^* A^*$.

At this point, an algebra can be defined.

Definition 2.1.9. An algebra $\mathcal{A}$ over a field $\mathbb{F}$ is a vector space equipped with a multiplication, under which it is closed, which is linear on both factors, i.e. it must satisfy

$$c(AB) = (cA)B = A(cB)\ \forall c \in \mathbb{F},\ A, B \in \mathcal{A},$$

$$A(B + C) = AB + AC\ \forall A, B, C \in \mathcal{A},$$

$$(A + B)C = AC + BC\ \forall A, B, C \in \mathcal{A}.$$

The multiplication is commonly assumed to have an identity element

$$\mathbb{1}A = A \ \forall A \in \mathcal{A}.$$

If the multiplication is associative, $\mathcal{A}$ is said to be an associative algebra

$$(AB)C = A(BC) = ABC \ \forall A, B, C \in \mathcal{A}.$$

If the underlying vector space is endowed with a norm $\|\cdot\|$, $\mathcal{A}$ is a normed algebra is the following condition is satisfied

$$\|AB\| \leq \|A\|\|B\|.$$

Finally, if the underlying vector space is complete w.r.t. the distance induced by this norm, $\mathcal{A}$ is said to be a Banach algebra.

There are certain types of maps that are particular importance, namely homomorphisms, isomorphisms, automorphisms and isometries. The existence of these between different mathematical structures leads to important relations of equivalence and help us represent abstract objects in terms of more concrete ones.

Definition 2.1.10. A map $\Phi : \mathcal{A}_1 \longrightarrow \mathcal{A}_2$ is called a homomorphism if for $A, B \in \mathcal{A}_1$ and $\lambda \in \mathbb{C}$ one has that $\Phi(\lambda A) = \lambda\Phi(A)$, $\Phi(A+B) = \Phi(A) + \Phi(B)$ and $\Phi(AB) = \Phi(A)\Phi(B)$. If Φ is a one-to-one homomorphism, then it is called an isomorphism. Additionally, if $A_2 = A_1$ an isomorphism is called an automorphism, since it maps A_1 to itself.

Definition 2.1.11. Given metric spaces X_1 and X_2, an isometry is a map $\Phi : X_1 \longrightarrow X_2$ such that

$$d_{X_2}(\Phi(x), \Phi(y)) = d_{X_1}(x, y). \tag{2.1}$$

At this point, *-algebras and C^*-algebras can be defined.

Definition 2.1.12. A *-algebra is an algebra $\mathcal{A}$ with a map $^* : \mathcal{A} \longrightarrow \mathcal{A}$ that is an automorphism and an involution (i.e. it is its own inverse), satisfying the following properties of the Hermitian adjoint:

$$(A + B)^* = A^* + B^*,$$

$$(cA)^* = \overline{c}A^*,$$

$$(AB)^* = B^*A^*,$$

$$(A^*)^* = A.$$

A^* is called the adjoint of A. For a Banach *-algebra the following additional condition is required

$$\|A^*\| = \|A\|.$$

Definition 2.1.13. A C^*-algebra is a Banach *-algebra over the field of the complex numbers $\mathbb{C}$, which satisfies the C^*-condition, i.e.

$$\|A^*A\| = \|A\|\|A^*\|.$$

This condition is equivalent to what historically been also called the B^*-condition

$$\|AA^*\| = \|A\|^2.$$

An element $A \in \mathcal{A}$ is said to be self-adjoint or Hermitian if it is equal to its adjoint A^*. In this case the C^*-condition implies

$$\|A^2\| = \|A\|^2. \tag{2.2}$$

An element $A \in \mathcal{A}$ is said to be normal if it commutes with its adjoint, i.e. $AA^* = A^*A$. In this case the previous property also applies. Additionally, it is said to be unitary if its adjoint is its inverse, i.e. $AA^* = \mathbb{1}$.

Let us now introduce the notions of states and representations of a C^*-algebra.

Definition 2.1.14. A state ρ on a C^*-algebra is a positive linear functional if it is normalised, i.e. $\|\rho\| = 1$. The set $\mathcal{D}(\mathcal{A})$ of all states ρ in $\mathcal{A}$ is said to be its state space.

Definition 2.1.15. Let $\mathcal{A}$, $\mathcal{B}$ be two C^*-algebras, and let $\Phi : \mathcal{A} \to \mathcal{B}$ be a homomorphism, then it is called a *-homomorphism if $\Phi(A^*) = \Phi(A)^*$. If Φ is an isomorphism satisfying the same condition, then it is a *-isomorphism. $\mathcal{A}$, $\mathcal{B}$ are said to be *-isomorphic if there exists a *-isomorphism mapping $\mathcal{A}$ onto $\mathcal{B}$.

Definition 2.1.16. Let X_1, X_2 be a two vector spaces. Then $B(X_1, X_2)$ is the set of linear bounded operators from X_1 to X_2, and $B(X_1)$ is the set of linear bounded operators from X_1 to itself.

Definition 2.1.17. Let $\mathcal{A}$ be a C^*-algebra. A *-representation of $\mathcal{A}$ is a *-homomorphism π of $\mathcal{A}$ into $B(\mathcal{H})$ ($\mathcal{H}$ some Hilbert space). We shall denote this *-representation of $\mathcal{A}$ by $\{\pi, \mathcal{H}\}$.

Finally, the concept of universal *-representations provides a way to represent the elements of $\mathcal{A}$ as linear bounded operators on some Hilbert space, while preserving distances. In a probabilistic setting, this would imply the preservation of probabilities and expectation values, which is of key importance to QM.

Definition 2.1.18. Let $\mathcal{A}$ be a C^*-algebra an let $\mathcal{D}(\mathcal{A})$ be its state space. For arbitrary $\rho \in \mathcal{D}(\mathcal{A})$, consider the *-representation $\{\pi_\rho, \mathcal{H}_\rho\}$. Let $K = \bigoplus_{\rho \in \mathcal{D}(\mathcal{A})} \mathcal{H}_\rho$ be the direct sum of $\{\mathcal{H}_\rho\}$. Put $U(a) = \bigoplus_{\rho \in \mathcal{D}(\mathcal{A})} \pi_\rho(a)$. Then $U(a) \in B(K)$, and the mapping $a \longrightarrow U(a)$ is a *-representation of $\mathcal{A}$. This *-representation $\{U, K\}$ is called the universal *-representation of $\mathcal{A}$.

Theorem 2.1.1. *The universal *-representation $\{U, K\}$ of $\mathcal{A}$ is an isometric isomorphism. Therefore, every C^*-algebra is *-isomorphic to a uniformly closed self-adjoint subalgebra of $B(\mathcal{H})$ on some Hilbert space $\mathcal{H}$*

This theorem provides a link between C^*-algebras and linear operators on a Hilbert space $\mathcal{H}$. Moreover, consider finite dimensional C^*-algebras and the basis $\{e_i\}$ of $\mathcal{H}$, then for every element $A \in \mathcal{A}$ we can construct the following matrix,

$$\pi(A) = \{\pi(A)e_1|\pi(A)e_2|...|\pi(A)e_n\}. \tag{2.3}$$

As for states, let us introduce the following theorem

Theorem 2.1.2 (Frigyes Riesz Representation Theorem)**.** *Let $\mathcal{H}$ be a Hilbert space. Then the space $\mathcal{H}^*$ of linear continuous functionals on $\mathcal{H}$ can be identified with $\mathcal{H}$ itself as $T \in \mathcal{H}^*$ iff there exists $B \in \mathcal{H}$ such that*

$$T(A) = T_B(A) = \langle A, B \rangle, \quad x \in \mathcal{H}. \tag{2.4}$$

Moreover, $\|T_B\| = \|B\|$.

Therefore, through the same procedure, both elements of a finite dimensional C^*-algebra and its states can be identified with matrices in $\mathbb{C}^{n \times n}$. In what follows, we will only consider finite dimensional systems. Since we are interested in numerical methods, this is a natural assumption. At this point we are ready to shift the focus towards the necessary notions of QM.

2.2 Quantum mechanics

Following the description provided by [32] and [35], QM is built around observables. These are associated to Hermitian elements A of a C^*-algebra $\mathcal{A}$, called observable algebra. Since we are working with finite dimensional systems, they can be represented as Hermitian matrices of dimension $n \times n$. More specifically, the involution * is taken to be the conjugate transposition $\cdot^* = \cdot^\dagger = \bar{\cdot}^\top$, and every observable must satisfy $A^\dagger = A$.

A state in QM is a linear functional mapping observables onto their expectation values $\langle A \rangle$. In other words, a state is an element of the dual $\mathcal{A}^*$ of $\mathcal{A}$.

In order for QM to have a valid probabilistic interpretation, and accordance with the theory of C^*-algebras, the following assumptions are made,

- Normalisation: $\langle \mathbb{1} \rangle = 1$.
- Positiveness: $\forall A \in \mathcal{A} : \langle A^\dagger A \rangle \geq 0$.

Take the Hilbert-Schmidt scalar product $\langle B, A \rangle = tr(B^\dagger A)$. Under this representation, and considering theorem 2.1.2, it can be shown that states are associated to density matrices $\rho \in \mathcal{D}(\mathcal{H})$, which are Hermitian, have unit trace and are semi-positive definite. Then, the expectation value of and observable would be given by its scalar product with a density matrix, i.e. $\langle A \rangle = tr(\rho A)$.

An important property of $\mathcal{D}(\mathcal{H})$, the set of all density matrices is its convexity. In this way, any convex combination of density matrices is a density matrix as well. This leads to the classification of states into pure and mixed states. Pure states are the extreme points of the convex set, i.e. those that can be represented by a rank 1 density matrix, whereas mixed states can only be written as convex combinations of different density matrices.

Since pure states have rank 1, they can also be written in a simpler representation, that of complex unit vectors. These unit vectors are usually written using Dirac's bra-ket notation, where a pure state is written as a ket $|\psi\rangle$, and its adjoint as a bra $\langle\psi|$. Under this notation, the expectation value of an observable would be given by $\langle A \rangle = \langle\psi|A|\psi\rangle$. In this way we recover the "wave function" representation of QM.

In QM composite systems are joined by a tensor product. When a composite state cannot be factorised or written as a mixture of factorised states, it is said to be entangled. Entangled states are separate physical objects that behave as a single entity, even if its parts are at a distance.

Definition 2.2.1. The tensor product of two vector spaces X_1, X_2 is the vector space of elements $x_1 \otimes x_2$, where $\otimes : (x_1, x_2) \mapsto x_1 \otimes x_2$ is a bilinear map such that for every bilinear map $\psi(x_1, x_2) : X_1 \times X_2 \to X_3$, there is a unique map $\tilde{\psi}(x_1 \otimes x_2) : X_1 \otimes X_2 \to X_3$ such that $\tilde{\psi}(x_1 \otimes x_2) = \psi(x_1, x_2)$. In the case of matrix vector spaces, $x_1 \otimes x_2$ is computed as

$$x_1 \otimes x_2 = \begin{bmatrix} x_{1_{11}} x_2 & \cdots & x_{1_{1n}} x_2 \\ \vdots & \ddots & \vdots \\ x_{1_{m1}} x_2 & \cdots & x_{1_{mn}} x_2 \end{bmatrix}, \tag{2.5}$$

where x_1 is of dimension $m \times n$.

The partial trace $tr_B(\rho_{AB}) = \sum_i (\mathbb{1} \otimes \langle i|)\rho_{AB}(\mathbb{1} \otimes |i\rangle)$ is used to obtain the reduced state of a subsystem in a composite quantum system. This is analogous to marginalisation in probability theory.

The evolution of a quantum system can be regarded either as the evolution of the observables or the evolution of the states. The first case is known as the Heisenberg picture, in which $\mathcal{A} \ni A \to \mathcal{T}(A)$; whereas the latter is the Schrödinger picture, in which $\rho \to \mathcal{T}^*(\rho)$. These two interpretations are equivalent and can be mixed into an interaction picture. Therefore, $tr(\rho\mathcal{T}(A)) = tr(\mathcal{T}^*(\rho)A)$ arises as a condition for consistency, where $\mathcal{T}$ and $\mathcal{T}^*$ are mutually adjoint maps.

Any physically meaningful evolution $\mathcal{T} : \mathcal{B}(\mathcal{H}) \to \mathcal{B}(\mathcal{H}')$ in the Schrödinger picture should satisfy the following properties

- Linearity:
$$\forall A, B \in \mathcal{B}(\mathcal{H}), c \in \mathbb{C} : \mathcal{T}(cA + B) = c\mathcal{T}(A) + \mathcal{T}(B). \tag{2.6}$$
- Trace preservation:
$$\forall A \in \mathcal{B}(\mathcal{H}) : tr(\mathcal{T}(A)) = tr(A). \tag{2.7}$$
- Complete positivity:
$$\mathcal{T} \otimes \mathbb{1}_n(A^\dagger A) \geq 0 \forall A \in \mathcal{B}(\mathcal{H}) \otimes \mathcal{M}_n \forall n \in \mathbb{N}. \tag{2.8}$$

The equivalent map $\mathcal{T}^* : \mathcal{B}(\mathcal{H}') \to \mathcal{B}(\mathcal{H})$ in the Heisenberg picture shall also satisfy the conditions of linearity and complete positivity. Whereas the trace preservation translates to unitality

$$\mathcal{T}^*(\mathbb{1}) = \mathbb{1}. \tag{2.9}$$

A mapping which fulfills either of these two sets of three conditions, depending on the picture being used, is called a quantum channel, e.i. a quantum channel is a Completely Possitive Trace Preserving (CPTP) map. A commonly used representation for quantum channels is the Kraus representation, which expresses the completely positive linear map $\mathcal{T} \in \mathcal{B}(\mathcal{M}_d, \mathcal{M}_{d'})$ as the following superoperator sum

$$\mathcal{T}(A) = \sum_{j=1}^{r} K_j A K_j^\dagger. \tag{2.10}$$

In this way, trace preservation is guaranteed iff $\sum_j K_j^\dagger K_j = \mathbb{1}$ and unitality iff $\sum_j K_j K_j^\dagger = \mathbb{1}$. If this last pair of conditions is relaxed and one allows $\sum_j K_j^\dagger K_j \leq \mathbb{1}$ or $\sum_j K_j K_j^\dagger \leq \mathbb{1}$, $\mathcal{T}$ is called a quantum operation [34], and each of the operators K_j is known as a operation elements or Kraus operator. Finally, the notation

$$\mathcal{T} \sim \{K_j\} \tag{2.11}$$

is commonly used to denote the quantum operation $\mathcal{T}(A) = \sum_{j=1}^{r} K_j A K_j^\dagger$.

Finally, after a measurement is performed, the state suffers a collapse. That is to say, it becomes a superposition of the eigenvectors of the observable that are compatible with the result of the measurement. Consider the eigendecomposition of an observable A

$$A = \sum_i \lambda_i \Pi_i, \tag{2.12}$$

where each of the Π_i is a projector, i.e. $\Pi_i^\dagger = \Pi_i$ and $\Pi_i^2 = \Pi_i$, λ_i are the possible values that can be obtained as result of a measurement. Then, after the measurement is performed, where the value λ_i is obtained, the state of the system becomes

$$\rho \longrightarrow \frac{\Pi_i \rho \Pi_i}{\mathrm{tr}(\Pi_i \rho \Pi_i)}. \tag{2.13}$$

3 Quantum error correction

3.1 Quantum error correction

The effect of a quantum environment on a quantum system of interest, often referred to as quantum noise, can be modelled as a quantum CPTP map in Schroedinger's picture, i.e. it is represented as a quantum channel. The changes it causes on the state of the system is called a quantum error. QEC aims to revert these unwanted changes caused by the environment. However, unlike in classical information processing, redundancy cannot be used to identify the error to be corrected. This is due to the no-cloning theorem [36], A consequence of the fact that measurements destroy quantum information. Thus, recovery action after a measurement process would not be possible. Another difference to classical information, is the fact quantum errors are continuous and of an infinite variety, instead of just random bit flips [27].

For these reasons, one has to rely heavily on the knowledge of the dynamical properties of the system at hand and the noise that affects it. Indeed, without a proper model of the noise, it may not be possible to recover the state of the system consistently. However, in many applications the noise is of a restricted form. For instance, a common assumption that can be made for systems of qubits is that the noise is independent for each qubit [14]. In other words, that each of the operation elements K_j can be factorised in a tensor product of one-qubit operators. Indeed, this is the kind of systems on which a substantial part of the present QEC theory is focused, and it does not directly lend itself to generalisation to physical systems that are not canonically decomposable into qubits or that are subject to non-independent (correlated) noises [15].

As an alternative to redundancy, the information of interest can be spread over a bigger Hilbert space through an adequate encoding. That is to say, the bigger Hilbert space is divided into equivalent subspaces, such that the noise can only map the state into its equivalent in one of the other subspaces, leaving the internal structure of the subspaces unaltered. This idea is the basis of what is known as subspace codes, the most common kind of QEC strategy. More precisely, a quantum code $\mathcal{C}$ is defined as a subspace of the Hilbert space $\mathcal{H}$ associated to a quantum system. The notation (n, d) is used to indicate the dimension of the full Hilbert space (n) and that of the code (d). With some abuse of notation, we may also call code the set of states with support on $\mathcal{C}$. Finally, a quantum error-correcting code is a pair $(\mathcal{C}, \mathcal{R})$ consisting of a subspace code $\mathcal{C}$ and a recovery map $\mathcal{R}$.

The following theorems give an insight at the conditions for correctability of a quantum code. These theorems have been derived in the original perspective of a set of errors. Later we focus on specific noise models.

Theorem 3.1.1. *Let $\mathcal{N}(\mathcal{C}, \mathcal{R})$ be the set of Kraus operators of all the noise channels that can be corrected by the quantum code $(\mathcal{C}, \mathcal{R})$. The Kraus operator N_a is in $\mathcal{N}(\mathcal{C}, \mathcal{R})$ iff when restricted to $\mathcal{C}$, $R_r N_a = \lambda_{ra}$ for each $R_r \in \mathcal{R}$. The family $\mathcal{N}(\mathcal{C}, \mathcal{R})$ is linearly closed and $(\mathcal{C}, \mathcal{R})$ is $\mathcal{N}(\mathcal{C}, \mathcal{R})$ correcting.*

Theorem 3.1.2. *The code $\mathcal{C}$ can be extended to an $\mathcal{N}$-correcting code iff for all basis elements $|i_L\rangle$, $|j_L\rangle$ $(i \neq j)$ of $\mathcal{C}$ and operators N_a, N_b in $\mathcal{N}$,*

$$\langle i_L | N_a^\dagger N_b | i_L \rangle = \langle j_L | N_a^\dagger N_b | j_L \rangle, \tag{3.1}$$

and

$$\langle i_L | N_a^\dagger N_b | j_L \rangle = 0. \tag{3.2}$$

These two conditions can be condensed into a single one as

$$\Pi N_a^\dagger N_b \Pi = \alpha_{ij} \Pi, \tag{3.3}$$

where Π is the projector onto the quantum code.

Theorem 3.1.3. *The recovery $\mathcal{R}$ has error 0 on $\mathcal{C}$, after the noise channel $\mathcal{N}$, iff $\mathbb{1} \otimes (\mathcal{R} \circ \mathcal{N}) \sum_i |i_L\rangle \otimes |i_L\rangle = \lambda \sum_i |i_L\rangle \otimes |i_L\rangle$.*

Given a code that satisfies theorem 3.1.2, let

$$\mathcal{V}^i = \mathrm{span}(\{N_a|i_L\rangle\}_a). \tag{3.4}$$

Due to condition (3.2) the $\mathcal{V}^i$ are orthogonal subspaces. Let also $|v_r^i\rangle$ be an orthonormal basis for ^{i}. Since the subspaces are orthonormal, we have that $\langle v_r^i|v_r^j\rangle = 0,\ \forall i \neq j$. Thus, there exists a unitary V_r such that $V_r|v_r^i\rangle = |i_L\rangle$. This leads to the recovery operation being a superoperator with the following elements $\{\mathcal{O}, R_1, ..., R_r, ...\}$, where $\mathcal{O}$ is the projector onto the orthogonal complement of $\bigoplus_i \mathcal{V}^i$, and

$$R_r = V_r \sum_r |v_r^i\rangle\langle v_r^i|. \tag{3.5}$$

In case that theorem 3.1.2 is not satisfied, the construction of a perfect recovery is not possible. However, a optimal recovery operation still exists and is given by what is known as Petz recovery map.

The Petz recovery map $\mathcal{R}_{\mathcal{N},\rho}$ for a quantum channel $\mathcal{N}$ and initial state ρ, as described in [28, 1], is given by the following expression,

$$\mathcal{R}_{\mathcal{N},\rho} \sim \{\rho^{1/2} N_i^\dagger \mathcal{N}(\rho)^{-1/2}\}. \tag{3.6}$$

In this work, we intend to adapt this recovery map to quantum codes and use it as recovery operation. Tho this end, let us show that $\mathcal{R}_{\mathcal{N},\Pi}$ is a perfect recovery for a correctable code with projector Π. First of all, let us substitute ρ in the previous expression by the projector onto the code Π, i.e.

$$\mathcal{R}_{\mathcal{N},\Pi} \sim \{\Pi^{1/2} N_i^\dagger \mathcal{N}(\Pi)^{-1/2}\}. \tag{3.7}$$

If $\mathcal{N}(\Pi)$ is full rank and $\mathcal{N}$ is trace-preserving, it is direct to show that this is a time-reversal of the quantum operation A, since $\sum_i N_i^\dagger N_i = \mathbb{1}$. If $\mathcal{N}(\Pi)$ is not full rank, consider the orthonormal basis $\{|\alpha_n\rangle\}_n$ of its kernel,

$$\sum_i \langle \alpha_n | N_i \Pi N_i^\dagger | \alpha_n \rangle = 0. \tag{3.8}$$

This implies that $N_i^\dagger|\alpha_n\rangle \in \ker(\Pi)$. Therefore, one could write

$$\begin{aligned}\mathbb{1} &= \sum_i N_i^\dagger N_i \\ &= \sum_i N_i^\dagger (\Pi_{\mathcal{N}(\Pi)} + \Pi_{\mathcal{N}(\Pi)}^\perp) N_i \\ &= \sum_i N_i^\dagger \mathcal{N}(\Pi)^{-1/2} \mathcal{N}(\Pi) \mathcal{N}(\Pi)^{-1/2} N_i + \sum_{in} N_i^\dagger |\alpha_n\rangle\langle\alpha_n| N_i\end{aligned} \quad . \tag{3.9}$$

Then, multiply on both sides by Π, one obtains

$$\Pi = \sum_i \Pi N_i^\dagger \mathcal{N}(\Pi)^{-1/2} \mathcal{N}(\Pi) \mathcal{N}(\Pi)^{-1/2} N_i \Pi. \tag{3.10}$$

This shows that the code is invariant under concatenation of the noise with the recovery $\mathcal{R} \circ \mathcal{N}(\Pi) = \Pi$. Let us now show that the application of this recovery corresponds to the time reversal of the states belonging to the code, i.e. the states such that $\Pi\rho\Pi = \rho$. In what follows, with some abuse of notation, we use $\Pi\rho\Pi$ to implicitly state that $\rho \in \mathcal{C}$.

Definition 3.1.1. For any ρ, the time reversal map $\mathcal{T}_\rho : \mathcal{N} \mapsto \mathcal{T}_\rho(\mathcal{N})$ is a map such that

$$\mathcal{T}_\rho(\mathcal{N}) \sim \{\rho^{1/2} N_i^\dagger (\mathcal{N}(\rho))^{-1/2}\}. \tag{3.11}$$

Computing explicitly $\mathcal{T}_{\mathcal{N}(\Pi)}(\mathcal{R}_{\mathcal{N},\Pi})$, we have that its Kraus operators are

$$\mathcal{N}(\Pi)^{1/2} R_i^\dagger \Pi = \mathcal{N}(\Pi)^{1/2} (\mathcal{N}(\Pi)^{-1/2} N_i \Pi) \Pi = \Pi_{\mathcal{N}(\Pi)} N_i \Pi. \tag{3.12}$$

Then, for $\Pi\rho\Pi$ one has that

$$\begin{aligned}\mathcal{T}_{\mathcal{N}(\Pi)}(\mathcal{R}_{\mathcal{N},\Pi})(\Pi\rho\Pi) &= \sum_i \Pi_{\mathcal{N}(\Pi)} N_i \Pi\Pi\rho\Pi\Pi N_i^\dagger \Pi_{\mathcal{N}(\Pi)} \\ &= \sum_i \Pi_{\mathcal{N}(\Pi)} N_i \Pi\rho\Pi N_i^\dagger \Pi_{\mathcal{N}(\Pi)}\end{aligned} \quad . \tag{3.13}$$

However, as seen above $\Pi N_i^\dagger \Pi_{\mathcal{N}(\Pi)}^\perp = 0$, therefore

$$\Pi N_i^\dagger = \Pi N_i^\dagger (\Pi_{\mathcal{N}(\Pi)} + \Pi_{\mathcal{N}(\Pi)}^\perp) = \Pi N_i^\dagger \Pi_{\mathcal{N}(\Pi)}, \tag{3.14}$$

$$\mathcal{T}_{\mathcal{N}(\Pi)}(\mathcal{R}_{\mathcal{N},\Pi})(\Pi\rho\Pi) = \sum_i N_i \Pi\rho\Pi N_i^\dagger = \mathcal{N}(\Pi\rho\Pi). \tag{3.15}$$

Different implementations of this map have been proposed. A discrete time implementation is proposed in [9], and a continuous time implementation is proposed in [20].

3.2 Fidelity measures

In order to measure how good a QEC code is, the fidelity with respect to the initial state can be used. The fidelity is a measure of similarity between two different states and has the following definition,

$$F(\rho_1, \rho_2) := \max |\langle\psi_1|\psi_2\rangle|^2, \tag{3.16}$$

where the maximum is taken over all purifications $|\psi_1\rangle$ and $|\psi_2\rangle$ of ρ_1 and ρ_2, respectively. A purification is a higher-dimensional pure state $|\psi\rangle$ that has the original state ρ as reduction, i.e. $\rho = Tr_A(|\psi\rangle\langle\psi|)$. If one of the two states in the fidelity calculation is already a pure state, the expression reduces to

$$F(|\psi_1\rangle, \rho_2) = \langle\psi_1|\rho_2|\psi_1\rangle. \tag{3.17}$$

Therefore, in the context of QEC, $F(\rho, \mathcal{R}\circ\mathcal{N}(\rho))$ would be a quantity that one would want to maximise. However, it has been argued that the entanglement fidelity is a better measure, rather than the fidelity [26]. The entanglement fidelity measures not only how well a state is preserved after a quantum operation, but also of how well its possible entanglement with the rest of a larger system is preserved. This latter property is the main reason given as to why it may be a better choice than $F(\rho, \mathcal{R}\circ\mathcal{N}(\rho))$. The entanglement fidelity has the following definition,

$$F_e(\rho, \mathcal{N}) := \langle\psi|(\mathcal{N}\otimes\mathbb{1})(|\psi\rangle\langle\psi|)|\psi\rangle, \tag{3.18}$$

where $|\psi\rangle$ is again a purification of ρ, and $(\mathcal{N}\otimes\mathbb{1})$ is the natural extension of the original quantum operation to the space on which ρ has been purified.

If the quantum operation $\mathcal{N}$ is given in the Kraus representation, then the entanglement fidelity has the following expression,

$$F_e(\rho, \mathcal{N}) = \sum_i tr(N_i \rho) tr(N_i^\dagger \rho). \tag{3.19}$$

A remarkable property of the entanglement fidelity is the following,

$$F_e(\rho, \mathcal{N}) \leq F(\rho, \mathcal{N}(\rho)). \tag{3.20}$$

Meaning that when the entanglement fidelity is maximised, one is also maximising a lower bound of the fidelity. Additionally, for pure states one has that

$$F_e(|\psi\rangle\langle\psi|, \mathcal{N}) = F(|\psi\rangle\langle\psi|, \mathcal{N}(|\psi\rangle\langle\psi|)). \tag{3.21}$$

Which is to be expected, since pure states cannot be the reduced state of a larger entangled state, and therefore there is no entanglement to be preserved. Finally, it has been proved in [14] that if the fidelity is kept high for all pure states, $F(|\psi\rangle\langle\psi|, \mathcal{N}(|\psi\rangle\langle\psi|)) \geq 1 - \epsilon \; \forall |\psi\rangle$, the entangled fidelity is also kept high for all states, $F_e(\rho, \mathcal{N}) \geq 1 - 3\epsilon/2 \; \forall \rho$.

However, these measures are dependent on the input state, and therefore not ideal to benchmark QEC operations in general. To overcome this limitation, the minimal or the average fidelity may be used. In fact, this question is explored in [8] with the aim of comparing different quantum processes in a more general way. In the context of QEC, it suffices to compare the noise and recovery operation with the identity operation. Thus, the definition of the average and minimal fidelity are the following, respectively

$$F_{avg}(\mathcal{N}, \mathcal{E}) := \int_\rho d\rho F(\mathcal{N}(\rho), \mathcal{E}(\rho)), \tag{3.22}$$

$$F_{min}(\mathcal{N}, \mathcal{E}) := \min_\rho F(\mathcal{N}(\rho), \mathcal{E}(\rho). \tag{3.23}$$

Then, if $\mathcal{E} = \mathbb{1}$, which is the case of interest for QEC, and $\mathcal{N}$ is given in the Kraus representation, one has

$$F_{avg}(\mathcal{N}, \mathbb{1}) := \frac{1}{n^2} \sum_k |tr(N_k)|, \tag{3.24}$$

$$F_{min}(\mathcal{N}, \mathbb{1}) := \min_\rho \sum_k |tr(N_k \rho)|^2. \tag{3.25}$$

3.3 Quantum error correction via optimisation

It is known that finding perfectly correctable codes, even if the system dynamics are perfectly known, is an NP-Hard problem [4]. Nonetheless, a different approach can be taken. One could instead try searching for approximate codes using optimisation algorithms. In fact, work has already been done in this direction and it may lead to better codes when the implementation details, such as the required amount of qubits, are taken into account [22].

In [29], it has been proposed an adapted version of the power method eigenvalue algorithm to maximise what they call channel fidelity, which is a special case of the entanglement fidelity with the maximally entangled state $|\Omega\rangle = \sum_k |kk\rangle/\sqrt{d}$ as initial state. Then, a line of work focused on the use of Semidefinite Programming (SDP) to optimise the recovery operation can be traced back to [37], in which they assume fixed encodings and aim to maximise the minimum fidelity. A similar scheme is proposed in [7], where they use SDP to find the optimum recovery operation, given a certain quantum code. Their objective function is the entanglement fidelity between the initial state and the recovered state after the noise. This work is then complemented by [6], in which they focus on the code, rather than the recovery.

In [19] they account for uncertainties in the model of the noise. They solve the optimisation problem by convexifying the constraint $C^\dagger C = \mathbb{1}_d$ and solve the problem with $C^\dagger C \leq \mathbb{1}_d$ instead. In this way, they are able to use SDP find codes that maximise the average fidelity. Then, they take replace the solution of the relaxed problem with its singular values, thus recovering the orthonormality constraint. Then, for the recovery, they use an approximation to the optimal recovery that can be calculated directly, thus skipping this part of the optimisation problem. Then, in [18] they include a step to optimise the recovery, using SDP as well. The solution to both problems is computed using Lagrange duality. Finally, in [17] a quantum process tomography (QPT) is added to eliminate the need for a previously known model of the noise that affects the system.

Another work that tries to eliminate the need for a known noise model is [25], where a reinforcement learning scheme is proposed to find optimum codes. The noise is then assumed to be a black box, together with the recovery operation. Finally, in [42] instead of relying on classical optimisation algorithms, they use a variational quantum algorithm to find the optimal code and recovery. However, the proposed algorithm requires the specification an initial state, like in [7].

On the other hand, this algorithm has the advantage that it would directly provide an implementation of these operations, since it can be run on the quantum hardware itself.

3.4 Our framework for code optimisation

In this section we derive intuitively the cost function and the specific problem we shall solve numerically. The problem derived in this section has the advantage that it is independent of the initial state of the system, since it is based only to the subspace code and the noise model.

Let $\mathcal{N}$ be a known noise channel defined as

$$\mathcal{N}(\rho) := \sum_{k=1}^{m} N_k \rho N_k^\dagger, \tag{3.26}$$

where $\sum_{k=1}^{m} N_k^\dagger N_k = \mathbb{1}_n$.

Consider the Petz recovery map for the code identified by Π

$$\mathcal{R}_{\mathcal{N},\Pi}(\rho) = \sum_{l=1}^{m} R_l \rho R_l^\dagger, \tag{3.27}$$

for $\rho \in \mathcal{D}(\mathcal{H})$. Recall that the Petz recovery map is a perfect recovery if Π corrects $\mathcal{N}$. However, we do not know the best Π.

Therefore, the problem is how to find an appropriate projection matrix Π, such that $\mathcal{R}_{\mathcal{N},\Pi}$ acts as the reversal of $\mathcal{N}$. Precisely, what we want is that the composition of the noise and the recovery $\mathcal{A} = \mathcal{R}_\Pi \circ \mathcal{N}(\rho)$ is as close to the identity map as possible, when restricted to the subspace of the code.

$$\mathcal{A}_\Pi(\rho) = \sum_{j,k=1}^{m} \underbrace{\Pi N_k^\dagger \mathcal{N}(\Pi)^{-1/2} N_j}_{A_{jk}} \rho N_j^\dagger \mathcal{N}(\Pi)^{-1/2} N_k \Pi. \tag{3.28}$$

In order to restrict the map to the code, consider only input states such that $\rho = \Pi\rho\Pi$. It is convenient to introduce the following map,

$$\check{\mathcal{A}}_\Pi = \sum_{j,k=1}^{m} \underbrace{\Pi N_k^\dagger \mathcal{N}(\Pi)^{-1/2} N_j \Pi}_{\check{A}_{jk}} \rho \Pi N_j^\dagger \mathcal{N}(\Pi)^{-1/2} N_k \Pi. \tag{3.29}$$

Notice that $\check{\mathcal{A}}$ being the composition of $\Pi \cdot \Pi$ with $\mathcal{A}_\Pi$, it is a trace non-increasing map.

Proposition 3.4.1. *Conjugate transposition of the operation elements $\check{A}_{jk}$ corresponds to a swapping of its indexes, which then implies that the map $\check{\mathcal{A}}_\Pi$ is self-adjoint.*

$$\check{A}_{jk}^\dagger = \check{A}_{kj} \Rightarrow \check{\mathcal{A}}_\Pi^\dagger = \check{\mathcal{A}}_\Pi. \tag{3.30}$$

Proof. It can be directly seen that $\check{A}_{jk}^\dagger = \check{A}_{kj}$. Then, the operator sum can be written as

$$\check{\mathcal{A}}_\Pi[\rho] = \sum_{jk} \check{A}_{jk}\rho\check{A}_{jk}^\dagger = \sum_{j\leq k}(\check{A}_{jk}\rho\check{A}_{jk}^\dagger + \check{A}_{kj}\rho\check{A}_{kj}^\dagger) = \sum_{j\leq k}(\check{A}_{jk}\rho\check{A}_{jk}^\dagger + \check{A}_{jk}^\dagger\rho\check{A}_{jk}). \tag{3.31}$$

□

Next, in order to better exploit the linear structure of the maps, we consider the vectorised representation.

Definition 3.4.1. Given a matrix

$$X = \begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{1m} & \dots & x_{mn} \end{bmatrix},$$

its vectorisation is defined as the vertical stacking of its columns, i.e.

$$\text{vec}(X) = \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{1m} \\ \vdots \\ x_{mn} \end{bmatrix}. \tag{3.32}$$

The vectorisation $\text{vec}(AXB)$ satisfies the property $\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X)$. Then, $\check{\mathcal{A}}_\Pi$ can be rewritten as

$$\mathrm{vec}(\check{\mathcal{A}}_\Pi(\rho)) = \mathrm{vec}(\sum_{jk} \check{A}_{jk}\rho\check{A}_{jk}^\dagger) = \underbrace{\sum_{jk} \overline{\check{A}_{jk}} \otimes \check{A}_{jk}}_{\hat{A}_\Pi} \mathrm{vec}(\rho), \tag{3.33}$$

where

$$\hat{A}_\Pi = \sum_{j,k=1}^{m} \Pi^\top N_j^\top \mathcal{N}(\Pi)^{-\top/2}\overline{N_k}\Pi^\top \otimes \Pi N_k^\dagger \mathcal{N}(\Pi)^{-1/2} N_j \Pi. \tag{3.34}$$

Proposition 3.4.2. *The operator $\hat{A}_\Pi$ is Hermitian*

$$\hat{A}_\Pi = \sum_{jk} \overline{\check{A}_{jk}} \otimes \check{A}_{jk}, \tag{3.35}$$

$$\hat{A}_\Pi^\dagger = \hat{A}_\Pi. \tag{3.36}$$

Proof. Using the fact that $\check{A}_{jk}^\dagger = \check{A}_{kj}$, one can write

$$\begin{aligned}\hat{A} = \sum_{j\leq k}(\overline{\check{A}_{jk}} \otimes \check{A}_{jk} + \overline{\check{A}_{kj}}\rho\check{A}_{kj}) &= \sum_{j\leq k}(\overline{\check{A}_{jk}} \otimes \check{A}_{jk} + (\overline{\check{A}_{jk}}\rho\check{A}_{jk})^\dagger) \\ &= 2\sum_{j\leq k}\mathrm{Sym}(\overline{\check{A}_{jk}} \otimes \check{A}_{jk})\end{aligned}. \tag{3.37}$$

□

Let us now define the set of vectorised code words as v_Π and study the structure that $\hat{A}_\Pi$ would have in this space.

Definition 3.4.2. The set of vectorised code words is v_Π defined as the set of the vectorised representations of the states in the support of the projector to the code Π, i.e.

$$v_\Pi := \{\mathrm{vec}(\rho) | \Pi\rho\Pi = \rho\}. \tag{3.38}$$

Proposition 3.4.3. *Considering the set of code words v_Π and decomposing the underlying space of the system as $\mathbb{C}^{n\times n} = v_\Pi \oplus v_\Pi^\perp$, there exists a unitary operator T such that for every state $\rho \in \mathcal{C}$,*

$$T\rho T^\dagger = \left[\begin{array}{c|c} \rho_\Pi & 0 \\ \hline 0 & 0 \end{array}\right]. \tag{3.39}$$

Under the same operator T, after vectorisation the operator $\hat{A}_\Pi$ would be

$$(\overline{T}\otimes T)\hat{A}_\Pi(\overline{T}\otimes T)^\dagger = \left[\begin{array}{c|c} \tilde{A}_\Pi & 0 \\ \hline 0 & 0 \end{array}\right], \tag{3.40}$$

where $\tilde{A}_P i$ is Hermitian

$$\tilde{A}_\Pi = \tilde{A}_\Pi^\dagger. \tag{3.41}$$

Proof. Since $\Pi = \Pi^\dagger$, it is unitarily diagonalisable, i.e. there exists some T, such that $TT^\dagger = \mathbb{1}$ and

$$T\Pi T^\dagger = \left[\begin{array}{c|c} \mathbb{1} & 0 \\ \hline 0 & 0 \end{array}\right]. \tag{3.42}$$

Under this transformation and since $\Pi\rho\Pi$ is in the support of Π, we have that

$$T\Pi\rho\Pi T^\dagger = T\Pi T^\dagger(T\rho T^\dagger)T\Pi T^\dagger = \left[\begin{array}{c|c} \rho_\Pi & 0 \\ \hline 0 & 0 \end{array}\right]. \tag{3.43}$$

Similarly, for the operator $\hat{A}_\Pi$ and since the tensor product preserves the block structure of the left term, we have that

$$\begin{aligned}(\overline{T}\otimes T)\hat{A}_\Pi(\overline{T}^\dagger\otimes T^\dagger) &= \sum_{jk}\overline{(T\check{A}_{jk}T^\dagger)}\otimes(T\check{A}_{jk}T^\dagger)\\ &= \sum_{jk}\overline{(T\Pi N_k^\dagger\mathcal{N}^{-1/2}(\Pi)N_j\Pi T^\dagger)}\otimes(T\Pi N_k^\dagger\mathcal{N}^{-1/2}(\Pi)N_j\Pi T^\dagger)\\ &= \left[\begin{array}{c|c} \overline{\check{A}_\Pi} & 0 \\ \hline 0 & 0 \end{array}\right]\otimes\left[\begin{array}{c|c} \check{A}_\Pi & 0 \\ \hline 0 & 0 \end{array}\right] \\ &= \left[\begin{array}{c|c} \tilde{A}_\Pi & 0 \\ \hline 0 & 0 \end{array}\right]\end{aligned} \quad . \tag{3.44}$$

Finally, Hermitianity of $\tilde{A}_\Pi$ is a direct consequence of the Hermitianity of $\hat{A}_\Pi$ and the block structure of $(\overline{T} \otimes T)\hat{A}_\Pi(\overline{T} \otimes T)^\dagger$. □

Theorem 3.4.4. *Let* $d = \mathrm{rank}(\Pi)$, *then*

1. *The product of* $\tilde{A}_\Pi$ *by its conjugate transpose is bounded by the identity,* $\tilde{A}_\Pi^\dagger \tilde{A}_\Pi \leq \mathbb{1}_{d^2}$.
2. $\mathcal{C} \sim \Pi$ *is a correctable code iff* $\tilde{A}_\Pi = \mathbb{1}_{d^2}$.

Proof. From the trace non-increase condition one has that

$$\sum_{jk} \check{A}_{jk}^\dagger \check{A}_{jk} \leq \mathbb{1}. \tag{3.45}$$

Each of the terms $\check{A}_{jk}^\dagger \check{A}_{jk}$ is positive semi-definite. Therefore, all of their eigenvalues are non-negative. At the same time, to comply with the previous condition, it must also be the case that

$$\langle\psi|(\sum_{jk} \check{A}_{jk}^\dagger \check{A}_{jk})|\psi\rangle \leq 1. \tag{3.46}$$

Putting this together with the non-negativity of the eigenvalues, one can conclude that

$$0 \leq \langle\psi|(\check{A}_{jk}^\dagger \check{A}_{jk})|\psi\rangle \leq 1, \tag{3.47}$$

and putting the last two expressions together, one has that

$$0 \leq \langle\psi|(\sum_{jk} \check{A}_{jk}^\dagger \check{A}_{jk})|\psi\rangle \leq 1. \tag{3.48}$$

Since this is true $\forall|\psi\rangle$, vectorising and using the same transformation T from the previous proposition, it is easy to see that

$$\tilde{A}^\dagger \tilde{A} \leq \mathbb{1}_{d^2}. \tag{3.49}$$

To prove the second part, it is easy to see that $\mathrm{rk}(\hat{A}_\Pi) \leq d^2$ due to the projector Π, which has $\mathrm{rk}(\Pi) = d$ and the fact that $\mathrm{rk}(A \otimes B) = \mathrm{rk}(A)\mathrm{rk}(B)$. This means that at least $n^2 - d^2$ of the eigenvalues of $\hat{A}_\Pi$ are zero.

For the study of the rest of the eigenvalues, consider $\{|\psi_k\rangle\}$ such that $|\psi_k\rangle\langle\psi_k| \in \mathcal{C}$. Then $\overline{|\psi_a\rangle} \otimes |\psi_b\rangle \notin \ker(\overline{\Pi} \otimes \Pi)$ and $\mathrm{span}(\{\overline{|\psi_a\rangle} \otimes |\psi_b\rangle\}) \supseteq \mathrm{Im}(\hat{A}_\Pi)$.

If the code is correctable, one has that $\check{\mathcal{A}}(|\psi_a\rangle\langle\psi_a|) = |\psi_a\rangle\langle\psi_a|$,

$$\Rightarrow \overline{|\psi_a\rangle} \otimes |\psi_a\rangle = (\sum_{jk} \overline{\check{A}_{jk}} \otimes \check{A}_{jk})\overline{|\psi_a\rangle} \otimes |\psi_a\rangle. \tag{3.50}$$

Therefore, there are d eigenstates with eigenvalue 1.

Now consider the following two states

$$\rho = (|\alpha|^2|\psi_a\rangle\langle\psi_a| + |\beta|^2|\psi_b\rangle\langle\psi_b|),$$
$$\sigma = (\alpha|\psi_a\rangle + \beta|\psi_b\rangle)(\overline{\alpha}\langle\psi_a| + \overline{\beta}\langle\psi_b|),$$

with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$. It is easy to see that $\sigma = \rho + (\alpha\overline{\beta}|\psi_a\rangle\langle\psi_b| + \overline{\alpha}\beta|\psi_b\rangle\langle\psi_a|)$.

By linearity, one gets that $\check{\mathcal{A}}(\rho) = \rho$ and thus $\hat{A}_\Pi \mathrm{vec}(\rho) = \mathrm{vec}(\rho)$.

Then $\check{\mathcal{A}}(\sigma) = \rho + \check{\mathcal{A}}(\alpha\overline{\beta}|\psi_a\rangle\langle\psi_b| + \overline{\alpha}\beta|\psi_b\rangle\langle\psi_a|)$ and $\hat{A}_\Pi \mathrm{vec}(\sigma) = \mathrm{vec}(\rho) + \hat{A}_\Pi \mathrm{vec}(\alpha\overline{\beta}|\psi_a\rangle\langle\psi_b| + \overline{\alpha}\beta|\psi_b\rangle\langle\psi_a|) = \mathrm{vec}(\rho) + \hat{A}_\Pi(\alpha\overline{\beta}\overline{|\psi_b\rangle} \otimes |\psi_a\rangle + \overline{\alpha}\beta\overline{|\psi_a\rangle} \otimes |\psi_b\rangle)$.

Hence, for $\mathcal{C}$ to be a correctable code, one would also need that $\hat{A}_\Pi\overline{|\psi_b\rangle} \otimes |\psi_a\rangle = \overline{|\psi_b\rangle} \otimes |\psi_a\rangle$, Since it must be true for any a, b, it results in $d^2 - d$ eigenstates having eigenvalue equal to 1.

Putting everything together, one sees that $\tilde{A}$ has all of its eigenvalues equal to 1, and being Hermitian, this also implies that it is equal to the identity. Proving the other direction of the implication is automatic, since if $\tilde{A} = \mathbb{1}$, for any state in the code one would have that $\mathcal{A}(\rho) = \rho$, due to the previous proposition. □

In the light of this theorem, it is natural to consider the following as a quality index for the correctability of Π,

$$J(\Pi) = tr(\tilde{A}_\Pi) = \sum_k \mathrm{eigs}(\hat{A}_k). \tag{3.51}$$

The following proposition clarifies the optimality of a code, as measured by $J(\Pi)$.

Proposition 3.4.5. *The cost function $J(\Pi)$ is non-negative and upper-bounded by the square of the rank of the projector Π,*

$$0 \le J(\Pi) \le d^2. \tag{3.52}$$

Furthermore, the code $\mathcal{C} \sim \Pi$ is a correctable code iff $J(\Pi) = d^2$.

Proof. The $J(\Pi) \leq d^2$ part is a direct consequence of the first item of the previous theorem, together with the fact that $\tilde{A}$ is Hermitian. The following proves the $J(\Pi) \geq 0$ part,

$$\begin{aligned} tr(N_\Pi) &= \sum_{kl} tr(\Pi^\top N_k^\top \mathcal{N}(\Pi)^{-\top/2}\overline{N_l}) tr(\Pi N_l^\dagger \mathcal{N}(\Pi)^{-1/2} N_k) \\ &= \sum_{kl} tr(\Pi N_l^\dagger \mathcal{N}(\Pi)^{-1/2} N_k \Pi)^2 = tr((\sum_l N_l \Pi)^\dagger \mathcal{N}(\Pi)^{-1/2} \underbrace{(\sum_k N_k \Pi)}_{O_\Pi})^2 . \\ &= tr(O_\Pi O_\Pi^\dagger \mathcal{N}(\Pi)^{-1/2}) \geq 0 \end{aligned} \tag{3.53}$$

Finally, using the results of the previous theorem, $J(\Pi) = d^2$ iff all the eigenvalues of $\tilde{A}$ are equal to 1. □

If we were to fix the rank of Π and consider it as a parameter of the cost function, i.e. $J_d(\Pi)$, the following relation to the average fidelity is found.

Proposition 3.4.6. *The cost function $J_d(\Pi)$ with d fixed is proportional to the average fidelity of the operation $\tilde{\mathcal{N}} \sim \{\tilde{A}\}$,*

$$J_d(\Pi) = d^2 F_{avg}(\mathcal{N}, \mathbb{1}). \tag{3.54}$$

Proof. By direct calculation,

$$J_d(\Pi) = tr(\tilde{A}) = |tr(\tilde{A})| = d^2 |\frac{1}{d^2} tr(\tilde{A})| = d^2 F_{avg}(\mathcal{N}, \mathbb{1}). \tag{3.55}$$

□

If a correctable code for a certain error is unknown, consider the following optimisation problem with fixed $d = rank(\Pi)$,

$$\Pi_C = \underset{\Pi}{\operatorname{argmax}} J(\Pi). \tag{3.56}$$

Since d is fixed, one could also decompose the projector onto the code as $\Pi = UU^\dagger$, where $U \in \mathbb{C}^{n \times d}$ and $U^\dagger U = \mathbb{1}_d$. This decomposition leads one to perform optimisation on the complex valued Stiefel manifold $V_d(\mathbb{C}^n)$, since $U \in V_d(\mathbb{C}^n)$, and the optimisation problem would be the following,

$$\Pi_U = \underset{U \in V_d(\mathbb{C}^n)}{\mathrm{argmax}} J(\Pi). \tag{3.57}$$

In the following chapter the Stiefel manifold $V_d(\mathbb{C}^n)$ will be defined and methods to solve this optimisation problem will be introduced.

4 Riemannian optimisation

Since the optimisation parameter is a complex matrix $U \in \mathbb{C}^{n\times d}$ with the constrained $U^\dagger U = \mathbb{1}_d$, commonly used steepest descent or Newton method should be modified. Let us introduce some basics of the complex Stiefel manifold $V_d(\mathbb{C}^n)$.

Relying on geometric and probabilistic intuition a numerical approach may be pursued: a subspace can be associated to a projector, which in turn can be expressed in terms of an orthonormal k-frame. Hence, the set of subspace codes is equal to the complex-valued Stiefel manifold. This observation opens the door to the use of optimisation algorithms for the search of error correcting codes. The Stiefel manifold is a Riemannian manifold, therefore, instead of performing constraint Eucledian optimisation on $\mathbb{C}^{n\times d}$, one can perform Riemannian optimisation without constraints to solve this problem.

This chapter starts with an introduction of the necessary concepts to understand the theory behind Riemannian optimisation. Then the focus shifts towards Stiefel manifolds, which can model quantum codes. Finally, a series of local and global optimisation algorithms on the Stiefel manifold are introduced.

4.1 Manifold prerequisites

Manifolds are a generalisation of surfaces on Eucledian space. As such, they are built on top of topological spaces, which are a more general kind of structure than vector spaces.

Definition 4.1.1. Given a set X, a collection τ of open subsets of X is called a topology on X if the following conditions are satisfied:

- The empty set and X itself belong to τ, i.e. $, X \in \tau$.
- τ is closed w.r.t. finite and infinite unions, i.e. given $\tau_i \in \tau$, $\bigcup\limits_{i=1}^{N} \tau_i \in \tau$, $\bigcup\limits_{i=1}^{\infty} \tau_i \in \tau$.
- τ is closed w.r.t. finite intersections, i.e. given $\tau_i \in \tau$, $\bigcap\limits_{i=1}^{N} \tau_i \in \tau$.

Additionally, X is called a topological space and its elements are called points.

However, some additional structure is necessary. The following definitions give the necessary notions of separability and distinguishability, as well as the relation to Eucledian space.

Definition 4.1.2. A neighbourhood W of a point U in a topological space X is a subset V of X that includes an open set containing U, i.e. $U \in V \subseteq W \subseteq X$.

Definition 4.1.3. A second countable space X is a topological space whose topology has a countable base, i.e. there exists some collection $\mathcal{V} = \{V_i\}_{i=1}^{\infty}$ of open subsets of X such that any open subset of X can be written as the union of elements of $\mathcal{V}$.

Definition 4.1.4. A Hausdorff space X is a topological space where for every pair points there exist neighbourhoods of each that are mutually disjoint, i.e. given points $U_1, U_2 \in X$, $\exists W_1, W_2$ such that $U_1 \in W_1$, $U_2 \in W_2$ and $W_1 \cap W_2 = \emptyset$

Definition 4.1.5. A map $\Phi : \mathcal{X}_1 \longrightarrow \mathcal{X}_2$ is called a homeomorphism if it satisfies:

- Bijectivity, i.e. $\forall U \in X_1 \exists \Phi(U) \in X_2$ and $\forall U_a, U_b \in X_1$, $\Phi(U_a) = \Phi(U_b)$ iff $U_a = U_b$.
- Continuity, i.e. for any neighbourhood W_2 of $\Phi(U)$ there exists some neighbourhood W_1 of U such that $\Phi(W_1) \subseteq W_2$.
- Continuity of the inverse, i.e. Φ^{-1} is continuous.

Two topological spaces are said to be homeomorphic if there exists such a map Φ between them.

Definition 4.1.6. Two topological spaces X_1, X_2 are locally homeomorphic if for every $U \in X_1$ there exists an open set $V_1 \ni U$ with a homeomorphism $\Phi : V_1 \to V_2$, where $V_2 \subset X_2$.

At this point, we have the necessary ingredients to define a manifold.

Definition 4.1.7. A manifold $\mathcal{M}$ is a second countable Hausdorff space that is locally homeomorphic to a Eucledian space $\mathbb{F}^n$ (e.g. $\mathbb{R}^n$, $\mathbb{C}^n$, etc.)

Similarly to optimisation in Eucledian space, optimisation algorithms on manifolds are usually based on the gradient. Therefore, we also need to introduce the notion of differentiability in a manifold.

Definition 4.1.8. A chart (V, Φ) is a homeomorphism Φ between an open set V in a topological space X and an open set in a Eucledian space.

Definition 4.1.9. An atlas is a collection of charts $\{(V_i, \Phi_i)\}_i$ such that they cover the entirety of the topological space X, i.e. $\bigcup_i V_i = X$. Additionally, if for V_a, V_b such that $V_a \cap V_b \neq$, one has that $\Phi_b \circ \Phi_a : \Phi(U_a \cap U_b) \to \Phi_b(U_a \cap U_b)$ is C^r differentiable, then the atlas is said to be a C^r atlas.

Definition 4.1.10. A C^r differentiable manifold is a manifold $\mathcal{M}$ with a C^r atlas. If $r = \infty$, then $\mathcal{M}$ is said to be a smooth manifold.

Just as how points on surfaces in Eucledian space have tangent planes, points on differentiable manifolds have tangent spaces. The definition of derivative on a manifold is also given here, which is a generalisation of the directional derivative in multivariable calculus.

Definition 4.1.11. Given a point $U \in \mathcal{M}$ and a curve $\gamma(t) : [t_0 - T, t_0 + T] \to \mathcal{M}$ such that $\gamma(t_0) = U$, the vector $X_p = \frac{d}{dt}\gamma(t)|_{t0}$ is said to be a tangent vector at U. The tangent space $T_p\mathcal{M}$ of $\mathcal{M}$ at U is the set all tangent vectors at U. The disjoint union of all tangent spaces $\bigsqcup_{U\in\mathcal{M}} T_U\mathcal{M} = \bigcup_{U\in\mathcal{M}}\{U\} \times T_U\mathcal{M}$ is known called the tangent bundle $T\mathcal{M}$.

Definition 4.1.12. Given two manifolds $\mathcal{M}_1$, $\mathcal{M}_2$, a map $\Phi : \mathcal{M}_1 \to \mathcal{M}_2$ and a curve $\gamma : [t_0 - T, t_0 + T] \to \mathcal{M}_1$, such that $\gamma(t_0) = U$, the derivative $D\Phi(U) : T_U\mathcal{M}_1 \to T_{\Phi(U)}\mathcal{M}_2$ is defined through γ on $\mathcal{M}_1$ and $\gamma_\Phi = \Phi \circ \gamma$ on $\mathcal{M}_2$ as

$$D\Phi(U)[\dot{\gamma}(t_0)] = \dot{\gamma}_\Phi(t_0). \tag{4.1}$$

At this point we are finally ready to define a Riemannian manifold, which is the kind of manifold on which we intend to perform optimisation.

Definition 4.1.13. Given a tangent bundle $T\mathcal{M}$, if an inner product $g_U(\cdot,\cdot)$ is defined in each of the $T_U\mathcal{M}$ tangent spaces, then the collection of these inner products is called Riemannian metric g. A smooth manifold together with a Riemannian metric is called a Riemannian manifold $(\mathcal{M}, g)$.

Theorem 4.1.1. *For any smooth manifold, there exists a Riemannian metric.*

Definition 4.1.14. Given a Riemannian manifold $(\mathcal{M}, g)$, the gradient of a map $\Phi : \mathcal{M} \longrightarrow \mathbb{R}$ is defined as the unique tangent vector $\mathrm{grad}\Phi(U)$ that satisfies

$$D\Phi(U)[\xi] = g_U(\mathrm{grad}\Phi(U), \xi), \quad \xi \in T_U\mathcal{M}. \tag{4.2}$$

The fact that $\mathrm{grad}\Phi(U)$ is unique can be checked by considering two tangent vectors $\eta, \zeta \in T_U\mathcal{M}$, such that $D\Phi(U)[\xi] = g_U(\eta, \xi) = g_U(\zeta, \xi)$, $\forall \xi \in T_U\mathcal{M}$. Then, by linearity of the inner product: $g_U(\eta - \zeta, \xi) = 0$, where $\eta - \zeta \in T_U\mathcal{M}$. Since ξ is arbitrary, we have that $\eta = \zeta$.

Proposition 4.1.2. *Given a Riemannian manifold $(\hat{\mathcal{M}}, \hat{g})$, its Riemannian submanifold $(\mathcal{M}, g)$ and the orthogonal projection $\Pi_U : T_U\hat{\mathcal{M}} \longrightarrow T_U\mathcal{M}$. Then the Riemannian gradients of $\hat{\Phi}$ on $\hat{\mathcal{M}}$ and Φ, its restriction to $\mathcal{M}$, satisfy*

$$\mathrm{grad}\Phi(U) = \Pi_U(\mathrm{grad}\hat{\Phi}(U)). \tag{4.3}$$

An important difference between Eucledian and Riemannian optimisation is the fact that the update rule must keep the iteration inside the manifold. It is not sufficient to linearly displace the estimate in the opposite direction of the gradient, since manifolds are not linear nor convex in general. Instead, this displacement is done through retractions on the manifold. That is how the same optimisation problem can be solved in an unconstrained manner, or with a simpler set of constraints.

Definition 4.1.15. A retraction is a smooth map $R : T\mathcal{M} \longrightarrow \mathcal{M}$ such that its restriction R_U to $T_U\mathcal{M}$ satisfies the following conditions:

- $R_U(0_U) = U$.
- $DR_U(0_U) = \mathbb{1}_{T_U\mathcal{M}}$.

We now have the necessary ingredients to talk about Riemannian optimisation problems,

$$\hat{U} = \underset{U\in\mathcal{M}}{\operatorname{argmin}}\Phi(U). \tag{4.4}$$

Theorem 4.1.3. *Given Φ of class C^1, if $U_* \in \mathcal{M}$ is a local optimal solution to the problem (4.4), then* $\operatorname{grad}\Phi(U) = 0$.

However, in this work we are interested in performing optimisation only on a particular kind of Riemannian manifold, which is the Stiefel manifold.

Definition 4.1.16. The set $V_d(\mathbb{F}^n) = \{U \in \mathbb{F}^{n\times d} | U^\dagger U = \mathbb{1}_d\}$, as a submanifold of $\mathbb{F}^{n\times d}$, where $\mathbb{F}$ (e.g. $\mathbb{R}$, $\mathbb{C}$, etc.) and $n \geq d$, is called Stiefel manifold.

In particular, as mentioned above, we are interested in the complex valued Stiefel manifold. Therefore, in what follows we will consider $\mathbb{F} = \mathbb{C}$,

Proposition 4.1.4. *Given $U \in V_d(\mathbb{C}^n)$, $\Pi = UU^\dagger$ is a projector.*

Proof. In order for Π to be a projector it needs to be Hermitian and idempotent, i.e.

$$\Pi^\dagger = (UU^\dagger)^\dagger = UU^\dagger = \Pi, \tag{4.5}$$

$$\Pi^2 = (UU^\dagger)(UU^\dagger) = U\mathbb{1}_d U^\dagger = UU^\dagger = \Pi. \tag{4.6}$$

□

Since any $U, V \in V_d(\mathbb{C}^n)$, there exists a smooth function $X : [0.1] \to V_d(\mathbb{C}^n)$ such that $X(0) = U$ and $X(1) = V$. From the constraint, $X(t)^\dagger X(t) = \mathbb{1}_d$ and therefore,

$$\frac{d}{dt}X(t)^\dagger X(t) = \frac{dX(t)^\dagger}{dt}X(t) + X(t)^\dagger\frac{dX(t)}{dt} = 0_{d,d}. \tag{4.7}$$

The above condition defines the tangent vector space at $U \in V_d(\mathbb{C}^n)$,

$$\mathcal{T}_U V_d(\mathbb{C}^n) := \{X \in \mathbb{C}^{n\times d} | X^\dagger U + U^\dagger X = 0_{d,d}\}. \tag{4.8}$$

Clearly, $\mathcal{T}_U V_1(\mathbb{C}^n)$ is a set of orthogonal vectors (with respect to the Euclidean inner product) of U. Consider the following maps,

$$\pi_U(X) := U\mathrm{Skew}(U^\dagger X) + (\mathbb{1}_n - UU^\dagger)X, \tag{4.9}$$

$$\pi_U^\perp(X) := U\mathrm{Sym}(U^\dagger X), \tag{4.10}$$

for $X \in \mathbb{C}^{n\times d}$, where $\mathrm{Sym}(Z) = (Z + Z^\dagger)/2$ and $\mathrm{Skew}(Z) := (Z - Z^\dagger)/2$ for $Z \in \mathbb{C}^{d\times d}$. From the definition, $X = \pi_U(X) + \pi_U^\perp(X)$ for any $X \in \mathbb{C}^{n\times d}$ and $\pi_U(U) = 0_{n,d}$. π_U is a projection onto $\mathcal{T}_U V_d(\mathbb{C}^n)$ and we have

$$g_U(\pi_U(X), \pi_U^\perp(X)) = 0 \forall X \in \mathbb{C}^{n\times d},$$

where g_U is an inner product at $U \in V_d(\mathbb{C}^n)$ defined as

$$g_U(X, Y) := tr(X^\dagger(\mathbb{1}_n - \frac{1}{2}UU^\dagger)Y), \tag{4.11}$$

for $X, Y \in \mathbb{C}^{n\times d}$. g_U is called the canonical inner product in the field of Riemannian geometry. See Section 2.4 of [5] for the reason why this definition is commonly used. Note that $U \in V_d(\mathbb{C}^n)$ and $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$ are orthogonal with respect to the canonical inner product (4.11), but it does not imply that $U^\dagger X = 0_{d,d}$ in general.

The main retractions on the Stiefel manifold are the following [24]:

- Exponential mapping
$$\mathrm{Retr}_Z^{\mathrm{Exp}}(\xi) = [Z \quad Q] \exp\left(\begin{bmatrix} -Z^\top\xi & -R^\top \\ R & 0 \end{bmatrix}\right) \begin{bmatrix} \mathbb{1}_d \\ 0 \end{bmatrix}, \tag{4.12}$$
where
$$QR = -(\mathbb{1}_d - ZZ^\top)\xi. \tag{4.13}$$

- Polar decomposition
$$\mathrm{Retr}_Z^{\mathrm{Polar}}(\xi) = (Z + \xi)(\mathbb{1} + \xi^\top\xi)^{-1/2}. \tag{4.14}$$

- QR decomposition
$$\mathrm{Retr}_Z^{\mathrm{QR}}(\xi) = \mathrm{QR}(Z + \xi). \tag{4.15}$$

- Cayley transformation
$$\mathrm{Retr}_Z^{\mathrm{Cayley}}(\xi) = (\mathbb{1}_n - \frac{1}{2}W(\xi))^{-1}(\mathbb{1}_n + \frac{1}{2}W(\xi))Z, \tag{4.16}$$
where
$$W(\xi) = (\mathbb{1} - ZZ^\top/2)\xi Z^\top - Z\xi^\top(\mathbb{1} - ZZ^\top/2). \tag{4.17}$$

4.2 Optimisation algorithms

In this section three different kinds of optimisation algorithms are introduced. First, gradient-based algorithms that result in local optimisers, and then, diffusion-based and consensus-based algorithms that approximate global optimisers.

4.2.1 Gradient-based optimisation algorithms

From the previous subsection, we obtain the Euclidean gradient of the cost function $J_d(U)$ at $U \in V_d(\mathbb{C}^n)$. Remember that since $\Pi_U = UU^\dagger = URR^\dagger U^\dagger$ for any unitary $R \in \mathbb{C}^{d\times d}$, the optimal solution is not unique U if it exists. If U changes but Π_U remains a certain matrix, it means that we may find a local optimum. Since we do not know whether the cost function J_d has a unique optimum on $V_d(\mathbb{C}^n)$, we run one of the algorithms below with several initial values (multi-start optimisation).

Remember that the gradient of J_d at $U \in V_d(\mathbb{C}^n)$ is given by (A.2), i.e.,

$$grad J_d(U) = \nabla_U J_d(U) - U(\nabla_U J_d(U))^\dagger U. \tag{4.18}$$

Therefore, the solution of the ordinary differential equation

$$\frac{d}{dt}U(t) = grad J_d(U(t)), \tag{4.19}$$

with $U(0) \in V_d(\mathbb{C}^n)$ always lies on $V_d(\mathbb{C}^n)$ and if there exists the steady state solution, it is a local optimum of J_d. As I mentioned above, a local optimal solution is not unique, but $Pi_U = UU^\dagger$ converges to the local optimum. [11] gives more details on gradient flow.

Note that numerical calculation does not make the flow $\{U(t)\}_{t\geq 0}$ lie on $V_d(\mathbb{C}^n)$. Usually the QR decomposition is employed to correct the error. QR decomposition of $X \in \mathbb{C}^{n\times d}$ is $X = QR$, where $Q \in V_d(\mathbb{C}^n)$ and R is an upper triangular matrix. Therefore, calculate QR decomposition of $(U(t) + dU)$, take the Q part of it, and update as $U(t + dt) = Q$. Another normalisation for a given $X \in \mathbb{C}^{n\times d}$ is to use $U = X(X^\dagger X)^{-1/2}$ if the inverse exists.

To describe the dynamics of the Stiefel manifold, it is useful to use a so-called exponential map explicitly. The exponential map on $V_d(\mathbb{C}^n)$ with respect to the canonical inner product (4.11) defined as follows.

Definition 4.2.1 (Exponential map)**.** A map

$$\exp_U : \mathcal{T}_U V_d(\mathbb{C}^n) \to V_d(\mathbb{C}^n) \tag{4.20}$$

is called an exponential map with respect to the canonical inner product (4.11) if

$$\exp_U(0_{n,d}) = U \text{ and } \frac{d}{dt}\exp_U(tX)|_{t=0} = X, \tag{4.21}$$

for $U \in V_d(\mathbb{C}^n)$ and $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$.

In order to obtain the exponential map, we start with the following exponential matrix. Let us consider the

$$Exp'_U(X) := \exp(XU^\dagger - UX^\dagger), U \in V_d(\mathbb{C}^n), X \in \mathbb{C}^{n\times d}. \tag{4.22}$$

Since $Exp'_U(X)^\dagger = Exp'_U(-X) = Exp'_U(X)^{-1}$ (i.e., $Exp'_U(X)$ is unitary),

$$Exp'_U(tX)U \in V_d(\mathbb{C}^n) \forall t \in \mathbb{R}. \tag{4.23}$$

Furthermore, for any $X \in \mathbb{C}^{n\times d}$,

$$\frac{d}{dt}Exp'_U(tX)U|_{t=0} = X - UX^\dagger U \in \mathcal{T}_U V_d(\mathbb{C}^n).$$

If $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$ (i.e., $U^\dagger X = -X^\dagger U$), then we have

$$\frac{d}{dt}Exp'_U(tX)U|_{t=0} = (\mathbb{1}_n + UU^\dagger)X.$$

It is easy to check that $(\mathbb{1}_n + UU^\dagger)X \in \mathcal{T}_U V_d(\mathbb{C}^n)$ for any $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$. Therefore, for any $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$, there exists $Y \in \mathcal{T}_U V_d(\mathbb{C}^n)$ such that $X = (\mathbb{1}_n + UU^\dagger)^{-1}Y$. The matrix inversion lemma (the Woodbury matrix identity) gives $(\mathbb{1}_n + UU^\dagger)^{-1} = \mathbb{1}_n - \frac{1}{2}UU^\dagger$. Therefore, the following exponential matrix,

$$Exp_U(X) := Exp'_U((\mathbb{1}_n - \frac{1}{2}UU^\dagger)X), U \in V_d(\mathbb{C}^n), X \in \mathcal{T}_U V_d(\mathbb{C}^n), \tag{4.24}$$

defines the exponential map with respect to the canonical inner product (4.11)

$$\exp_U(X) := Exp_U(X)U. \tag{4.25}$$

If n is large, it takes a long time to compute the exponential matrix. To reduce the computational burden, some efficient calculation methods have been proposed (see, e.g., [40, 41]). The following is one of them.

Lemma 4.2.1 ([40]). *For a given $U \in V_d(\mathbb{C}^n)$, consider $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$. Define $A := U^\dagger X \in \mathbb{C}^{d\times d}$ and consider the QR decomposition $QR = (\mathbb{1}_n - UU^\dagger)X$, where $Q \in V_d(\mathbb{C}^n)$ and $R \in \mathbb{C}^{d\times d}$ is upper triangular ($X = UU^\dagger X + (\mathbb{1}_n - UU^\dagger)X = UA + QR$). Then, the following map is an exponential map with respect to the inner product (4.11).*

$$\mathrm{exp}_U^c(X) := \begin{bmatrix} U & Q \end{bmatrix} \exp\begin{pmatrix} A & -R^\dagger \\ R & 0_{d,d} \end{pmatrix} \begin{bmatrix} I_d \\ 0_{d,d} \end{bmatrix} \in V_d(\mathbb{C}^n), \tag{4.26}$$

where exp *represents the matrix exponential function.*

Since the exponential map is uniquely determined [41], $\mathrm{exp}_U^c(X) = \mathrm{exp}_U(X)$ for all $X \in \mathcal{T}_U V_d(\mathbb{C}^n)$. For $t \in \mathbb{R}$, we interpret $tX = U(tA) + Q(tR)$ so that

$$exp_U^c(tX) = \begin{bmatrix} U & Q \end{bmatrix} \exp\left(t \begin{bmatrix} A & -R^\dagger \\ R & 0_{d,d} \end{bmatrix} \right) \begin{bmatrix} I_d \\ 0_{d,d} \end{bmatrix} \in V_d(\mathbb{C}^n). \tag{4.27}$$

Algorithm 1 Gradient descent

```
Require: $U_0 \in V_d(\mathbb{C}^n)$
Ensure: b
1: while not Stopping conditions do
2:     $d_0 \leftarrow gradJ_d(U_0)$
3:     $\Phi(t) \leftarrow J_d(Exp_{U_k}(td_k)U_k)$
4:     $t_k \leftarrow argmax_{t\in[0,\infty)}\Phi(t)$
5:     $U_{k+1} = Exp_{U_k}(t_k d_k)U_k$, $d_{k+1} = gradJ_d(U_{k+1})$
6: end while
```

As described above, numerical computation of an exponential matrix is sometimes hard. To avoid it, other algorithms have been proposed so far. Roughly speaking, these methods allow to leave from $V_d(\mathbb{C}^n)$ once, and using a pullback onto $V_d(\mathbb{C}^n)$ again. The pullback is called retraction and QR decomposition is usually employed for the retraction. QR decomposition of $X \in \mathbb{C}^{n\times d}$ is $X = QR$, where $Q \in V_d(\mathbb{C}^n)$ and R is an upper triangular matrix. We need the Q part, and let $\mathcal{Q} : \mathbb{C}^{n\times d} \to V_d(\mathbb{C}^n)$ be the operation to give the Q part.

These algorithms, however, include the following optimisation problem $t_k \leftarrow argmax_{t\in[0,\infty)}\Phi(t)$. The variable t_k is interpreted as a step length, and the objective is to use the step length that brings the estimate to the minimum of the curve

Algorithm 2 Retraction-based gradient method

```
Require: $U_0 \in V_d(\mathbb{C}^n)$, $d_0 = gradJ_d(U_0)$
Ensure: b
1: while not Stopping conditions do
2:     $\Phi(t) \leftarrow J_d(\mathcal{Q}(U_k + td_k))$
3:     $t_k \leftarrow argmax_{t\in[0,\infty)}\Phi(t)$
4:     $U_{k+1} \leftarrow \mathcal{Q}(U_k + t_k d_k) \in V_d(\mathbb{C}^n)$,
5:     $d_{k+1} \leftarrow gradJ_d(U_{k+1}) \in \mathcal{T}_{U_{k+1}}V_d(\mathbb{C}^n)$
6: end while
```

$\Phi(t)$, which follows the descent direction. This condition can be relaxed to the following,

$$\Phi(R_{U_k}(t_k\eta_k)) \leq \Phi(U_k) + c_1 t_k \langle \mathrm{grad}\Phi(U_k), \eta_k \rangle_{U_k}, \tag{4.28}$$

which is know as Armijo condition. This condition only requires the estimate update to be good enough, where "good enough" is regulated by the parameter $0 < c_1 < 1$. An additional conditional that is commonly used, together with Armijo condition is the following,

$$\langle \mathrm{grad}\Phi(R_{U_k}(t_k\eta_k)), DR_{U_k}(t_k\eta_k)[\eta_k] \rangle_{R_{U_k}}(t_k\eta_k) \geq c_2 \langle \mathrm{grad}\Phi(U_k), \eta_k \rangle_{U_k}. \tag{4.29}$$

This is known as Wolfe condition, and $0 < c_1 < c_2 < 1$. A stronger version of this, known as strong Wolfe condition, is the following,

$$|\langle \mathrm{grad}\Phi(R_{U_k}(t_k\eta_k)), DR_{U_k}(t_k\eta_k)[\eta_k] \rangle_{R_{U_k}(t_k\eta_k)}| \leq c_2 |\langle \mathrm{grad}\Phi(U_k), \eta_k \rangle_{U_k}|. \tag{4.30}$$

In order to find a step size that satisfies the Armijo condition the backtracking algorithm is used. Additionally, the two-point method developed in [2] can be used to speed up the calculation of the step size by substituting the fixed initial estimate of the backtracking algorithm by the one given by this method.

Given the current and previous estimates of the optimisation algorithm, U_k, U_{k-1}, the two-point method provides the following initial estimate for the step size t. Let $\Delta U_k = U_k - U_{k-1}$ be the difference between the last two iterations, and $\Delta G_k = G_k - G_{k-1}$ the difference between their gradients, then the two-point step size is

Algorithm 3 Backtracking

```
Require: $X_0^i \in V_d(\mathbb{F}^n)$,   $T$,   $h$
  $t \leftarrow t_0$
  while $\Phi(R_{U_k}(t_k \eta_k)) > \Phi(U_k) + c_1 t_k \langle \mathrm{grad}\Phi(U_k), \eta_k \rangle_{U_k}$ do
    $t \leftarrow \tau t_k$
  end while
  $t_k \leftarrow t$
```

$$t_k = \frac{\langle \Delta U_k, \Delta G_k \rangle}{\langle \Delta G_k, \Delta G_k \rangle}. \tag{4.31}$$

Finally, the following theorem establishes the convergence of this kind of optimisation algorithm.

Theorem 4.2.2 (Zoutendijk's theorem for convergence)**.** *Let θ_k be the angle between the search direction η_k and the steepest descent direction $-\mathrm{grad}\Phi(U_k)$, i.e.,*

$$\cos(\theta_k) = -\frac{\langle \mathrm{grad}\Phi(U_k), \eta_k \rangle_{U_k}}{\|\mathrm{grad}\Phi(U_k)\|_{U_k} \|\eta_k\|_{U_k}}. \tag{4.32}$$

Given a step length t_k that satisfies the Armijo-Wolfe conditions for every integer $k \geq 0$, if the objective function Φ is bounded below and is of class C^1, and if there exists a constant $L > 0$ such that

$$|D(\Phi \circ R_U)(t\eta)[\eta] - D(\Phi \circ R_U)(0)[\eta] \leq Lt, \tag{4.33}$$

where $\eta \in T_U\mathcal{M}$ with $\|\eta\|_U = 1$, $U \in \mathcal{M}$, $t \leq 0$. Then, for θ_k the following series converges,

$$\sum_{k=0}^{\infty} \cos^2(\theta_k) \|\mathrm{grad}\Phi(U_k)\|_{U_k}^2 < \infty. \tag{4.34}$$

4.2.2 Diffusion-based algorithms

The following global optimisation algorithms rely heavily on the dynamics of diffusion processes. Imagine to have a particle ensemble inside a volume. A diffusion process is the evolution of the concentration of these particles, due to the density gradient within the volume. In general, diffusion dynamics are governed by the following equation,

$$\partial_t \rho(x,t) = \nabla \cdot (D(x,t)\nabla \rho(x,t)). \tag{4.35}$$

If the density is normalised, a diffusion process can also be understood as the random walk of a single particle within a volume with initial probability distribution $w(x,t_0) = \rho(x,t_0)$. If the function $D(x,t)$ is a constant diffusion coefficient D, then the diffusion equation has the following solution,

$$w(x,t) = \int dx_0 W(x,t|x_0,t_0)w(x_0,t_0), \tag{4.36}$$

where

$$W(x,t|w_0,t_0) = \frac{1}{\sqrt{4\pi D(t-t_0)}} e^{-\frac{(x-x_0)^2}{4D(t-t_0)}}, \tag{4.37}$$

and $w(x_0,t_0)$ is the initial condition.

Under the action of an external force, one would get stochastic processes described by the Fokker-Planck equation,

$$\partial_t w(x,t) = \nabla \cdot [-f(x,t)w(x,t) + \nabla[D(x,t)w(x,t)]], \tag{4.38}$$

where $f(x(t),t) = F_{ext}(x(t),t)/\gamma$ is the drift velocity, γ is the viscous friction coefficient of the medium, and $F_{ext}(x(t),t)$ is the external force. In what follows, let us assume $\gamma = 1$.

Proposition 4.2.3. *If the external force is conservative, i.e. there exists a potential $V(x)$ such that $f(x) = -\nabla V(x)$, and assuming a constant diffusion coefficient $D(x,t) = D$, the stationary solution of the Fokker-Planck equation is*

$$W_*(x) = \frac{1}{Z} e^{-V(x)/D}, \tag{4.39}$$

which is known as Boltzmann distribution.

Proof. Multiply both sides of the Fokker-Planck equation by $e^{V(x)/(2D)}$ and define $\hat{w}(x,t) = e^{V(x)/(2D)}(x,t)$ and $\hat{V}(x) = V(x)/(2D)$,

$$\partial_t \hat{w}(x,t) = e^{\hat{V}(x)} \nabla \cdot (2D\nabla \hat{V}(x)w(x,t) + D\nabla w(x,t)). \tag{4.40}$$

Now apply the rule $e^{-f(x)}(\nabla \cdot e^{f(x)}G(x)) = (\nabla + \nabla f(x)) \cdot G(x)$ with $f(x) = \hat{V}(x)$ and $G(x) = 2D\nabla \hat{V}(x)w(x,t) + D\nabla w(x,t)$,

$$\begin{aligned}\partial_t \hat{w}(x,t) =& e^{\hat{V}(x)}\nabla \cdot G(x) \\ =& e^{\hat{V}(x)}e^{-\hat{V}(x)}(\nabla \cdot e^{\hat{V}(x)}G(x)) - e^{\hat{V}(x)}\nabla\hat{V}(x)\cdot G(x) \\ =& \nabla \cdot e^{\hat{V}(x)}G(x) - \nabla\hat{V}(x)\cdot e^{\hat{V}(x)}G(x) \\ =& (\nabla - \nabla\hat{V})\cdot(2D\nabla\hat{V}(x)\hat{w}(x,t) + De^{\hat{V}(x)}\nabla w(x,t))\end{aligned} \quad . \tag{4.41}$$

Now apply the rule $e^{-f(x)}(\nabla e^{f(x)}g(x)) = (\nabla + \nabla f(x))g(x)$ with $f(x) = \hat{V}(x)$ and $g(x) = w(x,t)$,

$$\begin{aligned}\partial_t \hat{w}(x,t) =& (\nabla - \nabla\hat{V})\cdot(2D\nabla\hat{V}(x)\hat{w}(x,t) + D\nabla\hat{w}(x,t) - D\nabla\hat{V}(x)\hat{w}(x,t)) \\ =& (\nabla - \nabla\hat{V})\cdot(D\nabla\hat{V}(x)\hat{w}(x,t) + D\nabla\hat{w}(x,t)) \\ =& D(\nabla - \nabla\hat{V})\cdot(\nabla + \nabla\hat{V})\hat{w}(x,t)\end{aligned} \quad . \tag{4.42}$$

Define now $A = \nabla\hat{V} + \nabla$ and $A^\dagger = (\nabla\hat{V} - \nabla)^\top$, then the Fokker-Planck equation can be written as

$$\partial_t \hat{w}(x,t) = -DA^\dagger A w. \tag{4.43}$$

Notice that

$$\int_X d^d x \phi(x) A^\dagger A \phi(x) = \int_X d^d x (A\phi(x))^\dagger (A\phi(x)) = \int_X d^d x \|A\phi(x)\|^2 \geq 0, \tag{4.44}$$

where the equality holds iff $A\phi(x) = 0$, that is $\hat{w}(x,t) \propto e^{-\hat{V}}$. Thus, the stationary solution of the Fokker-Planck equation under a conservative force is $W_* = e^{-V/D}/Z$, where $Z = \int_X d^d x e^{-V/D}$ is the normalisation constant. □

For the purposes of simulations, an approach based on Brownian motion can also be pursued. In particular, one would be interested in implementing the following discrete time Langevin equation,

$$\Delta x(t) = f(x(t),t)\Delta t + \sqrt{2D}\Delta B(t), \tag{4.45}$$

where

$$\Delta B(t) \sim \mathcal{N}(0, \Delta t). \tag{4.46}$$

The Langevin equation is a Stochastic differential equation (SDE). As such, the sampling time of $dB(t)$ is important, since different sampling times within each integration interval leads to different results. There are two standard frameworks, known as Itô an Stratonovich prescriptions. Take $\tau_k = \lambda t_k + (1-\lambda)t_{k-1}$, with $0 \leq \lambda \leq 1$. If the samples $dB(\tau_k)$ are taken with $\lambda = 0$, then it the Itô prescription, and if $\lambda = 1/2$, then it is the Stratonovich prescription. When one of these two prescriptions is used, the following notations are used,

$$S_{\text{Itô}} = \int_{t_0}^{t} B(\tau)dB(\tau) = \frac{B^2(t) - B^2(t_0)}{2} - \frac{t - t_0}{2}, \tag{4.47}$$

$$S_{\text{Stratonovich}} = \int_{t_0}^{t} B(\tau) \circ dB(\tau) = \frac{B^2(t) - B^2(t_0)}{2}, \tag{4.48}$$

i.e. the Stratonovich is denoted by an empty circle product sign and the Itô prescription is denoted by no product sign. If a general prescription is used, also called λ-prescription, the following notation is used

$$S_\lambda = \int_{t_0}^{t} B(\tau)dB(\tau)|_\lambda = \frac{B^2(t) - B^2(t_0)}{2} + (t - t_0)\frac{2\lambda - 1}{2}. \tag{4.49}$$

Additionally, equations (4.47,4.48,4.49) show the results of the integral of the identity function under the different prescriptions, i.e. the equivalent of the integral $\int_{t_0}^{t} f(\tau)df(\tau) = (f^2(t) - f^2(0))/2$ in non-stochastic calculus. The Itô prescription has the advantage that it preserves causality, since the noise acts before the state update. But the Stratonovich prescription has the advantage that it reproduces the results from non-stochastic calculus and it is invariant under time reversal.

As such, the continuous time Langevin equation under the Itô prescription would be

$$dx(t) = f(x(t), t)dt + \sqrt{2D}dB(t). \tag{4.50}$$

Notice how in the absence of noise, i.e. $D = 0$, and under conservative forces only, i.e. $f(x(t), t) = -\nabla V(x(t))$, simulating this equation is equivalent to performing the steepest descent method (with fixed step size) if we use the objective function as the potential $V(x)$.

We introduce now an algorithm that is based on these processes. It uses the cost function as a potential on which the walk of a single particle is simulated. It is called IDDM and was first introduced in [38], for the real-valued Stiefel manifold. Here it has been adapted to the problem at hand. Namely, the algorithm has been adapted to run over the complex-valued Stiefel manifold, instead of the real-valued one.

At its core, this algorithm is a version of simulated annealing, with a noise term that not only is diminishing but also intermittent. Another way to interpret it is as an alternation between simulated annealing and gradient descent.

The usual gradient descent method, when used for solving non-convex problems, may converge to local stationary points. Therefore, finding the global minimum of a function is not guaranteed by the algorithms mentioned up to this point. To overcome this obstacle, a noise term may be added, resulting in the following SDE,

$$dU(t) = -\mathrm{grad} J_d(U(t))dt + \sigma(t) \circ dB_{V_d(\mathbb{C}^n)}(t), \tag{4.51}$$

where $B_{V_d(\mathbb{C}^n)}(t)$ is an n-dimensional Brownian motion on the Stiefel manifold, and $\circ$ indicates the Stratonovich prescription. The diffusion coefficient $\sigma(t)$, if chosen with and appropiate decreasing profile, and given certain regularity conditions on $J_d(U)$, guarantees the convergence of the optimiser to the global minimum of the function. This method is called Simulated Annealing, or Continuous Diminishing Diffusion.

If the diffusion coefficient is chosen with a piecewise constant profile of the form $\sigma(t) = \sum_{i=1}^{N} \sigma_i \mathbb{1}_{[S_i, S_i+T_i]}(t)$, the method is called Intermittent Diffusion. This profile has value σ_i in the interval $[S_i, S_i + T_i]$ and zero in the interval $(S_i + T_i, S_{i+1})$. The proposed algorithm is a combination of Simulated Annealing and Intermittent Diffusion on the Stiefel manifold. Therefore, it is called Intermittent Diminishing Diffusion on Manifold (IDDM).

However, to make use of (4.51) in numerical optimisation, an extrinsic presentation is necessary. One using the embedding coordinates of $V_d(\mathbb{C}^n)$ into $\mathbb{C}^{n\times d}$ is proposed.

$$dU(t) = -\mathrm{grad} J_d(U(t))dt + \sigma(t) \sum_{u=1}^{n} \sum_{v=1}^{d} P_{uv}(U(t)) \circ dB_{uv}(t), \tag{4.52}$$

where $\{B_{uv}(t)\}$ is a series of independent standard Brownian motions and

P_{uv} is the projection of the one-entry matrices E_{uv} onto the tangent space at $U(t)$.

The equivalent Itô SDE is

$$dU(t) = \left(-\mathrm{grad}J_d(U(t)) - \frac{n-1}{2}\sigma^2(t)U(t)\right)dt + \sigma(t)P_U(dB(t)), \tag{4.53}$$

where $P_U(dB(t)) = \sum_{u=1}^{n}\sum_{v=1}^{d} P_{uv}(U(t))dB_{uv}(t)$.

In order to solve the SDE numerically, first project the random noise in the ambient space onto the tangent space of the Stiefel manifold. After that, apply the Cayley transformation. The following update rule is proposed,

$$Z_k = -\delta_k G_k + \sigma_k(\mathbb{1}_n - \beta Y_k Y_k^T)\delta B_k, \tag{4.54}$$

$$W_k = Z_k Y_k^T - Y_k Z_k^T, \tag{4.55}$$

$$Y_{k+1} = (\mathbb{1} - \frac{W_k}{2})^{-1}(\mathbb{1} + \frac{W_k}{2})Y_k, \tag{4.56}$$

where G_k is the Eucledian gradient and δ_k is the time difference at time k.

Algorithm 4 Numerical scheme for the diffusion process

```
**Require:** $\sigma(t)$,  $t_0 = \tau_0 < \tau_1 < \ldots < \tau_K = T$,  $Y_0 = U(t_0)$
  $\delta_k \leftarrow \tau_{k+1} - \tau_k$
  $\sigma_k \leftarrow \sigma(\tau_k)$
  $G_k \leftarrow \nabla_E J_d(Y_k)$
  $\{\delta B_k\}_{k=0}^{K-1} \sim N(0, \delta_k)$
  **for** $k = 0 : K-1$ **do**
    $Z_k \leftarrow -\delta_k G_k + \sigma_k(\mathbb{1}_n - \beta Y_k Y_k^T)\delta B_k$
    $W_k \leftarrow Z_k Y_k^T - Y_k Z_k^T$
    $Y_{k+1} \leftarrow (\mathbb{1} - \frac{W_k}{2})^{-1}(\mathbb{1} + \frac{W_k}{2})Y_k$
  **end for**
  $U(\tau_k) \leftarrow Y_k$
```

Instead of the exponential map, an implicit mid-point update scheme is used to compute the next estimate,

$$Y(\tau) = U - \tau W\left(\frac{U + Y(\tau)}{2}\right), \tag{4.57}$$

where $-\mathrm{grad}J_d(U) = -WU$. This update scheme is equivalent to the following Cayley transformation,

$$Y(\tau) = (\mathbb{1} + \frac{\tau}{2}W)^{-1}(\mathbb{1} - \frac{\tau}{2}W)U. \tag{4.58}$$

Hence, as shown in the previous section, this is a retraction in the Stiefel manifold and orthonormality is guaranteed at each iteration.

Under certain conditions the Frobenius norm of the gradient converges to zero. These conditions are the following.

- W is continuous in U.
- $J'_{d_\tau}(Y(0)) \leq -\sigma\|W\|_F^2$, where $J'_{d_\tau} = \frac{\partial J_d(Y(\tau))}{\partial \tau}|_{\tau=0} = -\frac{1}{2}\|W\|_F^2$ and $\sigma > 0$ is constant.

Algorithm 5 Intermittent Diminishing Diffusion on the Stiefel Manifold (IDDM)

```
Require: $N$,   $\sigma_n$,   $T_n$,   $U_0$
  while Terminal conditions not satisfied do
    if $k \geq N$ then
      break
    end if
    Use Alg. 4 to solve (4.52)
    Use Alg. 2 until convergence to local minimum
    if $f(U_{k+1}) < f(U_{opt})$ then
      $U_{opt} \leftarrow U_{k+1}$;
    end if
    $k \leftarrow k + 1$
  end while
```

4.2.3 Consensus method

The last algorithm to be considered is a CBO algorithm, developed in [13]. Here, the algorithm is adapted to the complex-valued Stiefel manifold. This algorithm is similar to the previous one in the sense that it also simulates random walks and relies of the solution of a SDE. However, in this case many many interacting particles are simulated and the gradient of the cost function is not necessary. This interaction is weighted by the cost function, in a way such that the particle with the lowest cost exerts a greater attraction.

The CBO model is given by a stochastic interacting particle system for N agents $x_t^i \in \mathbb{R}^d$,

$$dx_t^t = -\lambda(x_t^i - \bar{x}_t^*)dt + \sigma|x_t^i - \bar{x}_t^*|dw_t^i, \tag{4.59}$$

where $i = 1, ..., N$, w_t^i is the standard Wiener process on $\mathbb{R}^d$, representing a Brownian motion, σ is the noise strength for exploration and λ represents the strength of the consensus dynamics. N is the number of particles. A larger number of particles provides a better approximation of global minimisers. Finally, $\bar{x}_t^*$ is the weighted average of the particle positions, according to the Boltzmann distribution, i.e.

$$\bar{x}_t^* = \sum_{i=1}^{N} (\frac{\omega_f^\beta(x_t^i)}{\sum_{j=1}^N \omega_f^\beta(x_t^j)}) x_t^i, \tag{4.60}$$

and

$$\omega_f^\beta(s) = \exp(-\beta f(x)), \tag{4.61}$$

where β is the inverse temperature. A larger inverse temperature provides a better approximation.

Over the complex-valued Stiefel manifold, the previous model would be

$$dU_t^i = -\lambda P_{U_t^i}(U_t^i - \bar{U}_t^*)dt + \sigma|U_t^i - \bar{U}_t^*|P_{U_t^i}(dW_t) - C_{n,d}\frac{\sigma_t^2|U_t^i - \bar{U}_t^*|^2}{2}U_t^i dt, \tag{4.62}$$

where $i = 1, ..., N$ and $\{W_t\}$ is the standard Wiener process on $\mathbb{C}^{n\times d}$, which is defined by Itô's sense, and

$$C_{n,d} = \frac{2n - d - 1}{2}, \tag{4.63}$$

$$\bar{U}_t^* = \sum_{i=1}^{N} U_t^i [\frac{\omega_f^\beta(U_t^i)}{\sum_{j=1}^N \omega_f^\beta(U_t^j)}]. \tag{4.64}$$

However, since $P_{U_t^i}(U_t^i) = 0$, the stochastic system can be reduced to

$$dU_t^i = (\lambda P_{U_t^i}(\bar{U}_t^*) - C_{n,d}\frac{\sigma^2|U_t^i - \bar{U}_t^*|^2}{2}U_t^i)dt + \sigma|U_t^i - \bar{U}_t^* P_{U_t^i}(dW_t). \tag{4.65}$$

Following the Euler-Maruyama method, the SDE is discretised as

$$U^i_{n+1} = U^i_n + (\lambda P_{U^i_n}(\bar{U}^*_n) - C_{n,d}\frac{\sigma^2|U^i_n - \bar{U}^*_n|^2}{2}U^i_n)\Delta t + \sigma|U^n_i - \bar{U}^*_n|P_{U^n_i}(\Delta W^i_{n+1}), \tag{4.66}$$

where Δt is the time step size and ΔW^i_{n+1} denotes an $n \times d$ matrix with i.i.d. $\mathcal{N}(0, \Delta t)$ components.

However, this update rule does not preserve orthogonality. Therefore, a projection step is added, so that $U = W\Sigma V^T$ is mapped to $\Pi(U) = W\mathbb{1}_{n,d}V^T$.

The term $-C_{n,d}\frac{\sigma^2|U^i_t - \bar{U}^*_t|^2}{2}U^i_t dt$ keeps the points inside the manifold.

Algorithm 6 Consensus method

```
Require: $U^i_0 \in V_d(\mathbb{F}^n)$,   $T$,   $h$
  $\delta_d \leftarrow \tau_{d+1} - \tau_d$
  while $nh < T$ do
    $\Delta W^i_{n+1} \sim \mathcal{N}(0, \Delta t)$
    $U^i_{n+\frac{1}{2}} \leftarrow U^i_n + (\lambda P_{U^i_n}(\bar{U}^*_n) - C_{n,d}\frac{\sigma^2|U^i_n - \bar{U}^*_n|^2}{2}U^i_n)\Delta t + \sigma|U^n_i - \bar{U}^*_n|P_{U^n_i}(\Delta W^i_{n+1})$
    $U^i_{n+1} \leftarrow \Pi(U^i_{n+\frac{1}{2}})$
  end while
```

5 Numerical tests and applications

The cases that have been chosen for study are the following:

- Bit-flip channel: This is a simple noise channel that can be corrected with as few as three qubits. This made an ideal candidate for testing the correctness of the algorithms and other tests.
- Bit-flip channel with correlated term: A modified version of the previous channel was designed in order to test a case that is not considered in the standard QEC theory. This also served as a benchmark of the robustness of the known and the optimised codes.
- Amplitude damping channel: This is another commonly studied noise channel, which represents the effect of energy loss to the environment.
- Depolarising channel: This is the most general noise channel. Finding a code able to correct this channel would be equivalent to finding a code able to correct any non-correlated qubit noise. The smallest code able to correct it has a size of five qubits.

5.1 Bit-flip

In a system of qubits, the bit-flip noise is a quantum channel that rotates the state of a random qubit around the X axis of the Bloch sphere, with probability p. This rotation is represented by the first Pauli matrix

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{5.1}$$

which maps $|0\rangle \mapsto |1\rangle$ and $|1\rangle \mapsto |0\rangle$. That is why it is called bit-flip.

The codes considered in this section are three-qubit codes. As such, the Kraus operators for the noise channel are the following,

$$\begin{aligned} \mathcal{N} \sim \{ & \sqrt{1-p}\mathbb{1}^{\otimes 3}, \\ & \sqrt{\frac{p}{3}}\sigma_x \otimes \mathbb{1} \otimes \mathbb{1}, \\ & \sqrt{\frac{p}{3}}\mathbb{1} \otimes \sigma_x \otimes \mathbb{1}, \\ & \sqrt{\frac{p}{3}}\mathbb{1} \otimes \mathbb{1} \otimes \sigma_x \} \end{aligned} . \tag{5.2}$$

As benchmark, the Shor code [27] for bit-flip errors is used in this section. The basis for this code is the following,

$$\{|0_L\rangle, |1_L\rangle\} = \{|000\rangle, |111\rangle\}. \tag{5.3}$$

This code belongs to a family of subspace codes known as stabiliser codes. These codes are described by a set of operators called generators, and the code's subspace is given by the intersection of the +1 eigenspaces of these generators. It is easy to confirm that indeed the basis elements given above are +1 eigenvectors of the generators given in table5.1.

g_1	Z	Z	I
g_2	I	Z	Z

Table 5.1: Generators of the known $(2^3, 2^1)$ bit-flip code

In order to obtain a code for this channel, the gradient descent algorithm was used. After several runs of the algorithm, it has been concluded that the optima of the cost function $J_3(U)$, under this noise channel, has multiple local optima. But they are all equal to the global optimum. Figures 5.1, 5.2 show a comparison of the performance of the optimised code with the Shor code and a case without any recovery action. The basis of the optimised code is the following,

$$\begin{aligned}\{|0_L\rangle, |1_L\rangle\} = \{&(-0.4478+0.5384i)|000\rangle + (-0.0645+0.1473i)|001\rangle + \\ &(-0.2461-0.2396i)|010\rangle + (0.0465+0.2486i)|011\rangle + \\ &(-0.0890+0.1012i)|100\rangle + (-0.1945-0.2070i)|101\rangle + \\ &(0.0048+0.2100i)|110\rangle + (0.3616-0.1678i)|111\rangle, \\ &(0.2566+0.1149i)|000\rangle + (-0.1515+0.1936i)|001\rangle + \\ &(-0.0486-0.3594i)|010\rangle + (-0.1573+0.0784i)|011\rangle + \\ &(-0.1387+0.2397i)|100\rangle + (-0.0669-0.4054i)|101\rangle + \\ &(-0.1670+0.1194i)|110\rangle + (-0.6112+0.1921i)|111\rangle\} \quad . \end{aligned} \tag{5.4}$$

This basis spans a space that is different to that of the Shor code. It is also easy to verify that these basis elements are not eigenvectors of the generators 5.1. For illustration of the performance of the obtained code, the fidelity of the output of the noise and recovery with respect to the original state $|0_L\rangle$ is shown in 5.1. However, considering theorem 3.1.3, the actual fidelity of importance to understand the correctability of the code is that between the output of $\mathbb{1} \otimes \mathcal{A}$ and the input state $(|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|)$, where $\mathcal{A} = \mathcal{R}_\Pi \circ \mathcal{N}$. This latter case is shown in 5.2.

The Shor code does not protect against correlated errors. In order to test the possibilities of our optimisation scheme, let us consider the following modification to the bit-flip channel,

$$\begin{aligned}\mathcal{N} \sim \{&\sqrt{1-p}\mathbb{1}^{\otimes 3}, \\ &\sqrt{\frac{p(1-q)}{3}}\sigma_x \otimes \mathbb{1} \otimes \mathbb{1}, \\ &\sqrt{\frac{p(1-q)}{3}}\mathbb{1} \otimes \sigma_x \otimes \mathbb{1}, . \\ &\sqrt{\frac{p(1-q)}{3}}\mathbb{1} \otimes \mathbb{1} \otimes \sigma_x, \\ &\sqrt{pq}\mathbb{1} \otimes \sigma_x \otimes \sigma_x\}\end{aligned} \tag{5.5}$$

Here a correlated term $\sigma_x \otimes \sigma_x$ has been introduced, such that a simultaneous bit-flip can occur in the second and third qubits. After running the gradient descent algorithm with multiple initial points, with and without the correlated

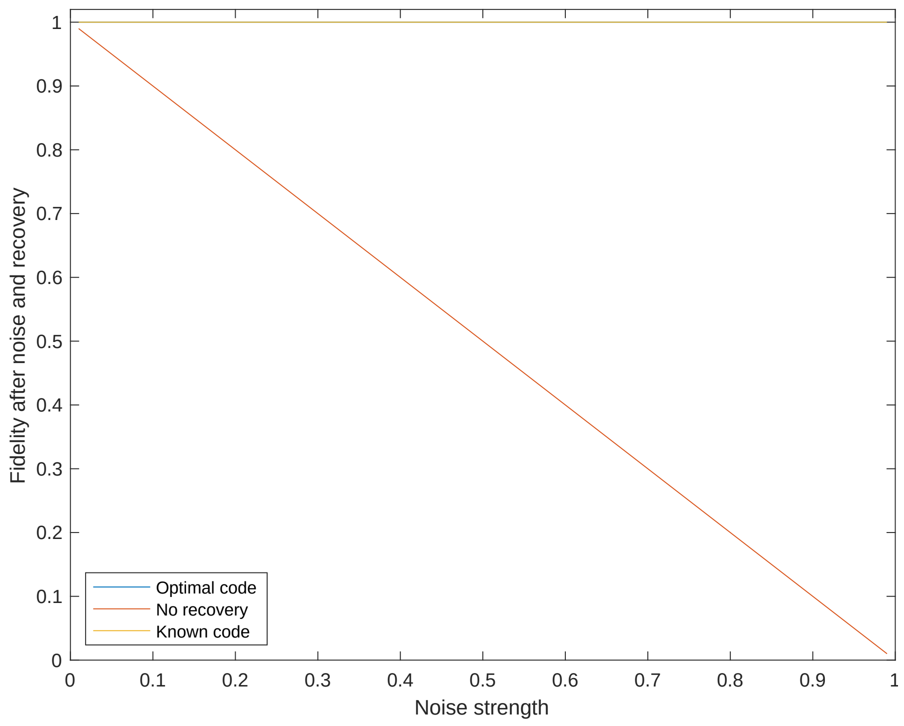

Figure 5.1: Fidelity of $\mathcal{A}(|0_L\rangle\langle 0_L|)$ under the $(2^3, 2^1)$ bit-flip codes

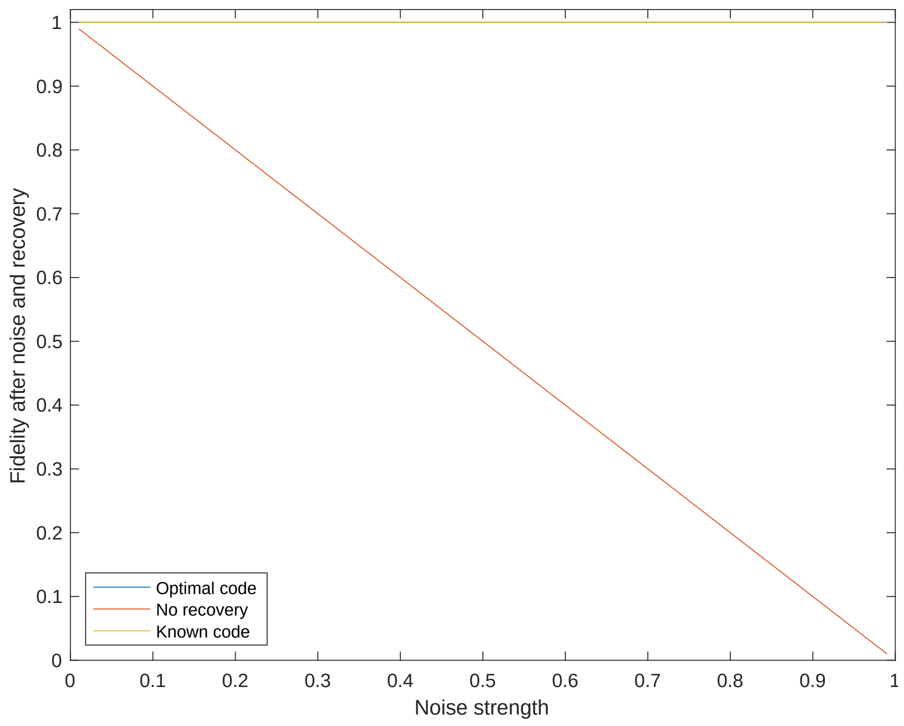

Figure 5.2: Fidelity of $\mathbb{1} \otimes \mathcal{A}((|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|))$ under the $(2^3, 2^1)$ bit-flip codes

term, it has been concluded that the addition of this term does not alter the structure of the optima of the cost function. Instead, it only lowers the optimal value. This effect can be seen in figure 5.3.

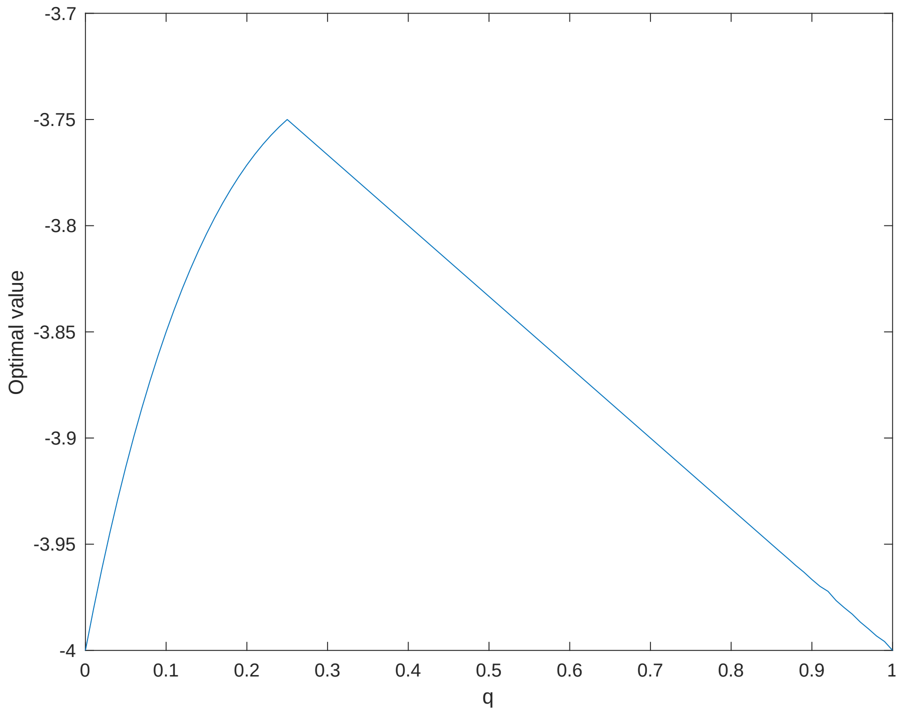

Figure 5.3: Optimal value for different correlation strengths

Figures 5.4, 5.5 show the performance when the channel is modified with a correlation term $q = 0.1$. The basis of the optimised code for this case is

$$\{|0_L\rangle, |1_L\rangle\} = \left\{ \begin{array}{l} (-0.1058 - 0.2861i)|000\rangle + (0.3977 - 0.1287i)|001\rangle + \\ (0.3040 - 0.5191i)|010\rangle + (0.0861 + 0.2100i)|011\rangle + \\ (0.1457 + 0.3208i)|100\rangle + (-0.1082 + 0.1389i)|101\rangle + \\ (0.0756 - 0.3557i)|110\rangle + (-0.0347 - 0.1741i)|111\rangle, \\ (-0.1174 - 0.0094i)|000\rangle + (-0.3317 - 0.0562i)|001\rangle + \\ (0.2150 + 0.2200i)|010\rangle + (0.3028 - 0.0747i)|011\rangle + \\ (0.1545 - 0.0283i)|100\rangle + (-0.4871 - 0.4338i)|101\rangle + \\ (0.0191 - 0.3940i)|110\rangle + (-0.2698 + 0.0507i)|111\rangle \end{array} \right\}. \tag{5.6}$$

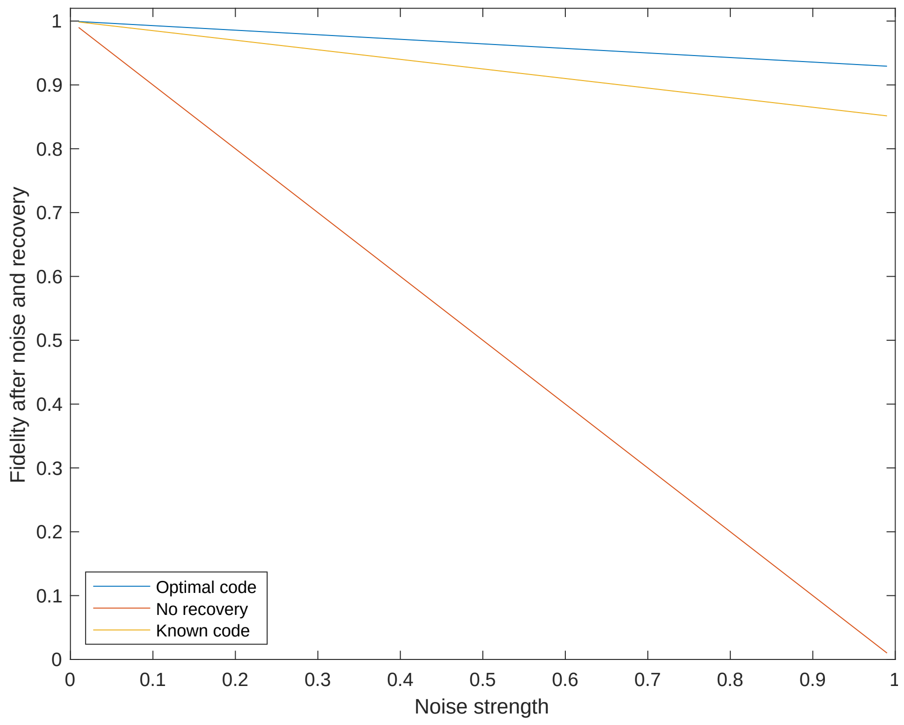

Figure 5.4: Fidelity of $\mathcal{A}(|0_L\rangle\langle 0_L|)$ under the $(2^3, 2^1)$ codes and a correlated bit-flip channel

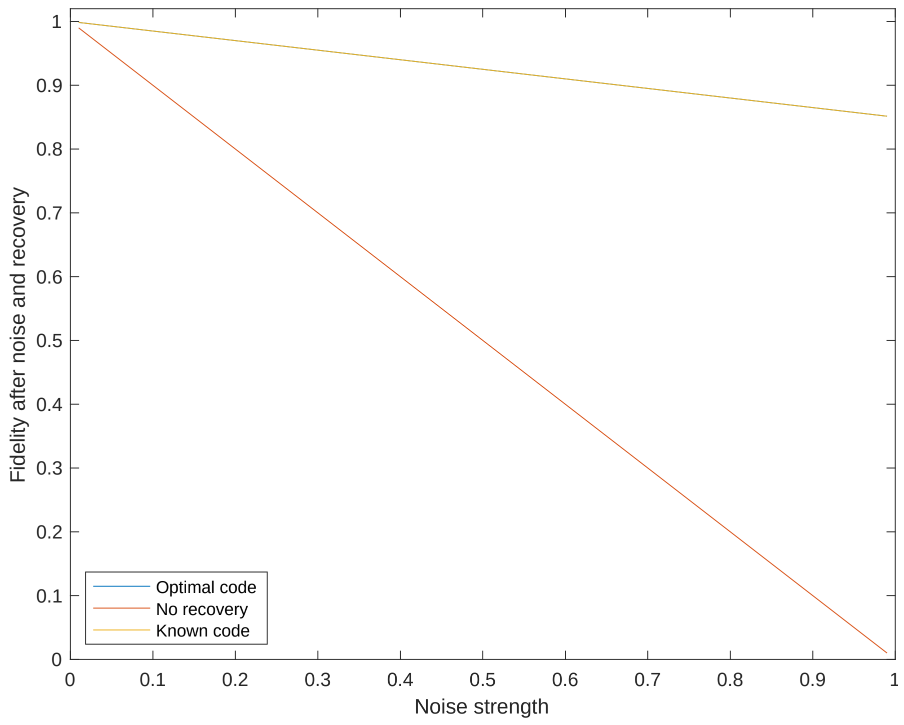

Figure 5.5: Fidelity of $\mathbb{1} \otimes \mathcal{A}((|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|))$ under the $(2^3, 2^1)$ codes and a correlated bit-flip channel

5.2 Amplitude damping

The amplitude damping channel models the energy loss of the quantum system to its environment, for instance through spontaneous emission. Its Kraus operators are based on the following two operators,

$$E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, \tag{5.7}$$

$$E_1 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}. \tag{5.8}$$

E_1 corresponds to an energy decay with probability p, whereas E_0 ensures trace preservation. In the case of four qubits, the following would be the Kraus representation of the noise channel,

$$\begin{aligned} \mathcal{N} \sim \{&\sqrt{\frac{1}{4}}\mathbb{1}^{\otimes 4}, \\ &\sqrt{\frac{1}{4}}E_0 \otimes \mathbb{1}^{\otimes 3}, \sqrt{\frac{1}{4}}E_1 \otimes \mathbb{1}^{\otimes 3}, \\ &\sqrt{\frac{1}{4}}\mathbb{1} \otimes E_0 \otimes \mathbb{1}^{\otimes 2}, \sqrt{\frac{1}{4}}\mathbb{1} \otimes E_1 \otimes \mathbb{1}^{\otimes 2},. \\ &\sqrt{\frac{1}{4}}\mathbb{1}^{\otimes 2} \otimes E_0 \otimes \mathbb{1}, \sqrt{\frac{1}{4}}\mathbb{1}^{\otimes 2} \otimes E_1 \otimes \mathbb{1}, \\ &\sqrt{\frac{1}{4}}\mathbb{1}^{\otimes 3} \otimes E_0, \sqrt{\frac{1}{4}}\mathbb{1}^{\otimes 3} \otimes E_1\} \end{aligned} \tag{5.9}$$

An approximate four qubit code is known for this channel [22]. It has the following basis,

$$\{|0_L\rangle, |1_L\rangle\} = \{\frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle), \frac{1}{\sqrt{2}}(|0011\rangle + |1100\rangle)\}. \tag{5.10}$$

The amplitude damping channel is an interesting study case, because its optima are not unique as in the previous case. As such, it is good to study the performance of the two global optimisation algorithms. Figure 5.6 shows the value of the cost function for the state returned by each algorithm at different damping rates. It is interesting to notice that the local algorithm with 100 initialisations performed better than IDDM and the CBO algorithm, and that the known code performs better than the ones obtained by optimisation.

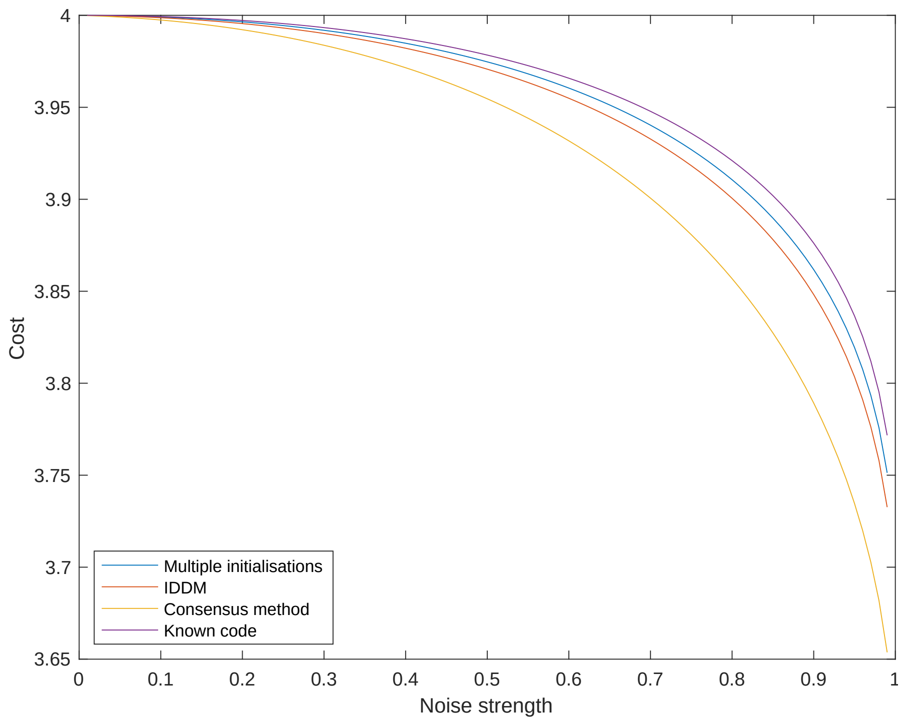

Figure 5.6: Comparison of the $(2^4, 2^1)$ codes obtained from different algorithms

Then, figures 5.7, 5.8 show the performance of the best code obtained by multiple initialisations. The base of this code is the following,

$$\begin{aligned}
\{|0_L\rangle, |1_L\rangle\} = \{&(-0.0627 + 0.6273i)|0000\rangle + (-0.0599 + 0.0851i)|0001\rangle + \\
&(0.0554 - 0.0828i)|0010\rangle + (0.0210 + 0.0010i)|0011\rangle + \\
&(-0.0675 + 0.0702i)|0100\rangle + (0.0179 - 0.1380i)|0101\rangle + \\
&(-0.1659 - 0.0845i)|0110\rangle + (-0.0940 - 0.0587i)|0111\rangle + \\
&(-0.0588 + 0.0917i)|1000\rangle + (0.1784 + 0.0131i)|1001\rangle + \\
&(0.0006 + 0.1471i)|1010\rangle + (-0.1024 - 0.0308i)|1011\rangle + \\
&(-0.0178 + 0.0063i)|1100\rangle + (0.0800 + 0.0637i)|1101\rangle + \\
&(-0.0928 - 0.0770i)|1110\rangle + (0.1401 + 0.6194i)|1111\rangle, \\
&(0.2668 - 0.0316i)|0000\rangle + (-0.0455 - 0.0103i)|0001\rangle + \\
&(0.0531 + 0.0162i)|0010\rangle + (0.0040 - 0.0177i)|0011\rangle + \\
&(-0.0577 - 0.0125i)|0100\rangle + (0.3753 + 0.0163i)|0101\rangle + \\
&(0.0963 - 0.5096i)|0110\rangle + (0.0065 - 0.0649i)|0111\rangle + \\
&(-0.0587 - 0.0141i)|1000\rangle + (0.0725 + 0.5210i)|1001\rangle + \\
&(-0.3699 - 0.0144i)|1010\rangle + (-0.0127 - 0.0627i)|1011\rangle + \\
&(0.0019 + 0.0135i)|1100\rangle + (0.0118 + 0.0600i)|1101\rangle + \\
&(0.0041 - 0.0677i)|1110\rangle + (0.2469 - 0.1159i)|1111\rangle\}
\end{aligned} \quad . \tag{5.11}$$

5.3 Depolarising channel

The depolarising channel models the full erasure of the information contained in a qubit. It involves flips around all three axes of the Bloch sphere, i.e. bit-flip, phase-flip and bit-phase-flip. As mentioned before, the bit-flip is represented by the first Pauli matrix σ_x. Let us now introduce the other two Pauli matrices,

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{5.12}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{5.13}$$

It is easy to see that the third Pauli matrix generates a phase-flip, since it maps

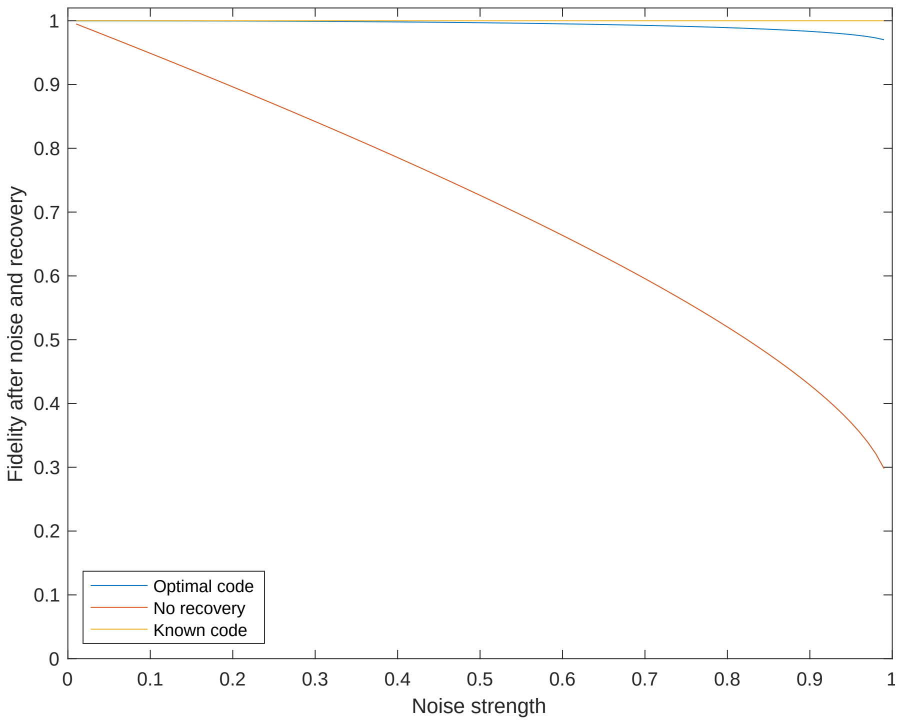

Figure 5.7: Fidelity of $\mathcal{A}(|0_L\rangle\langle 0_L|)$ under the $(2^4, 2^1)$ codes and the amplitude damping channel, optimised with the steepest descent algorithm and 100 starting points

$|0\rangle \mapsto |0\rangle$ and $|1\rangle \mapsto -|1\rangle$, i.e. if one has as initial state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the final state would have an additional phase of π between $|0\rangle$ and $|1\rangle$, as $\sigma_z|\psi\rangle = \alpha|0\rangle + e^{i\pi}\beta|1\rangle$. As for σ_y, it induces a bit-phase-flip, since it maps $|0\rangle \mapsto i|1\rangle$, and $|1\rangle \mapsto -i|0\rangle$. The global phase of i does not have any effect, since it is cancelled by the adjoint in any probability or expectation computation ($-ii = 1$).

For a single qubit this channel has the following Kraus representation,

$$\begin{aligned}\mathcal{N} \sim \{&\sqrt{1-p}\mathbb{1}, \\ &\sqrt{\frac{p}{3}}\sigma_x, \sqrt{\frac{p}{3}}\sigma_y, \sqrt{\frac{p}{3}}\sigma_z\}.\end{aligned} \tag{5.14}$$

As example, consider $p = 3/4$ and a general qubit pure state $\rho = |\alpha|^2|0\rangle\langle 0| + \alpha\overline{\beta}|0\rangle\langle 1| + \overline{\alpha}\beta|1\rangle\langle 0| + |\beta|^2|1\rangle\langle 1|$, then,

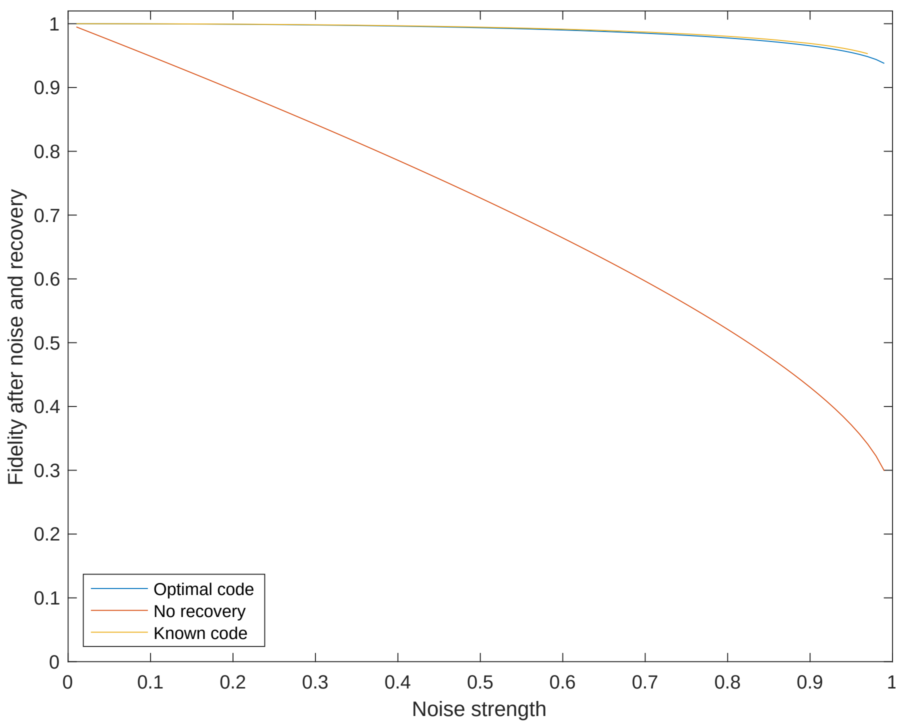

Figure 5.8: Fidelity of $\mathbb{1} \otimes \mathcal{A}((|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|))$ under the $(2^4, 2^1)$ code and the amplitude damping channel, optimised with the steepest descent algorithm using 100 initial points

$$\begin{aligned}
\mathcal{N}(\rho) =& \frac{1}{4}(|\alpha|^2|0\rangle\langle 0| + \alpha\overline{\beta}|0\rangle\langle 1| + \overline{\alpha}\beta|1\rangle\langle 0| + |\beta|^2|1\rangle\langle 1|) + \\
& \frac{1}{4}(|\alpha|^2|1\rangle\langle 1| + \alpha\overline{\beta}|1\rangle\langle 0| + \overline{\alpha}\beta|0\rangle\langle 1| + |\beta|^2|0\rangle\langle 0|) + \\
& \frac{1}{4}(|\alpha|^2|1\rangle\langle 1| - \alpha\overline{\beta}|1\rangle\langle 0| - \overline{\alpha}\beta|0\rangle\langle 1| + |\beta|^2|0\rangle\langle 0|) + . \\
& \frac{1}{4}(|\alpha|^2|0\rangle\langle 0| - \alpha\overline{\beta}|0\rangle\langle 1| - \overline{\alpha}\beta|1\rangle\langle 0| + |\beta|^2|1\rangle\langle 1|) = \\
=& \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \frac{1}{2}\mathbb{1}
\end{aligned} \tag{5.15}$$

Therefore, under this setting, the depolarising channel maps every initial pure state onto the fully mixed state $\mathbb{1}/2$, removing entirely any information stored in the initial state. Another representation of this channel is

$$\mathcal{N}(\rho) = (1-q)\rho + \frac{q}{2}\mathbb{1}, \tag{5.16}$$

where it has been made explicit the fact that it mixes the input state with the fully mixed state.

For a system of five qubits, the following Kraus representation is used,

$$\begin{aligned}\mathcal{N} \sim \{&\sqrt{1-p}\mathbb{1}^{\otimes 5},\\ &\sqrt{\frac{p}{15}}\sigma_x \otimes \mathbb{1}^{\otimes 4}, \sqrt{\frac{p}{15}}\sigma_y \otimes \mathbb{1}^{\otimes 4}, \sqrt{\frac{p}{15}}\sigma_z \otimes \mathbb{1}^{\otimes 4},\\ &\sqrt{\frac{p}{15}}\mathbb{1} \otimes \sigma_x \otimes \mathbb{1}^{\otimes 3}, \sqrt{\frac{p}{15}}\mathbb{1} \otimes \sigma_y \otimes \mathbb{1}^{\otimes 3}, \sqrt{\frac{p}{15}}\mathbb{1} \otimes \sigma_z \otimes \mathbb{1}^{\otimes 3},\\ &\sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 2} \otimes \sigma_x \otimes \mathbb{1}^{\otimes 2}, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 2} \otimes \sigma_y \otimes \mathbb{1}^{\otimes 2}, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 2} \otimes \sigma_z \otimes \mathbb{1}^{\otimes 2},\\ &\sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 3} \otimes \sigma_x \otimes \mathbb{1}, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 3} \otimes \sigma_y \otimes \mathbb{1}, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 3} \otimes \sigma_z \otimes \mathbb{1},\\ &\sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 4} \otimes \sigma_x, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 4} \otimes \sigma_y, \sqrt{\frac{p}{15}}\mathbb{1}^{\otimes 4} \otimes \sigma_z\}\end{aligned} . \tag{5.17}$$

The expression for five qubits is given, because a perfect five qubit code is known for this noise [3, 21]. This code has the following basis,

$$\begin{aligned}\{|0_L\rangle, |1_L\rangle\} = \{&\frac{1}{4}(|00000\rangle - |00011\rangle + |00101\rangle - |00110\rangle + |01001\rangle + |01010\rangle\\ &- |01100\rangle - |01111\rangle - |10001\rangle + |10010\rangle + |10100\rangle - |10111\rangle\\ &- |11000\rangle - |11011\rangle - |11101\rangle - |11110\rangle),\\ &\frac{1}{4}(-|00001\rangle - |00010\rangle - |00100\rangle - |00111\rangle - |01000\rangle + |01011\rangle\\ &+ |01101\rangle - |01110\rangle - |10000\rangle - |10011\rangle + |10101\rangle + |10110\rangle\\ &- |11001\rangle + |11010\rangle - |11100\rangle + |11111\rangle)\}\end{aligned} . \tag{5.18}$$

This code is another stabiliser code and its generators are given in table 5.2.

g_1	Y	Y	Z	I	Z
g_2	X	I	X	Z	Z
g_3	X	Z	Z	X	I
g_4	Y	Z	I	Z	Y

Table 5.2: Generators of the known $(2^5, 2^1)$ perfect code

Figures 5.9, 5.10 show the performance of the optimised code, compared to the known five qubit code and to a case without recovery. The basis elements of the optimised code is

$$\begin{aligned}
|0_L\rangle =&(-0.1456+0.1749i)|00000\rangle+(-0.1717-0.0812i)|00001\rangle+\\
&(0.0905-0.0352i)|00010\rangle+(-0.0320+0.0350i)|00011\rangle+\\
&(0.1609+0.1728i)|00100\rangle+(0.1309-0.0932i)|00101\rangle+\\
&(0.0074+0.0380i)|00110\rangle+(-0.2595+0.0072i)|00111\rangle+\\
&(-0.2086-0.0672i)|01000\rangle+(-0.0204+0.0629i)|01001\rangle+\\
&(0.0128-0.1244i)|01010\rangle+(-0.2869+0.0094i)|01011\rangle+\\
&(0.0656-0.0516i)|01100\rangle+(-0.0137+0.1449i)|01101\rangle+\\
&(0.2145-0.1220i)|01110\rangle+(0.0615+0.0820i)|01111\rangle+\\
&(0.1521-0.0685i)|10000\rangle+(-0.1711+0.0146i)|10001\rangle+\\
&(-0.0735-0.1011i)|10010\rangle+(-0.2424-0.1370i)|10011\rangle+\\
&(-0.0725+0.0849i)|10100\rangle+(0.0572-0.0867i)|10101\rangle+\\
&(-0.2394-0.1354i)|10110\rangle+(0.0212-0.0144i)|10111\rangle+\\
&(-0.1053-0.1596i)|11000\rangle+(0.1026-0.0729i)|11001\rangle+\\
&(-0.0254+0.2108i)|11010\rangle+(-0.0242-0.0378i)|11011\rangle+\\
&(0.0646-0.0801i)|11100\rangle+(0.0749-0.3175i)|11101\rangle+\\
&(0.0341-0.1582i)|11110\rangle+(-0.0145+0.0839i)|11111\rangle
\end{aligned}, \tag{5.19}$$

$$\begin{aligned}|1_L\rangle\} =&(-0.0833-0.0177i)|00000\rangle+(-0.1596-0.0270i)|00001\rangle+\\&(-0.3227-0.0481i)|00010\rangle+(0.0984-0.0303i)|00011\rangle+\\&(-0.0261-0.0365i)|00100\rangle+(-0.2052-0.0545i)|00101\rangle+\\&(0.1058-0.0683i)|00110\rangle+(-0.1092-0.1570i)|00111\rangle+\\&(-0.0213+0.0144i)|01000\rangle+(0.0370+0.2725i)|01001\rangle+\\&(0.1017-0.0209i)|01010\rangle+(0.1058-0.0359i)|01011\rangle+\\&(0.0374+0.2759i)|01100\rangle+(-0.0667-0.1057i)|01101\rangle+\\&(0.0769-0.1535i)|01110\rangle+(0.1200-0.1158i)|01111\rangle+\\&(0.0534+0.0875i)|10000\rangle+(0.1928-0.1540i)|10001\rangle+\\&(-0.1396-0.0410i)|10010\rangle+(-0.0722+0.0418i)|10011\rangle+\\&(-0.1151+0.2630i)|10100\rangle+(-0.1203-0.0343i)|10101\rangle+\\&(0.0660+0.0043i)|10110\rangle+(-0.0150+0.2187i)|10111\rangle+\\&(-0.1029+0.2383i)|11000\rangle+(0.0326+0.0209i)|11001\rangle+\\&(-0.1351+0.0870i)|11010\rangle+(-0.1009-0.2135i)|11011\rangle+\\&(0.0444-0.0168i)|11100\rangle+(0.0662-0.0710i)|11101\rangle+\\&(0.0118+0.1896i)|11110\rangle+(0.2164-0.0704i)|11111\rangle\end{aligned}\quad. \tag{5.20}$$

Let us now consider theorem 3.1.1. This theorem states that if a code $\mathcal{C}$ can correct a noise $\mathcal{N}$ with Kraus operators $\{N_i\}$, then it can correct any noise $\mathcal{N}'$ with Kraus operators $\{\sum_j \lambda_{ij}N_j\}$. Since the Pauli matrices, together with the identity form a bases of $\mathbb{C}^{2\times 2}$, this means that a code able to correct the depolarising channel is also able to correct any other single qubit noise. In order to test this claim, let us consider the five qubit amplitude damping channel with the following representation,

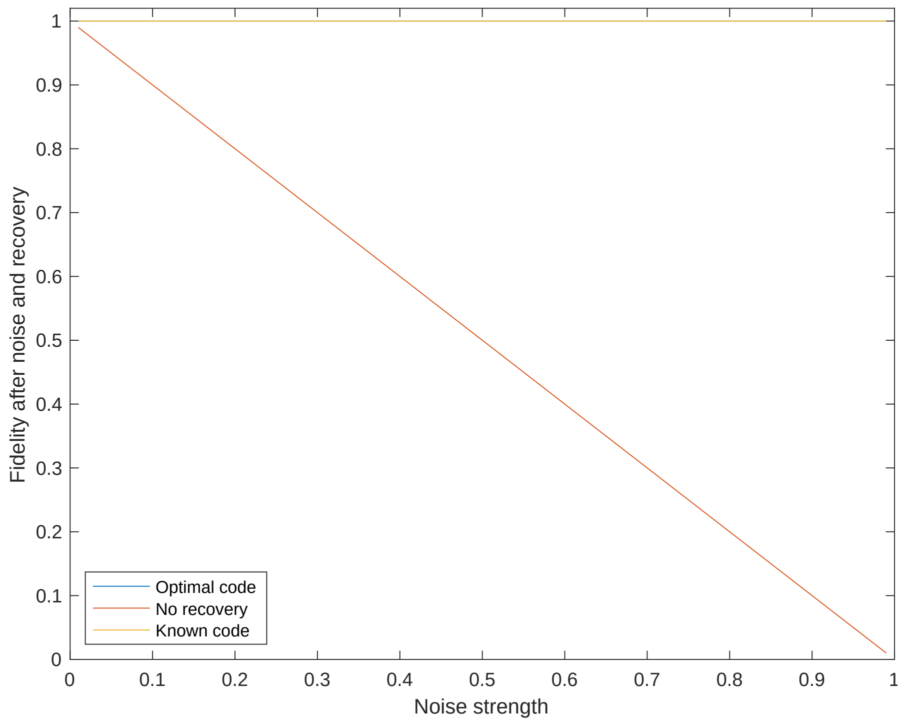

Figure 5.9: Fidelity of $\mathcal{A}(|0_L\rangle\langle 0_L|)$ under the $(2^5, 2^1)$ code and the depolarising channel

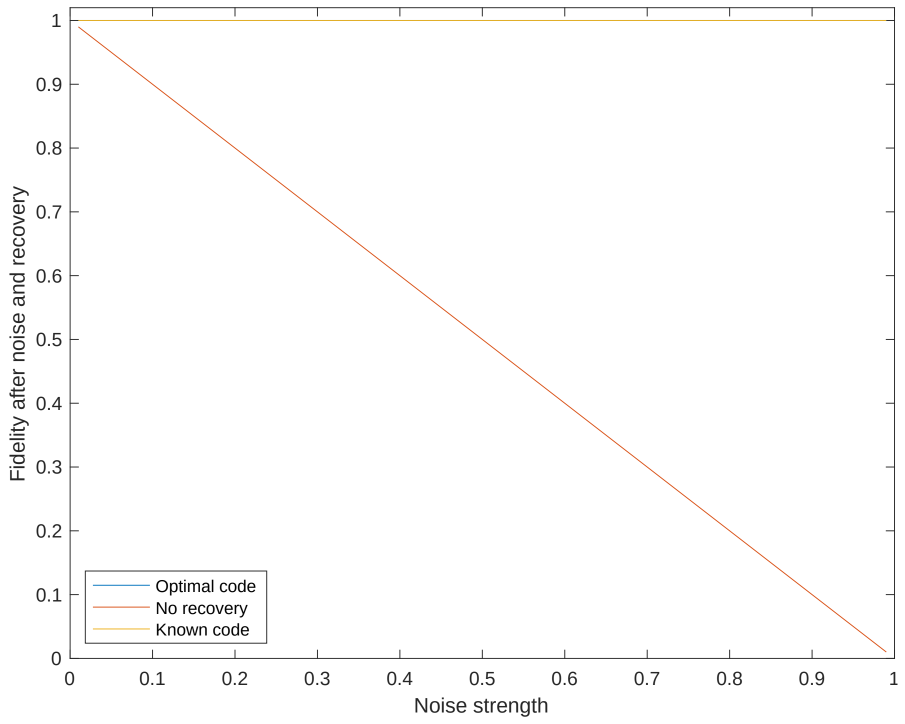

Figure 5.10: Fidelity of $\mathbb{1} \otimes \mathcal{A}((|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|))$ under the $(2^5, 2^1)$ code and the depolarising channel

$$
\begin{aligned}
\mathcal{N} \sim \{&\sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 5}, \\
&\sqrt{\frac{1}{5}}E_0 \otimes \mathbb{1}^{\otimes 4}, \sqrt{\frac{1}{5}}E_1 \otimes \mathbb{1}^{\otimes 4}, \\
&\sqrt{\frac{1}{5}}\mathbb{1} \otimes E_0 \otimes \mathbb{1}^{\otimes 3}, \sqrt{\frac{1}{5}}\mathbb{1} \otimes E_1 \otimes \mathbb{1}^{\otimes 3}, \\
&\sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 2} \otimes E_0 \otimes \mathbb{1}^{\otimes 2}, \sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 2} \otimes E_1 \otimes \mathbb{1}^{\otimes 2}, \\
&\sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 3} \otimes E_0 \otimes \mathbb{1}, \sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 3} \otimes E_1 \otimes \mathbb{1}, \\
&\sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 4} \otimes E_0, \sqrt{\frac{1}{5}}\mathbb{1}^{\otimes 4} \otimes E_1\}
\end{aligned}
\quad . \tag{5.21}
$$

Figures 5.11, 5.12 show the performance of the same two codes under the amplitude damping channel. As it can be seen, it indeed perfectly corrects this noise channel as well.

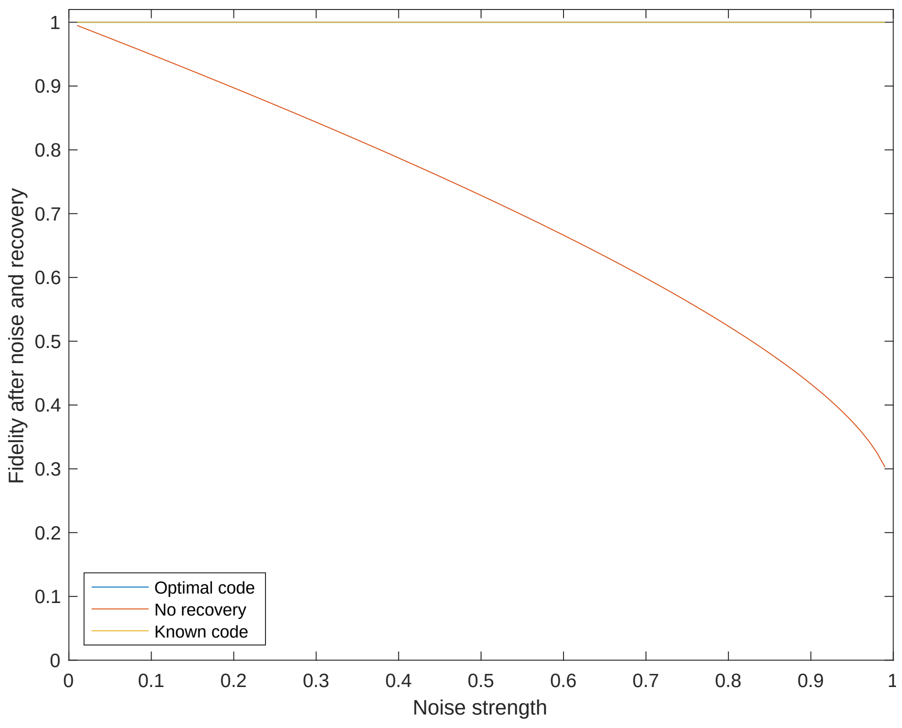

Figure 5.11: Fidelity of $\mathcal{A}(|0_L\rangle\langle 0_L|)$ under the $(2^5, 2^1)$ code and the amplitude damping channel

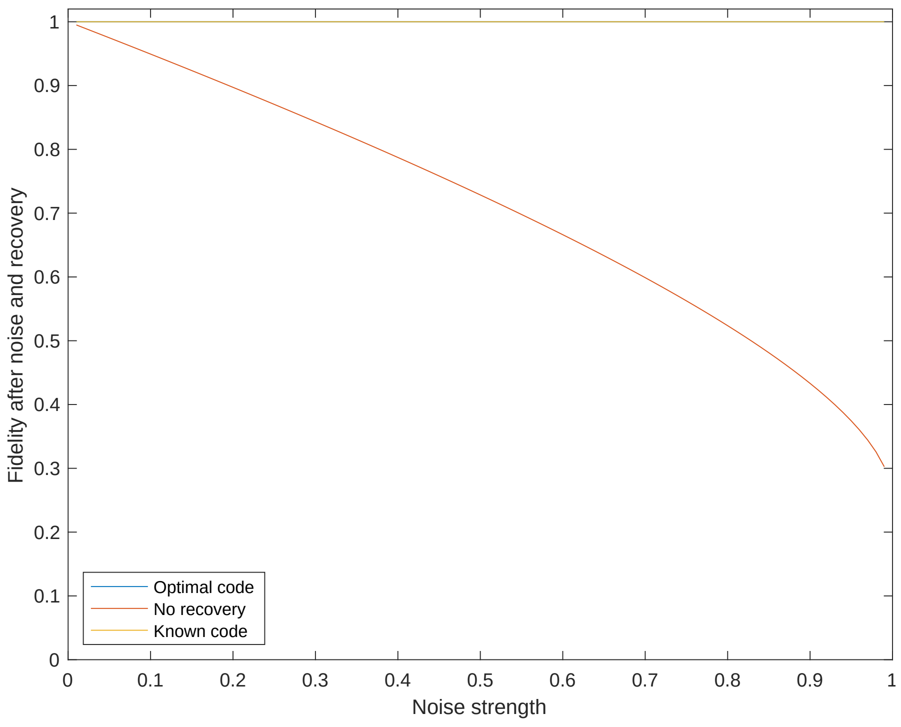

Figure 5.12: Fidelity of $\mathbb{1} \otimes \mathcal{A}((|0_L\rangle \otimes |0_L\rangle + |1_L\rangle \otimes |1_L\rangle)(\langle 0_L| \otimes \langle 0_L| + \langle 1_L| \otimes \langle 1_L|))$ under the $(2^5, 2^1)$ code and the amplitude damping channel

6 Conclusions and Future Works

As shown in the previous chapter, when a perfectly correctable codes exist, the implemented algorithms are able to find them. On the other hand, when only approximate codes exist, the algorithms are still able to find codes with performance similar to that of the codes known in the literature. Furthermore the cost function $J_d(\Pi)$ can be used to benchmark different codes, and aid in the design process of an engineered quantum system.

Even though the codes obtained with these algorithms do not necessarily have the kind of symmetries that would be desired for a physical implementation, they could be approximated by means of the Schmidt operator decomposition. An interesting line of research for future work would be exploring how the performance of the codes would be affected by such an approximation.

The methods developed in this work can also be applied to theoretical models. They can also be used to empirically explore the properties of different QEC protocols or different noise channels. For instance, to study the degeneracy of optimal codes, or how perturbations of a noise model affect the correctability, as done in the previous chapter.

Extension of the implemented algorithms for finding codes are possible. For instance, the implementation of conjugate descent methods [30, 39] or the relaxation to convex versions, using a combination of duality theory, barrier functions and geometric reformulation of the problem, in the effort of producing faster and more accurate algorithms.

Finally, since all of the above assumes knowledge of a model of the system and the noise, a possible line of research would be through the study of system identification theory, in order to develop model learning algorithms for quantum systems and for the noises present in the system. That is to include system identification procedures within the control strategies and to integrate them with the present work in a sort of data-driven information-protection design. Having such a design pipeline may prove to be crucial as engineered quantum systems continue to scale up in size and complexity.

A

Gradient computation

In order to use gradient methods to obtain a local optima of our optimisation problem, we need to compute the gradient of the cost function. First, let us define the gradient of $J_d(U)$ over $V_d(\mathbb{C}^n)$. Since we consider a complex function $J_d : \mathbb{C}^{n\times d} \supset V_d(\mathbb{C}^n) \longrightarrow [0,\infty)$, the Eucledian gradient of J_d becomes

$$\nabla_U J_d(U) := \frac{\partial}{\partial U_{Re}} J_d(U) + i \frac{\partial}{\partial U_{Im}} J_d(U) \in \mathbb{C}^{n\times d}, \tag{A.1}$$

where $U_{Re} := (U+\overline{U})/2$ and $U_{Im} := (U-\overline{U})/(2i)$ are the real and imaginary matrices of U, respectively. The gradient at $U \in V_d(\mathbb{C}^n)$ is calculated as

$$grad J_d(U) = \nabla_U J_d(U) - U(\nabla_U J_d(U))^\dagger U. \tag{A.2}$$

It is easy to verify that $grad J_d(U) \in \mathcal{T}_U V_d(\mathbb{C}^n)$.

Remark. For complex cost functions, the complex derivative does not make sense in general. For example, let us consider $J(z) := |z-\alpha|^2$ where $\alpha \in \mathbb{C}$. In this case, the complex derivative does not exist. However, for $z = x + iy$, the derivative over the real and imaginary axes exist and are calculated as

$$\frac{\partial J(z)}{\partial x} = 2(x - Re(\alpha)), \ \frac{\partial J(z)}{\partial y} = 2(y - Re(\alpha)).$$

Then,

$$\nabla J(z) = 2(z-\alpha),$$

and this is the natural gradient for the cost function.

Since $tr(A \otimes B) = tr(A)tr(B)$ and $tr(A) = tr(A^\top)$ for any square complex matrices A and B, the cost function (3.51)

$$\begin{aligned} J_d(U) &= \sum_{k,l=1}^{m} tr\left((\Pi_U^\top \otimes \Pi_U)N_k^\top \mathcal{N}(\Pi_U)^{-\top/2}\overline{N_l} \otimes N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}N_k\right) \\ &= \sum_{k,l=1}^{m} tr\left(\Pi_U^\top N_k^\top \mathcal{N}(\Pi_U)^{-\top/2}\overline{N_l}\right) tr\left(\Pi_U N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}N_k\right) \\ &= \sum_{k,l=1}^{m} tr\left(N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}N_k \Pi_U\right) tr\left(\Pi_U N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}N_k\right) \\ &= \sum_{k,l=1}^{m} tr\left(\Pi_U N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}N_k\right)^2 \end{aligned} . \tag{A.3}$$

Now we need to obtain the Eucledian gradient of J_d at any $U_0 \in V_d(\mathbb{C}^n)$.

A.1 Useful gradients

The following are gradients that are going to appear due to the chain rule, when computing the gradient of the cost function,

$$\nabla_U tr(C_1(\Pi_{U_0})\Pi_U)|_{U=U_0} \text{ and } \nabla_U tr(C_2(\Pi_{U_0})\mathcal{N}(\Pi_U)^{-1/2})|_{U=U_0}, \tag{A.4}$$

where $C_i(U_0) = C_i(U_0)^\dagger$.

Lemma A.1.1. *Let $A \in \mathbb{C}^{n\times n}$ be a given Hermitian matrix. then,*

$$\nabla_U tr(A\Pi_U) = 2AU. \tag{A.5}$$

Proof. Since $U = U_{Re} + iU_{Im}$,

$$\begin{aligned} tr(A\Pi_U) &= tr(U^\dagger AU) \\ &= tr((U_{Re} - iU_{Im})^\top A(U_{Re} + iU_{Im})) \\ &= tr(U_{Re}^\top AU_{Re}) - itr(U_{Im}^\top AU_{Re}) + itr(U_{Re}^\top AU_{Im}) + tr(U_{Im}^\top AU_{Im}) \end{aligned} . \tag{A.6}$$

Therefore,

$$\frac{\partial}{\partial U_{Re}} tr(A\Pi_U) = (A + A^\top)U_{Re} + i(A - A^\top)U_{Im} \in \mathbb{R}^{n\times d}, \tag{A.7}$$

$$\frac{\partial}{\partial U_{Im}} tr(A\Pi_U) = (A + A^\top)U_{Im} - i(A - A^\top)U_{Re} \in \mathbb{R}^{n\times d}, \tag{A.8}$$

and

$$\nabla_U tr(A\Pi_U) = \frac{\partial}{\partial U_{Re}} tr(A\Pi_U) + i\frac{\partial}{\partial U_{Im}} tr(A\Pi_U) = 2AU. \tag{A.9}$$

□

Lemma A.1.2. *Let $A \in \mathbb{C}^{n\times n}$ be a given Hermitian matrix. Then,*

$$\nabla_U tr(A\mathcal{N}(\Pi_U)^{-1/2}) = 2\sum_{j=1}^{m}\sum_{i=1}^{3} N_j^\dagger \mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(C_i)^\dagger N_j)U, \tag{A.10}$$

where

$$C_1 = -\mathcal{N}(\Pi_{U_0})^{-1/2}A\mathcal{N}(\Pi_{U_0})^{-1/2}, \tag{A.11}$$

$$C_2 = \mathcal{N}(\Pi_{U_0})^{-1}A(\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2}\mathcal{N}(\Pi_{U_0})^{1/2}), \tag{A.12}$$

$$C_3 = (\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2}\mathcal{N}(\Pi_{U_0})^{1/2})A\mathcal{N}(\Pi_{U_0})^{-1}, \tag{A.13}$$

$$\hat{C} = \mathcal{N}(\Pi_U)^{\top/2} \otimes \mathbb{1}_n + \mathbb{1}_n \otimes \mathcal{N}(\Pi_U)^{1/2}. \tag{A.14}$$

Proof. For any non-zero matrix $X = (X_{ij}) \in \mathbb{C}^{n\times n}$ and $A \in \mathbb{C}^{n\times n}$, derivative of the Moore-Penrose pseudo inverse is given as

$$\frac{\partial}{\partial X_{ij}}X^{-1} = -X^{-1}\frac{\partial X}{\partial X_{ij}}X^{-1} + (\mathbb{1}_n - X^{-1}X)\frac{\partial X^\dagger}{\partial X_{ij}}(X^{-1})^\dagger X^{-1} + X^{-1}(X^{-1})^\dagger\frac{\partial X^\dagger}{\partial X_{ij}}(\mathbb{1}_n - XX^{-1}), \tag{A.15}$$

where the derivative is the Eucledian derivative [10, 16]. Therefore, we only consider the derivative of the underlined parts of the following equation,

$$\begin{aligned}
&\nabla_U tr(A\mathcal{N}(\Pi_U)^{-1/2})|_{U=U_0} \\
&= -\nabla_U tr(A\mathcal{N}(\Pi_{U_0})^{-1/2}\underline{\mathcal{N}(\Pi_U)^{1/2}}\mathcal{N}(\Pi_{U_0})^{-1/2})|_{U=U_0} \\
&\quad + \nabla_U tr(A(\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2}\mathcal{N}(\Pi_{U_0})^{1/2})\underline{\mathcal{N}(\Pi_U)^{1/2}}\mathcal{N}(\Pi_{U_0})^{-1})|_{U=U_0} \\
&\quad + \nabla_U tr(A\mathcal{N}(\Pi_{U_0})^{-1}\underline{\mathcal{N}(\Pi_U)^{1/2}}(\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2}\mathcal{N}(\Pi_{U_0})^{1/2}))|_{U=U_0}
\end{aligned}. \tag{A.16}$$

Note that if $\mathcal{N}(\Pi_U)$ is invertible, i.e., positive definite, the above equation becomes

$$\nabla_U tr(A\mathcal{N}(\Pi_U)^{1/2})|_{U=U_0} = -\nabla_U tr(A\mathcal{N}(\Pi_{U_0})^{-1/2}\mathcal{N}(\Pi_U)^{1/2}\mathcal{N}(\Pi_{U_0})^{-1/2})|_{U=U_0}. \tag{A.17}$$

Next, we establish the derivative of $\mathcal{N}(\Pi_U)^{1/2}$ in A.16. Since $\mathcal{N}(\Pi_U) = \mathcal{N}(\Pi_U)^{1/2}\mathcal{N}(\Pi_U)^{1/2}$, we have the following Sylvester equation,

$$\frac{\partial}{\partial U_{ij}}\mathcal{N}(\Pi_U) = \frac{\partial \mathcal{N}(\Pi_U)^{1/2}}{\partial U_{ij}}\mathcal{N}(\Pi_U)^{1/2} + \mathcal{N}(\Pi_U)^{1/2}\frac{\partial \mathcal{N}^{1/2}}{\partial U_{ij}}. \tag{A.18}$$

By using vectorisation,

$$\mathrm{vec}(\frac{\partial}{\partial U_{ij}}\mathcal{N}(\Pi_U)) = (\mathcal{N}(\Pi_U)^{\top/2} \otimes \mathbb{1}_n + \mathbb{1}_n \otimes \mathcal{N}(\Pi_U)^{1/2})\mathrm{vec}(\frac{\partial \mathcal{N}(\Pi_U)^{1/2}}{\partial U_{ij}}), \tag{A.19}$$

$$\hat{C} := \mathcal{N}(\Pi_U)^{\top/2} \otimes \mathbb{1}_n + \mathbb{1}_n \otimes \mathcal{N}(\Pi_U)^{1/2}.$$

For $C \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$, the general solution of $Cx = b$ is given by $x = C^{-1}b + (\mathbb{1}_n - C^{-1}C)d$ if the solution exists, where $d \in \mathbb{C}^n$ is an arbitrary vector (we consider the Moore-Penrose pseudo-inverse if $det(C) = 0$). Hence,

$$\frac{\partial \mathcal{N}(\Pi_U)^{1/2}}{\partial U_{ij}} = \mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(\frac{\partial}{\partial U_{ij}}\mathcal{N}(\Pi_U)) + (\mathbb{1}_{n^2} - \hat{C}^{-1}\hat{C})z_{ij}), z_{ij} \in \mathbb{C}^{n^2}, \tag{A.20}$$

where $z_{ij} \in \mathbb{C}^{n^2}$ is arbitrarily chosen, Now we analyze the effect of z_{ij}. Note that since $\mathcal{N}(\Pi_{U_0})$ is Hermitian and non-negative, there exists a unitary matrix $V \in \mathbb{C}^{n\times n}$ such that

$$\mathcal{N}(\Pi_{U_0}) = V \begin{bmatrix} \Lambda & \\ & O_p \end{bmatrix} V^\dagger,$$

where $\Lambda \in \mathbb{R}^{(n-p)\times(n-p)}$ is a diagonal matrix with strictly positive diagonal element and $1 \leq p < n$ is the rank of the kernel of $\mathcal{N}(\Pi_{U_0})$ (here, we do not consider

$p = 0$ or $p = n$). Hence, we have

$$\hat{C} = V^* \otimes V \left(\begin{bmatrix} \sqrt{\Lambda} & \\ & O_p \end{bmatrix} \otimes I_n + I_n \otimes \begin{bmatrix} \sqrt{\Lambda} & \\ & O_p \end{bmatrix} \right) V^\top \otimes V^\dagger \tag{A.21}$$

and

$$\begin{aligned} I_{n^2} - \hat{C}^{-1}\hat{C} &= V^* \otimes V \begin{bmatrix} O_{n(n-p)} & & & & & \\ & O_{n-p} & & & & \\ & & I_p & & & \\ & & & \ddots & & \\ & & & & O_{n-p} & \\ & & & & & I_p \end{bmatrix} V^\top \otimes V^\dagger \\ &= V^* \otimes V \left(\begin{bmatrix} O_{n-p} & \\ & I_p \end{bmatrix} \otimes \begin{bmatrix} O_{n-p} & \\ & I_p \end{bmatrix} \right) V^\top \otimes V^\dagger. \end{aligned}$$

From (A.20), $\mathcal{N}(\Pi_U)^{-1/2}$ is multiplied from, at least, one of the both sides of the derivative of $\mathcal{N}^{-1/2}$, so for any $X \in \mathbb{C}^{n\times n}$,

$$(\mathcal{N}(\Pi_U)^{-1/2} \otimes X)(I_{n^2} - \hat{C}^{-1}\hat{C}) = (X \otimes \mathcal{N}(\Pi_U)^{-1/2})(I_{n^2} - \hat{C}^{-1}\hat{C}) = 0.$$

Therefore, without loss of generality, we can put $z_{ij} = 0$.

For $C = C^\dagger \in \mathbb{C}^{n\times n}$, by using $tr(A^\dagger B) = \mathrm{vec}(A)^\dagger \mathrm{vec}(B)$ and the fact that $\hat{C}^\dagger = \hat{C}$,

$$\begin{aligned} \nabla_U tr(C\mathcal{N}(\Pi_U)^{1/2}) &= \nabla_U \mathrm{vec}(C)^\dagger (\hat{C}^{-1}\mathrm{vec}(\mathcal{N}(\Pi_U))) \\ &= \nabla_U (\hat{C}^{-1}\mathrm{vec}(C))^\dagger \mathrm{vec}(\mathcal{N}(\Pi_U)) \\ &= \nabla_U tr(\mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(C))^\dagger \mathcal{N}(\Pi_U)) \\ &= \nabla_U \sum_{k=1}^{n} tr(N_k^\dagger \mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(C))^\dagger N_k \Pi_U) \\ &= 2\left(\sum_{k=1}^{n} N_k^\dagger \mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(C))^\dagger N_k\right) U \end{aligned} . \tag{A.22}$$

From (A.16) and (A.22), we obtain the Eucledian gradient of $tr(A\mathcal{N}(\Pi_U))^{-1/2}$.

□

A.2 Gradient of the cost function

We can now proceed to compute the Eucledian gradient of the cost function

$$\begin{aligned}\nabla_U J_d(U)|_{U=U_0} =& \nabla_U \sum_{k,l=1}^{m} tr\left(\Pi_U N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2} N_k\right)^2 |_{U=U_0}\\ =& \sum_{k,l=1}^{m} \left(2tr\left(\Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k\right)\right.\\ &(\nabla_U tr\left(N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k \Pi_U\right)|_{U=U_0} +\\ &\left.\nabla_U tr\left(N_k \Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}\right)|_{U=U_0})\right)\end{aligned} . \tag{A.23}$$

By applying lemma A.1.1

$$\begin{aligned}\nabla_U J_d(U)|_{U=U_0} =& \sum_{k,l=1}^{m} \left(2tr\left(\Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k\right)\right.\\ &(2N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k U +\\ &\left.\nabla_U tr\left(N_k \Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_U)^{-1/2}\right)|_{U=U_0})\right)\end{aligned} . \tag{A.24}$$

Finally, by applying lemma A.1.2

$$\begin{aligned}\nabla_U J_d(U)|_{U=U_0} =& 4\sum_{j,k,l=1}^{m} \left(tr\left(\Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k\right)\right.\\ &\left.(N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2} N_k U + \sum_{i=1}^{3}(N_j^\dagger \mathrm{vec}^{-1}(\hat{C}^{-1}\mathrm{vec}(C_i)^\dagger N_j)U)\right)\end{aligned} , \tag{A.25}$$

where

$$C_1 = -\mathcal{N}(\Pi_{U_0})^{-1/2} N_k \Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1/2}, \tag{A.26}$$

$$C_2 = \mathcal{N}(\Pi_{U_0})^{-1} N_k \Pi_{U_0} N_l^\dagger (\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2} \mathcal{N}(\Pi_{U_0})^{1/2}), \tag{A.27}$$

$$C_3 = (\mathbb{1}_n - \mathcal{N}(\Pi_{U_0})^{-1/2} \mathcal{N}(\Pi_{U_0})^{1/2}) N_k \Pi_{U_0} N_l^\dagger \mathcal{N}(\Pi_{U_0})^{-1}, \tag{A.28}$$

$$\hat{C} = \mathcal{N}(\Pi_U)^{\top/2} \otimes \mathbb{1}_n + \mathbb{1}_n \otimes \mathcal{N}(\Pi_U)^{1/2}. \tag{A.29}$$

B MATLAB code

```matlab
function res = armijo(U, X, N, grad, norm_grad, cost_U, d, t, c1)
% This function returns a boolean indicating whether Armijo's
    condition
% is satisfied or not
res = cost_function(ret_exp(U, X, t), N) > cost_U - c1 * t *
    norm_grad^2;
end
```

Code B.1: Armijo condition

```matlab
function t = backtracking(U, X, N, grad, norm_grad, cost_U, d, t0, c1
    , tau, min_bt)
% This function performs the backtracking algorithm for step-size
% computation, used in line search based optimisation algorithms
t = t0;
% grad = grad_cost_function(U, N);
while armijo(U, X, N, grad, norm_grad, cost_U, d, t, c1)
    t = t * tau;
    if t < min_bt
        disp('Armijo not satisfied')
        break
    end
end

end
```

Code B.2: Backtracking

```matlab
function res = boltz_weight(U, N, d, beta)
% This function computes the Boltzmann weight used in the consensus
    method
res = exp(-beta * (cost_function(U, N) + d^2));
end
```

Code B.3: Boltzmann weight

```matlab
function res = canon_inner(U, X, Y)
% Canonincal Riemannian inner product in the complex valued Stifel
    manifold
n = size(U, 1);
res = trace(X' * (eye(n) - U*U'/2) * Y);
end
```

Code B.4: Canonical inner product

```matlab
function res = consensus(U0_cell, N, h, T, beta, lambda, sigma)
% Consensus method for global optimisation in the complex valued
% Stiefel manifold
U_cell = U0_cell;
n = size(U_cell{1}, 1);
d = size(U_cell{1}, 2);
l = length(U_cell);
m = floor(T/h);

C = (2*n - d - 1)/2;
omega_array = zeros(1, l);
DW = cell(1,l);

for i = 1:m
    % parfor j = 1:l
    for j = 1:l
        omega_array(j) = boltz_weight(U_cell{j}, N, d, beta);
    end

    omega_sum = sum(omega_array);
    omega_max = max(omega_array);
    U_bar = zeros(n,d);
    for j = 1:l
        U_bar = U_bar + ((U_cell{j} * omega_array(j) * omega_max) / (
    omega_sum * omega_max));
    end

    for j = 1:l
        DW{j} = randn(n,d)*sqrt(h/2) + 1i*randn(n,d)*sqrt(h/2);
        DW{j} = tang_proj(U_cell{j}, DW{j});

        U_cell{j} = U_cell{j} + (lambda * tang_proj(U_cell{j}, U_bar)
         - C * sigma^2 * trace((U_cell{j} - U_bar)' * (U_cell{j} - U_bar))
         * U_cell{j} / 2) * h + sigma * sqrt(trace((U_cell{j} - U_bar)' *
        (U_cell{j} - U_bar))) * DW{j};
        [U, ~, V] = svd(U_cell{j});
        U_cell{j} = U * eye(n,d) * V';
    end

    disp(['Iteration ', num2str(i), ' finished. ', 'Min cost: ',
    num2str(-log(min(omega_array))/beta)])
end

res = U_cell;
end
```

Code B.5: Consensus method

```
function res = cost_function(U, N)
% Cost function to optimise quantum codes
    P = U * U';
    res = real(-trace(vec_proj_noise_recovery(P, N)));
end
```

Code B.6: Cost function

```
function res = egrad2grad(U,egrad)
% This function converts the Eucledian gradient into the Riemannian
% gradient in the complex valued Stiefel manifold
res = egrad - U * egrad' * U;
end
```

Code B.7: Eucledian gradient to Riemannian gradient

```
function res = egrad_cost_function(U, N)
% Eucledian gradient of the cost function for quantum code
  optimisation
    P = U * U';
    scrN_sqrt = noise(N, P)^(1/2);
    scrN_inv = pinv(noise(N, P));
    scrN_invsqrt = scrN_inv^(1/2);

    A_hat_invT = pinv(kron(transpose(scrN_sqrt), eye(size(P,1))) +
   kron(eye(size(P,1)), scrN_sqrt));

    res = zeros(size(U));
    for i = 1:length(N)
        for j = 1:length(N)
            for k = 1:length(N)
                for l = 1:length(N)
                    A0 = N{i}' * scrN_invsqrt * N{j} * P;
                    A1 = N{i}' * scrN_invsqrt * N{j};
                    A2 = N{j} * P * N{i}';

                    res = res + 2 * trace(A0) * ( ...
                        gradAux3(U, A1) + ...
                        gradAux2(U, N, A2, scrN_inv, scrN_sqrt,
   scrN_invsqrt, A_hat_invT));
                end
            end
        end
    end
    res = -res;
end
```

Code B.8: Eucledian gradient of the cost function

```
function res = gradAux1(U, N, C, A_hat_invT)
% Auxiliary function to compute the gradient of the cost function
    res = zeros(size(N{1}));
    for i = 1:length(N)
        res = res + N{i}' * unvec(A_hat_invT * vec(C))' * N{i};
    end
    res = 2 * res * U;
end
```

Code B.9: Auxilary gradient 1

```
function res = gradAux2(U, N, Ag, scrN_inv, scrN_sqrt, scrN_invsqrt,
    A_hat_invT)
% Auxiliary function to compute the gradient of the cost function
    res = - gradAux1(U, N, scrN_invsqrt * Ag * scrN_invsqrt,
   A_hat_invT);
    res = res + gradAux1(U, N, scrN_inv * Ag * (eye(size(Ag,1)) -
   scrN_invsqrt * scrN_sqrt), A_hat_invT);
    res = res + gradAux1(U, N, (eye(size(Ag,1)) - scrN_sqrt *
   scrN_invsqrt) * Ag * scrN_inv, A_hat_invT);
```

```
end
```

Code B.10: Auxilary gradient 2

```
function res = gradAux3(U, Ag)
% Auxiliary function to compute the gradient of the cost function
    res = 2 * Ag * U;
end
```

Code B.11: Auxilary gradient 3

```
function res = grad_cost_function(U, N)
% Riemannian gradient of the cost function for quantum code
    optimisation
egrad = egrad_cost_function(U, N);
res = egrad2grad(U,egrad);
end
```

Code B.12: Riemannian gradient of the cost function

```
function [U, costs, steps] = grad_descent(U0, t0, N, c1, tau,
    max_iter, min_step, min_bt, min_grad)
% Steepest descent algorithm on the complex valued Stiefel manifold

U = U0;
cost_U = cost_function(U, N);
grad = grad_cost_function(U,N);
[n, d] = size(U);
costs = zeros(1,max_iter);
steps = zeros(1,max_iter);
for k = 1:max_iter
    norm_grad = sqrt(real(canon_inner(U, grad, grad)));
    if norm_grad < min_grad
        break
    end

    t = backtracking(U, -grad, N, grad, norm_grad, cost_U, d, t0, c1,
     tau, min_bt);

    if t < min_step
        break
    else
        U = ret_exp(U, -grad, t);

        U = U / sqrtm(U'*U);
        [Q, ~] = qr(U);
        U = Q(1:n, 1:d);

        grad = grad_cost_function(U,N);
        cost_U = cost_function(U, N);
        costs(k) = cost_U;
        steps(k) = t;
        disp([num2str(k), ' ', num2str(cost_U)])
    end

    if k == max_iter
        disp("Max iter reached")
    end
end

costs(k) = cost_function(U, N);
steps(k) = t;

costs = costs(1,1:k);
steps = steps(1,1:k);
end
```

Code B.13: Steepest descent algorithm

```
function [U_opt, costs] = iddm(U0, t, t0, N, c, rho1, tau, eta,
    min_change, min_grad, max_iter, max_iter_local, min_step, max_step
    , min_bt)
% IDDM algorithm on the complex valued Stiefel manifold

U = U0;
[n, d] = size(U);
costs = zeros(1,max_iter);
U_prev = zeros(n,d);
U_opt = U;
cost_U = cost_function(U, N);
cost_opt = cost_U;

for k = 1:max_iter
    if norm(U_prev - U) < min_change
        break
    else
        sigma = c / ( (k*t(end))^(1/(2*(n-1))) );
        U = sde_solver(U,N,t,sigma);
        U = local_solver(U, t0, N, rho1, tau, eta, min_grad,
    max_iter_local, min_step, max_step, min_bt);

        cost_U = cost_function(U, N);
        if cost_U < cost_opt
            U_opt = U;
            cost_opt = cost_U;
        end
        costs(k) = cost_U;
        save('sanitycheck.mat', 'k', 'U_opt', 'cost_opt')
        disp(['Iteration ', num2str(k), ', cost: ', num2str(costs(k))
    , '. Time: ' datestr(datetime)])
    end

    if k == max_iter
        disp("Max iter reached")
    end
end

if k < max_iter
    costs(k) = cost_function(U, N);
    costs = costs(1,1:k);
end

end
```

Code B.14: IDDM algorithm

```
function res = kraus_check(N)
% Function to check the correctness of a set of Kraus operators
    res = zeros(size(N{1}));
    for i = 1:length(N)
        res = res + N{i}' * N{i};
    end
end
```

Code B.15: Kraus operators correctness check

```
function res = local_solver(U0, t0, N, rho1, tau, eta, min_grad,
    max_iter, min_step, max_step, min_bt)
% Local solver of the IDDM algorithm

U = U0;
egrad = egrad_cost_function(U,N);
[n, d] = size(U);
```

```matlab

Q = 1;
C = cost_function(U, N);
t = t0;
%costs = zeros(1,max_iter);
for k = 1:max_iter
    grad = egrad2grad(U,egrad);
    norm2_grad = real(canon_inner(U, grad, grad));
    if sqrt(norm2_grad) <= min_grad
        %disp("Minimum gradient reached")
        break
    end

    A = egrad * U' - U * egrad';
    %norm2_A = trace(A * A');
    % disp(sqrt(norm2_grad))

    Y = pinv(eye(n) + t*A/2) * (eye(n) - t*A/2) * U;
    %Y = ret_exp(U, -grad, t);

    while cost_function(Y, N) >= C - rho1 * t * norm2_grad
        %disp([cost_function(Y, N), C - rho1 * t * norm2_grad])
        t = tau*t;
        Y = pinv(eye(n) + t*A/2) * (eye(n) - t*A/2) * U;
        %Y = ret_exp(U, -grad, t);
        if t < min_bt
            disp("Armijo-Wolf not satisfied. Returning")
            res = U;
            return
        end
    end

    %U = U / sqrtm(U'*U);
    %[Qnorm, ~] = qr(Y);
    %U_next = Qnorm(1:n, 1:d);
    U_next = Y;
    egrad_next = egrad_cost_function(U_next,N);
    Q_next = eta*Q + 1;
    C_next = (eta * Q * C + cost_function(U_next, N)) / Q_next;
    t_next = max(min(twopoint(U_next, U, egrad_next, egrad), max_step
    ), min_step);

    if t_next == max_step
```

```
        disp("max step used in local solver")
    elseif t_next == min_step
        disp("min step used in local solver")
    end
    %costs(k) = cost_function(U, N);
    U = U_next;
    egrad = egrad_next;
    Q = Q_next;
    C = C_next;
    t = t_next;

    if k == max_iter
        disp("Max iter reached in local solver")
    end
end
%costs(k) = cost_function(U, N);
%plot(costs)
res = U;
end
```

Code B.16: Local solver of the IDDM algorithm

```
function res = noise(N, rho)
% This function applies a noise channel to a quantum state
    res = zeros(size(N{1}));
    for i = 1:length(N)
        res = res + N{i} * rho * N{i}';
    end
end
```

Code B.17: Noise channel

```
function res = noise_recovery(P, N, rho)
% This function applies the composition of noise and Petz recovery to a
% quantum state
    res = recovery(P, N, noise(N, rho));
end
```

Code B.18: Noise and recovery composed operation

```
function res = noise_recovery_ops(P, N)
% This function returns the Kraus operators of the composition of
% noise and Petz recovery
    res = cell(size(N{1}));
    for j = 1:length(N)
        for k = 1:length(N)
            res{j,k} = P * N{j}' * pinv(noise(N,P))^(1/2) * N{k};
        end
    end
end
```

Code B.19: Noise and recovery composed operators

```
function res = proj_noise_recovery(P, N, rho)
% This function applies the composition of projection onto the code,
   noise
% and Petz recovery
    res = recovery(P, N, noise(N, P * rho * P));
end
```

Code B.20: Projection noise recovery operation

```
function res = proj_noise_recovery_ops(P, N)
% This function returns the Kraus operators of the composition of
% projection onto the code, noise and Petz recovery
    res = cell(size(N));
    for j = 1:length(N)
        for k = 1:length(N)
            res{j,k} = P * N{j}' * pinv(noise(N,P))^(1/2) * N{k} * P;
        end
    end
end
```

Code B.21: Projection noise recovery operators

```
function res = recovery(P, N, rho)
%  This function applies the Petz recovery channel
    res = zeros(size(N{1}));
    for i = 1:length(N)
        res = res + P * N{i}' * pinv(noise(N, P))^(1/2) * rho * pinv(
   noise(N, P))^(1/2) * N{i} * P;
    end
end
```

Code B.22: Petz recovery

```
function res = ret_exp(U, X, t)
% This function performs an exponential retraction on the complex
   valued
```

```matlab
% Stiefel manifold
[n, d] = size(U);
A = U' * X;
[Q, R] = qr((eye(n) - U*U') * X);
R = R(1:d, 1:d);
Q = Q(1:n, 1:d);
O = zeros(d,d);

res = [U Q] * expm(t * [A -R'; R O]) * [eye(d); O];
[Q, ~] = qr(res);
res = Q(1:n, 1:d);

end
```

Code B.23: Exponential retraction

```matlab
function res = sde_solver(U0, N, t, sigma)
% This function simulates the Stochastic process of the IDDM
    algorithm
beta = 1 - sqrt(2)/2;
U = U0;
n = size(U, 1);
d = size(U, 2);

k = 1;

for delta = t(2:end) - t(1:end-1)
    G = egrad_cost_function(U, N);
    B = randn(n,d)*sqrt(delta/2) + 1i*randn(n,d)*sqrt(delta/2);
    Z = -delta * G + sigma * (eye(n) - beta * (U * U')) * B;
    A = Z * U' - U * Z';

    U = pinv(eye(n) - A/2) * (eye(n) + A/2) * U;
    k = k+1;
end

res = U;
end
```

Code B.24: SDE solver of the IDDM algorithm

```matlab
function res = skew(Z)
% This function returns the anti-Hermitian part of a matrix
res = (Z - Z')/2;
```

```
end
```

Code B.25: Anti-Hermitian part

```
function res = sym(Z)
% This function returns the Hermitian part of a matrix
res = (Z + Z')/2;
end
```

Code B.26: Hermitian part

```
function res = tang_proj(U, X)
% This function performs a projection onto the tangent space of a
    point of
% the complex valued Stiefel manifold
n = size(U, 1);
res = U * skew(U' * X) + (eye(n) - U * U') * X;
end
```

Code B.27: Projection onto tangent space

```
function res = tang_proj_perp(U, X)
% This function perform a projection onto the orthogonal space of the
% tangent space of a point in the complex valued Stiefel manifold
res = U * sym(U' * X);
end
```

Code B.28: Projection onto space perpendicular to the tangent

```
function t = twopoint(U, U_prev, grad, grad_prev)
% This function performs the two-point algorithm to compute the step-
    size
% used in line-search based algorithms
DU = U - U_prev;
Dgrad = grad - grad_prev;
t = abs(trace(DU' * Dgrad)) / trace(Dgrad' * Dgrad);
end
```

Code B.29: Two-point step size

```
function res = unvec(X)
% This function performs the inverse of the vectorisation operation
    res = reshape(X, sqrt(size(X,1)), sqrt(size(X,1)));
end
```

Code B.30: De-vectorisation

```
function res = vec(X)
% This function performs the vectorisation operation
    res = reshape(X, size(X,1)*size(X,2), 1);
end
```

Code B.31: Vectorisation

```
function res = vec_proj_noise_recovery(P, N)
% This function returns the matrix that applies the composition of
% projection onto the code, noise and Petz recovery to a vectorised
    density
% matrix
    Ac  = proj_noise_recovery_ops(P,N);
    res = zeros(size(Ac{1,1}).^2);
    for j = 1:length(Ac)
        for k = 1:length(Ac)
            res = res + kron(conj(Ac{j,k}), Ac{j,k});
        end
    end
end
```

Code B.32: Vectorised projection, noise and recovery

References

[1] H. Barnum and E. Knill. “Reversing Quantum Dynamics with Near-Optimal Quantum and Classical Fidelity”. In: *Journal of Mathematical Physics* 43.5 (2002), p. 2097. ISSN: 00222488. DOI: 10.1063/1.1459754. URL: https://pubs.aip.org/aip/jmp/article/43/5/2097-2106/382640 (visited on 06/13/2023).

[2] Jonathan Barzilai and Jonathan M. Borwein. “Two-Point Step Size Gradient Methods”. In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. ISSN: 0272-4979, 1464-3642. DOI: 10.1093/imanum/8.1.141. URL: https://academic.oup.com/imajna/article-lookup/doi/10.1093/imanum/8.1.141 (visited on 04/08/2023).

[3] Charles H. Bennett et al. “Mixed-State Entanglement and Quantum Error Correction”. In: *Physical Review A* 54.5 (Nov. 1, 1996), pp. 3824–3851. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.54.3824. URL: https://link.aps.org/doi/10.1103/PhysRevA.54.3824 (visited on 08/28/2023).

[4] Robin Blume-Kohout et al. “Information Preserving Structures: A General Framework for Quantum Zero-Error Information”. In: *Physical Review A* 82.6 (Dec. 7, 2010), p. 062306. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.82.062306. arXiv: 1006.1358 [quant-ph]. URL: http://arxiv.org/abs/1006.1358 (visited on 04/08/2023).

[5] Alan Edelman, Tomás A. Arias, and Steven T. Smith. “The Geometry of Algorithms with Orthogonality Constraints”. In: *SIAM Journal on Matrix Analysis and Applications* 20.2 (Jan. 1998), pp. 303–353. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/S0895479895290954. URL: http://epubs.siam.org/doi/10.1137/S0895479895290954 (visited on 06/12/2023).

[6] Andrew S. Fletcher, Peter W. Shor, and Moe Z. Win. “Channel-Adapted Quantum Error Correction for the Amplitude Damping Channel”. In: *IEEE*

Transactions on Information Theory 54.12 (Dec. 2008), pp. 5705–5718. ISSN: 0018-9448. DOI: 10.1109/TIT.2008.2006458. URL: http://ieeexplore.ieee.org/document/4675715/ (visited on 06/19/2023).

[7] Andrew S. Fletcher, Peter W. Shor, and Moe Z. Win. "Optimum Quantum Error Recovery Using Semidefinite Programming". In: *Physical Review A* 75.1 (Jan. 31, 2007), p. 012338. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.75.012338. URL: https://link.aps.org/doi/10.1103/PhysRevA.75.012338 (visited on 06/17/2023).

[8] Alexei Gilchrist, Nathan K. Langford, and Michael A. Nielsen. "Distance Measures to Compare Real and Ideal Quantum Processes". In: *Physical Review A* 71.6 (June 13, 2005), p. 062310. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.71.062310. URL: https://link.aps.org/doi/10.1103/PhysRevA.71.062310 (visited on 06/25/2023).

[9] András Gilyén et al. "Quantum Algorithm for Petz Recovery Channels and Pretty Good Measurements". In: *Physical Review Letters* 128.22 (June 1, 2022), p. 220502. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.128.220502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.128.220502 (visited on 05/12/2023).

[10] G H Golub and V Pereyra. "The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate". In: *SIAM Journal on Numerical Analysi* 10.2 (1973), pp. 413–43.

[11] Uwe Helmke and John B. Moore. *Optimization and Dynamical Systems*. Red. by B. W. Dickinson et al. Communications and Control Engineering. London: Springer London, 1994. ISBN: 978-1-4471-3469-5 978-1-4471-3467-1. DOI: 10.1007/978-1-4471-3467-1. URL: http://link.springer.com/10.1007/978-1-4471-3467-1 (visited on 06/12/2023).

[12] *IBM Quantum Computing | Roadmap*. URL: https://www.ibm.com/quantum/roadmap (visited on 08/26/2023).

[13] Jeongho Kim et al. "A Stochastic Consensus Method for Nonconvex Optimization on the Stiefel Manifold". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020 59th IEEE Conference on Decision and Control (CDC). Jeju, Korea (South): IEEE, Dec. 14, 2020, pp. 1050–1057. ISBN: 978-1-72817-447-1. DOI: 10.1109/CDC42340.2020.9304325. URL: https://ieeexplore.ieee.org/document/9304325/ (visited on 04/08/2023).

[14] Emanuel Knill and Raymond Laflamme. “Theory of Quantum Error-Correcting Codes”. In: *Physical Review A* 55.2 (Feb. 1, 1997), pp. 900–911. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.55.900. URL: https://link.aps.org/doi/10.1103/PhysRevA.55.900 (visited on 04/11/2023).

[15] Emanuel Knill, Raymond Laflamme, and Lorenza Viola. “Theory of Quantum Error Correction for General Noise”. In: *Physical Review Letters* 84.11 (Mar. 13, 2000), pp. 2525–2528. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.84.2525. URL: https://link.aps.org/doi/10.1103/PhysRevLett.84.2525 (visited on 04/10/2023).

[16] J J Koliha. “Continuity and Differentiability of the Moore-Penrose Inverse in C∗-Algebras”. In: *Mathematica Scandinavica* 88.1 (2001), pp. 154–160.

[17] Robert L. Kosut. “Optimal Quantum Control via Adaptive Error Correction”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference*. 2009 Joint 48th IEEE Conference on Decision and Control (CDC) and 28th Chinese Control Conference (CCC). Shanghai, China: IEEE, Dec. 2009, pp. 386–391. ISBN: 978-1-4244-3871-6. DOI: 10.1109/CDC.2009.5400129. URL: http://ieeexplore.ieee.org/document/5400129/ (visited on 04/08/2023).

[18] Robert L. Kosut and Daniel A. Lidar. “Quantum Error Correction via Convex Optimization”. In: *Quantum Information Processing* 8.5 (Oct. 2009), pp. 443–459. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-009-0120-2. URL: http://link.springer.com/10.1007/s11128-009-0120-2 (visited on 04/08/2023).

[19] Robert L. Kosut, Alireza Shabani, and Daniel A. Lidar. “Robust Quantum Error Correction via Convex Optimization”. In: *Physical Review Letters* 100.2 (Jan. 16, 2008), p. 020502. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.100.020502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.100.020502 (visited on 06/18/2023).

[20] Hyukjoon Kwon, Rick Mukherjee, and M. S. Kim. “Reversing Lindblad Dynamics via Continuous Petz Recovery Map”. In: *Physical Review Letters* 128.2 (Jan. 11, 2022), p. 020403. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.128.020403. URL: https://link.aps.org/doi/10.1103/PhysRevLett.128.020403 (visited on 05/12/2023).

[21] Raymond Laflamme et al. "Perfect Quantum Error Correcting Code". In: *Physical Review Letters* 77.1 (July 1, 1996), pp. 198–201. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.77.198. URL: https://link.aps.org/doi/10.1103/PhysRevLett.77.198 (visited on 08/28/2023).

[22] Debbie W. Leung et al. "Approximate Quantum Error Correction Can Lead to Better Codes". In: *Physical Review A* 56.4 (Oct. 1, 1997), pp. 2567–2573. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.56.2567. URL: https://link.aps.org/doi/10.1103/PhysRevA.56.2567 (visited on 04/23/2023).

[23] D. A. Lidar, I. L. Chuang, and K. B. Whaley. "Decoherence-Free Subspaces for Quantum Computation". In: *Physical Review Letters* 81.12 (Sept. 21, 1998), pp. 2594–2597. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.81.2594. URL: https://link.aps.org/doi/10.1103/PhysRevLett.81.2594 (visited on 08/26/2023).

[24] Tianyi Lin et al. "Projection Robust Wasserstein Distance and Riemannian Optimization". In: ().

[25] Hendrik Poulsen Nautrup et al. "Optimizing Quantum Error Correction Codes with Reinforcement Learning". In: *Quantum* 3 (Dec. 16, 2019), p. 215. ISSN: 2521-327X. DOI: 10.22331/q-2019-12-16-215. arXiv: 1812.08451 [quant-ph]. URL: http://arxiv.org/abs/1812.08451 (visited on 04/08/2023).

[26] M. A. Nielsen. *The Entanglement Fidelity and Quantum Error Correction*. June 13, 1996. arXiv: quant-ph/9606012. URL: http://arxiv.org/abs/quant-ph/9606012 (visited on 06/17/2023). preprint.

[27] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 1st ed. Cambridge University Press, June 5, 2012. ISBN: 978-1-107-00217-3 978-0-511-97666-7. DOI: 10.1017/CBO9780511976667. URL: https://www.cambridge.org/core/product/identifier/9780511976667/type/book (visited on 06/13/2023).

[28] Dénes Petz. "Sufficiency of Channels over von Neumann Algebras". In: *The Quarterly Journal of Mathematics* 39.1 (1988), pp. 97–108. ISSN: 0033-5606, 1464-3847. DOI: 10.1093/qmath/39.1.97. URL: https://academic.oup.com/qjmath/article-lookup/doi/10.1093/qmath/39.1.97 (visited on 06/13/2023).

[29] M. Reimpell and R. F. Werner. “Iterative Optimization of Quantum Error Correcting Codes”. In: *Physical Review Letters* 94.8 (Mar. 2, 2005), p. 080501. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.94.080501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.94.080501 (visited on 06/21/2023).

[30] Hiroyuki Sato. *Riemannian Optimization and Its Applications*. SpringerBriefs in Electrical and Computer Engineering. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-62389-0 978-3-030-62391-3. DOI: 10.1007/978-3-030-62391-3. URL: http://link.springer.com/10.1007/978-3-030-62391-3 (visited on 04/08/2023).

[31] L. J. Stephenson et al. “High-Rate, High-Fidelity Entanglement of Qubits Across an Elementary Quantum Network”. In: *Physical Review Letters* 124.11 (Mar. 16, 2020), p. 110501. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.124.110501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.124.110501 (visited on 08/26/2023).

[32] F Strocchi. *An Introduction to the Mathematical Structure of Quantum Mechanics: A Short Course for Mathematicians*. 2nd ed. Vol. 28. Advanced Series in Mathematical Physics. WORLD SCIENTIFIC, Oct. 2008. ISBN: 978-981-283-522-2 978-981-283-523-9. DOI: 10.1142/7038. URL: https://www.worldscientific.com/worldscibooks/10.1142/7038 (visited on 07/31/2023).

[33] Lorenza Viola, Emanuel Knill, and Seth Lloyd. “Dynamical Decoupling of Open Quantum Systems”. In: *Physical Review Letters* 82.12 (Mar. 22, 1999), pp. 2417–2421. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.82.2417. URL: https://link.aps.org/doi/10.1103/PhysRevLett.82.2417 (visited on 04/27/2023).

[34] Christian Weedbrook et al. “Gaussian Quantum Information”. In: *Reviews of Modern Physics* 84.2 (May 1, 2012), pp. 621–669. ISSN: 0034-6861, 1539-0756. DOI: 10.1103/RevModPhys.84.621. URL: https://link.aps.org/doi/10.1103/RevModPhys.84.621 (visited on 06/13/2023).

[35] Michael M. Wolf. *Quantum Channels & Operations*. July 5, 2012.

[36] W. K. Wootters and W. H. Zurek. “A Single Quantum Cannot Be Cloned”. In: *Nature* 299.5886 (Oct. 1982), pp. 802–803. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/299802a0. URL: https://www.nature.com/articles/299802a0 (visited on 06/13/2023).

[37] Naoki Yamamoto, Shinji Hara, and Koji Tsumura. “Suboptimal Quantum-Error-Correcting Procedure Based on Semidefinite Programming”. In: *Physical Review A* 71.2 (Feb. 28, 2005), p. 022322. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.71.022322. URL: https://link.aps.org/doi/10.1103/PhysRevA.71.022322 (visited on 06/21/2023).

[38] Honglin Yuan et al. “Global Optimization with Orthogonality Constraints via Stochastic Diffusion on Manifold”. In: *Journal of Scientific Computing* 80.2 (Aug. 2019), pp. 1139–1170. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/s10915-019-00971-w. URL: http://link.springer.com/10.1007/s10915-019-00971-w (visited on 04/08/2023).

[39] Xiaojing Zhu. “A Riemannian Conjugate Gradient Method for Optimization on the Stiefel Manifold”. In: *Computational Optimization and Applications* 67.1 (May 2017), pp. 73–110. ISSN: 0926-6003, 1573-2894. DOI: 10.1007/s10589-016-9883-4. URL: http://link.springer.com/10.1007/s10589-016-9883-4 (visited on 06/12/2023).

[40] Ralf Zimmermann. “A Matrix-Algebraic Algorithm for the Riemannian Logarithm on the Stiefel Manifold under the Canonical Metric”. In: *SIAM Journal on Matrix Analysis and Applications* 38.2 (Jan. 2017), pp. 322–342. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/16M1074485. URL: https://epubs.siam.org/doi/10.1137/16M1074485 (visited on 06/12/2023).

[41] Ralf Zimmermann and Knut Hüper. *Computing the Riemannian Logarithm on the Stiefel Manifold: Metrics, Methods and Performance*. Feb. 8, 2022. arXiv: 2103.12046 [cs, math]. URL: http://arxiv.org/abs/2103.12046 (visited on 06/12/2023). preprint.

[42] Fabio Zoratti et al. *Improving the Speed of Variational Quantum Algorithms for Quantum Error Correction*. Mar. 27, 2023. arXiv: 2301.05273 [quant-ph]. URL: http://arxiv.org/abs/2301.05273 (visited on 04/08/2023). preprint.

Acknowledgments

To my supervisors Francesco Ticozzi and Kentaro Ohki.