1 Method

The pulses will be optimized using Krotov's method [1], a gradient-based optimization algorithm available publicly as a ready-to-use Python package [2]. In this chapter Krotov's method for quantum optimal control will be briefly introduced and an implementation as a Python package. Then the numerical simulatons will be explained.

1.1 Krotov's Method for Quantum Optimal Control

Krotov's method for quantum control fundamentally relies on the variational principle to minimize a functional

$$J\Big[\Big\{\Big|\phi_k^{(i)}(t)\Big\rangle\Big\}, \Big\{u_l^{(i)}(t)\Big\}\Big]$$

that depends on the pulse shapes and the quantum states of the system evolved under these pulse shapes [2]. A detailed explanation of this functional can be found in [1]. However a quick summary of how the method works will be presented here. Starting with some guess pulse shapes, the initial state is forward propagated in time using a Hamiltonian of the same form as **??**, and the value of the functional is calculated. Then changes to the pulse shapes which will decrease the value of the functional are calculated, by propagating states with these new pulse shapes, and these updates become the new pulse shapes. This continues until convergence is reached.

The underlying mathematical principles of Krotov's method will not be adressed in the thesis, as that is outside the scope. Therefore the method will be treated as a black box function provided by the Python package which will be presented in the next section.

1.2 Krotov: the Python Package

A Python implementation of the Krotov's method is available at [2], where an abstraction layer for doing optimizations is provided. The package will be referred to as `krotov`. The reason `krotov` was chosen is primarily because it allows for optimization of multiple objectives simultaneously. This is necessary in order to create a single encoding pulse which will realise the state transfer into the cat code basis for any arbitrary state.

The package is built on top of QuTiP, a popular software package for simulating the dynamics of quantum systems [3]. That means it can leverage all the useful features of that package, which is also another reason this method was chosen. The package provides a set of submodules which are used to setup the optimization problem. The important modules will be presented here.

The `objectives` module describes the goal of the optimization, be it a state-to-state evolution or a quantum gate. It is used to specify the initial and target states, the Hamiltonian and the initial guess pulses.

The `functionals` module provides a set of functionals which can be used for Krotov's method. In this work the functional `chis_ss` has been chosen for all optimizations.

In the `convergence` module, help functions for the stopping criteria are provided for the optimization. It exposes values such as the overlap of the forward propagated state and target state $\langle\psi(T)|\psi^{\text{tar}}\rangle$ during the optimization which can be used to check when convergence is reached. The stopping criteria chosen for this thesis is either when a minimum fidelity is obtained, the fiedlity being given by [4]

$$F = \frac{1}{S}|\sum_{k=1}^{S}\big\langle\psi_k(T)\big|\psi_k^{\text{tar}}\big\rangle|^2, \tag{1.1}$$

where S is the number of (simultaneous) state transfers we want to realise, or when the change in fidelity between iterations ΔF falls below a specified value (where a local minimum can be assumed to have been reached). The values of these will be specified in **??**.

Finally, the `optimization` module is used to setup the whole optimization. It is possible to choose the time discretization, convergence criteria, optimization functional, and a function which will run after every iteration. Also the step size $1/\lambda$ of the pulse updates can be chosen, which is further explained in [2]. Note however that although a large step size could achieve faster convergence, it can also lead to numerical instability.

1.3 Hardware and Operating System

All simulations and optimizations were run on a Lenovo Yoga 520-14IKB PC with an Intel(R) Core(TM) i7-8550 CPU running Windows 10. On the software side, they were run through a Jupyter notebook running Python 3.6.7 installed on Ubuntu 18.04.2 under Windows Subsystem for Linux (WSL). The source code is given in **??**.