Numerical Optimization of Microwave and Radio Frequency Control Pulses for the Nitrogen-Vacancy Electron-Nuclear Spin Register

Author

Mark IJspeert

Supervisors

Ir. B. Hensen
Dr. N. V. Budko

Group Leader

Prof. Dr. Ir. R. Hanson

BACHELOR THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science Applied Physics and Mathematics
at the Delft University of Technology

January, 2015

Abstract

Numerical optimizations of the microwave and radio frequency control pulses for the nitrogen vacancy electron-nuclear spin register were performed by means of a gradient ascent pulse engineering method (GRAPE). Three examples of spin control — specific quantum state preparation and the implementation of a controlled-NOT and unconditional π - gate have been optimized through parallel computation. The optimized control pulses reach fidelities of $F = 0.99944$, $F = 0.77929$ and $F = 0.99143$ respectively, whereas simulation of Rabi oscillation based control yields corresponding results of $F = 0.96325$, $F = 0.21884$ and $F = 0.42702$. The fixed step size in the GRAPE algorithm imposes a trade-off between monotone and fast convergence of optimized control pulses and the unspecific performance function may lead to erroneous optimizations. These complications require future research to be resolved. Furthermore, implementing the effect of decoherence yields no significant average fidelity improvement, but has a positive effect on reducing the spread of final density states for the case of quantum state preparation.

Acknowledgements

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to gratefully acknowledge the enthusiastic supervision of Bas Hensen and Neil Budko who inspired, encouraged and fully supported me in every trial that came my way. Deepest gratitude is due to Gilmar Baboolal for his motivating and reassuring attitude in times of darkness and despair. Finally, yet importantly, I would like to express my heartfelt thanks to my family and friends for their help and wishes for the prosperous realization of this project.

«Какой-то математик сказал, что наслаждение не в открытии истины, но в искании её.»

- Лев Николаевич Толстой -

'Some mathematician, I believe, has said that true pleasure lies not in the discovery of truth, but in the search for it.'

- Lev Nikolayevich Tolstoy -

Contents

Chapter 1

Introduction

Ever since the creation of the modern theory of quantum mechanics, it has been an indispensable part of science which is now widely regarded as one of the most successful theories in physics. For the rules of quantum mechanics are simple, but counterintuitive, physicists have laid the foundations of quantum computation and quantum information in pursuit of sharpening our intuition and fundamental understanding of quantum mechanics. Furthermore, these counterintuitive predictions have triggered great interest in the development of new quantum technologies, such as the realization of quantum simulators, quantum computers and quantum cryptography. For any of these applications, the ability to fully control single quantum systems is essential. To this end, promising technologies including phosphorus donor electrons in silicon nanostructures, Gallium Arsenide quantum dots, superconducting Cooper-pair boxes and nitrogen-vacancy centres in diamond constitute research areas of great interest. A prime candidate for a solid-state multi-qubit quantum register is the electron-nuclear spin register in the nitrogen-vacancy centre, which has a long-lived electronic spin with a robust optical interface. The electron-nuclear spin register can be manipulated via irradiation by microwave and radio frequency pulses, which is currently performed through Rabi oscillation based control.

This thesis describes the numerical optimization of the pulses that constitute this form of control by means of gradient ascent pulse engineering (GRAPE). The main contribution of the present thesis lies in the self-contained mathematical analysis of the GRAPE algorithm, its implementation in Python and the fine tuning of the various numerical and experimental parameters for several configurations of practical importance.

Throughout this work, the molecular and energy level structrure of the nitrogen-vacancy centre shall be discussed first, followed by an introduction to the principles of Rabi oscillation based control. The subsequent sections provide a self-contained mathematical description of the GRAPE algorithm. Although we emphasize the authenticity of all proofs within, note that many proofs through-

out this work are rather fundamental and therefore likely to be reported in literature. This chapter is followed by all simulation results and the analysis of optimized quantum state preparation and synthesis of unitary transformations, decoherence based optimization and parallel computation. All conclusions are summarized within the final chapter of this thesis.

Chapter 2

Theory

2.1 Nitrogen-vacancy centre

2.1.1 Molecular structure

Imperfections in the crystal lattice of diamond are common. Such crystallographic defects in diamond may be the result of lattice irregularities or impurities, introduced during or after the diamond growth. One of these defects is called the nitrogen-vacancy defect (NV), which consists of a substitutional nitrogen atom next to a missing lattice vacancy. [1] See Figure 2.1. Two charge

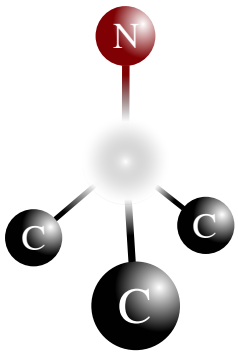

Figure 2.1: Schematic representation of the nitrogen-vacancy centre in diamond. The centered gap in this tetrahedral structure represents the lattice vacancy.

states of this defect are known, a neutral state NV^0 and a negatively charged state NV^-. The former hosts five electrons, consisting of a nitrogen non-bonding valence electron pair and three unpaired carbon electrons[1]. The latter, which is commonly referred to as the general NV centre and which shall be of exclusive interest throughout this work, hosts an additional electron that is captured from the environment.

2.1.2 Energy level structure

Orbital ground state

In the negative charge state NV^-, an extra electron is located at the vacancy site, causing the total of 6 electrons to couple in a way to form a spin $S = 1$ system. Incorporating the effect of one electron on the others leads to the separation of the triplet state sublevel energies, which is called the zero field splitting. In the spin triplet orbital ground state, levels with different values of magnetic spin quantum number ($m_s = 0, \pm 1$) are separated by a zero field splitting of $D = 2\pi \times 2.878\,\text{GHz}$ [1] due to this effect. See Figure 2.2 (B). In

[1] Two of these electrons form a quasi covalent bond, while the other remains unpaired.

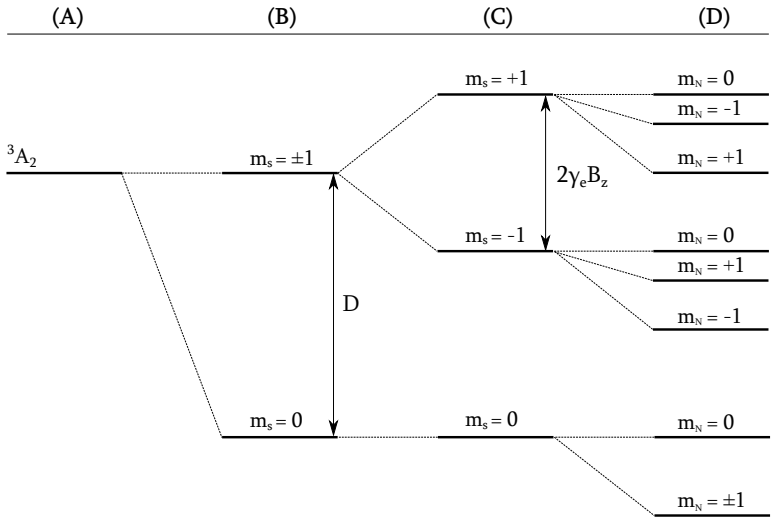

Figure 2.2: Schematic representation of the triplet spin level structure in the orbital ground state. Different degrees of splitting are shown: (A) No splitting. (B) Zero field splitting of $D = 2\pi \times 2.878\,\text{GHz}$. (C) Zeeman splitting that separates the $m_s = \pm 1$ levels by $2\gamma_e B_z$, where γ_e and B_z denote the electron gyromagnetic ratio and the z-component of the applied magnetic field respectively. (D) Nuclear quadrupole splitting separates the $m_N = \pm 1$ states from the $m_N = 0$ level, while hyperfine interaction between the electronic and nuclear spin splits the $m_N = \pm 1$ levels only if $m_s = \pm 1$.

the presence of an applied magnetic field in the z-direction[2] (denoted by B_z), the Zeeman effect causes further separation of these levels by $2\gamma_e B_z$, where γ_e denotes the electron gyromagnetic ratio ($\gamma_e = 2\pi \times 2.802\,\text{MHz}$). See Figure 2.2 (C). The corresponding hamiltonian of the orbital ground state up to this description of the spin system equals

$$H = DS_z^2 + \gamma_e B_z S_z, \tag{2.1}$$

where S_z denotes the spin-1 operator for the z-component of the electronic spin (see Appendix A). The two terms on the right-hand side in this equation account for the zero field splitting and Zeeman effect respectively. Other interactions must be taken into account for the spin system to be modeled accurately. One of these effects is the hyperfine interaction between the nuclear spins of surrounding atoms and the NV electronic spin which leads to small shifts and splittings in the energy levels. The NV electronic spin couples to the nitrogen nuclear spin of $I_N = 1$ with an interaction strength of $A_N = 2\pi \times 2.186\,\text{MHz}$ [1]. Another effect is the quadrupolar splitting of the nitrogen nuclear spin due to the non-spherical charge distribution of the nucleus [2]. See Figure 2.2 (D). Its

[2]The z-axis is defined to be in the [111] crystalline direction, coinciding with the axis through the nitrogen-vacancy pair.

interaction strength is $Q = 2\pi \times 4.946\,\mathrm{MHz}$ [1]. A secular approximation [1] allows the terms containing the x and y-components of the spin-1 operators (i.e. S_x, S_y, I_{N_x} and I_{N_y}) to be neglected, which allows the hamiltonian describing both the electronic and nitrogen nuclear spin in the orbital ground state to be written as

$$\begin{aligned} H_{\mathrm{GR}} = D(S_z^2 \otimes \mathbb{I}_3) + \gamma_e B_z (S_z \otimes \mathbb{I}_3) \\ + \gamma_N B_z (\mathbb{I}_3 \otimes I_{N_z}) - Q(\mathbb{I}_3 \otimes I_{N_z}^2) - A_N (S_z \otimes I_{N_z}), \end{aligned} \tag{2.2}$$

where I_{N_z} denotes the spin-1 operator for the z-component of the nuclear spin (in fact equal to S_z), $\mathbb{I}_3$ represents a 3×3 identity matrix and γ_N is the nuclear gyromagnetic ratio. First note the use of the Kronecker (tensor) product. Since the hamiltonian now describes two spin-1 systems, it must be a 9×9 complex matrix requiring every term in the equation above to be of equal size. Taking the Kronecker product of the spin-1 operators with the identity matrix satisfies this condition. Secondly, a Zeeman splitting of the nuclear spin due to the applied magnetic field accounts for the third term in the hamiltonian. The last two terms represent the quadrupole and hyperfine splitting respectively.

Control hamiltonians

Full control over the spin state of a quantum system enables applications in spin-based quantum information processing [1] As for the NV centre, spin control is achieved via the application of microwave (MW) and radio frequency (RF) radiation, driving the electron and nuclear spin respectively. In a rotating frame, the rotating wave approximation (RWA) is made allowing for the hamiltonian H_e corresponding to the microwave spin control, which contains the driving amplitude Ω_e and frequency ω_e, to be written as

$$H_e = \tfrac{1}{2}\sqrt{2} \cdot \Omega_e \left(S_x \otimes \mathbb{I}_3\right) - \omega_e \left(S_z^2 \otimes \mathbb{I}_3\right). \tag{2.3}$$

A similar expression exists for the RF field driving the nuclear spin transitions where Ω_N and ω_N denote the driving amplitude and frequency respectively:

$$H_N = \tfrac{1}{2}\sqrt{2} \cdot \Omega_N \left(\mathbb{I}_3 \otimes S_x\right) + \omega_N \left(\mathbb{I}_3 \otimes S_z^2\right). \tag{2.4}$$

Spin system hamiltonian

The spin system hamiltonian can now be calculated by simply adding the results from the preceding Equations 2.2, 2.3 and 2.4. Throughout this work, the emphasis lies on the optimization of the driving amplitudes given fixed driving frequencies. For this reason, it is convenient to separate the control-dependent terms from the constants in this system hamiltonian:

$$\begin{aligned} H = \underbrace{H_{\mathrm{GR}} - \omega_e \left(S_z^2 \otimes \mathbb{I}_3\right) + \omega_N \left(\mathbb{I}_3 \otimes S_z^2\right)}_{H_0} \\ + \Omega_e \cdot \underbrace{\tfrac{1}{2}\sqrt{2} \left(S_x \otimes \mathbb{I}_3\right)}_{H_1} + \Omega_N \cdot \underbrace{\tfrac{1}{2}\sqrt{2} \left(\mathbb{I}_3 \otimes S_x\right)}_{H_2}. \end{aligned} \tag{2.5}$$

Throughout the next sections, the control amplitudes Ω_e and Ω_N shall be denoted by u_1 and u_2 respectively. H can thus be written[3] as:

$$H = H_0 + \sum_{k=1}^{2} u_k H_k. \tag{2.6}$$

2.1.3 Non-optimized control

Rabi oscillations

The phenomenon of Rabi oscillations is a basic process used to manipulate qubits — two-level quantum systems. These oscillations are obtained by exposing qubits to periodic electric or magnetic fields during suitably adjusted time intervals. [3] For example, let $|1\rangle$ and $|2\rangle$ denote two spin states of an electron subject to a magnetic field $\vec{B} = B_0\hat{z} + B_1\left(\cos\omega t\ \hat{x} - \sin\omega t\ \hat{y}\right)$. If the qubit is in the initial state $|1\rangle$ at $t = 0$, the probability $P_{1\to2}(t)$ that it is found in the state $|2\rangle$ at time t is given by

$$P_{1\to2}(t) = \left(\frac{\omega_1}{\Omega}\right)^2 \sin^2\left(\frac{\Omega t}{2}\right). \tag{2.7}$$

If we define $\omega_0 = \frac{\gamma_e B_0}{\hbar}$ and $\omega_1 = \frac{\gamma_e B_1}{\hbar}$, the constant Ω in the equation above equals

$$\Omega = \sqrt{\left(\omega - \omega_0\right)^2 + \omega_1^2}. \tag{2.8}$$

Note that the highest transition probability occurs when $\omega = \omega_0$. In this case, the driving field is said to be on resonance with the qubit transition and Equation 2.7 reduces to

$$P_{1\to2}(t) = \sin^2\left(\frac{\omega_1 t}{2}\right). \tag{2.9}$$

The first spin transition from state $|1\rangle$ to $|2\rangle$, i.e. $P_{1\to2}(t) = 1$ now occurs at $t = \pi/\omega_1$. This driving field having a specific amplitude, frequency and duration is known as a π-pulse and is widely used as a quantum control mechanism. Although it seems that Rabi oscillations are a robust form of control, many quantum systems are far more complicated than an ideal two-state system. Applying Rabi pulses to the NV centre (which has, to certain approximation, 9 spin states with specific energy levels), drives multiple spin transitions. Especially when the transition frequencies are close together in comparison with the driving amplitude, this may result in poor rotation fidelities. The following two sections describe the application of Rabi oscillations to the NV centre for the purpose of quantum state preparation and the engineering of unitary transformations.

[3] This notation provides an elegant way of denoting additional control fields.

Quantum state preparation

Suppose the NV electron-nuclear spin is in the pure state $m_s = -1$, $m_N = 0$ at $t = 0$ and we would like to drive the spin to $m_s = 0$, $m_N = 0$. For the driving field to be on resonance, it is required to equal the appropriate energy splitting, i.e. $\omega_e = D - \gamma_e B_z$. Irradiating with an amplitude of $\Omega_e = 10\,\text{MHz}$ during a time of $T = 50\,\text{ns}$ would in this case define a suitable π-pulse. The time evolution of all 9 state probabilities can be simulated by solving the Liouville-von Neumann equation (see Appendix B). The solution for the $m_N = 0$ states is shown in Figure 2.3. Other spin state probabilities remain practically unaffected due to

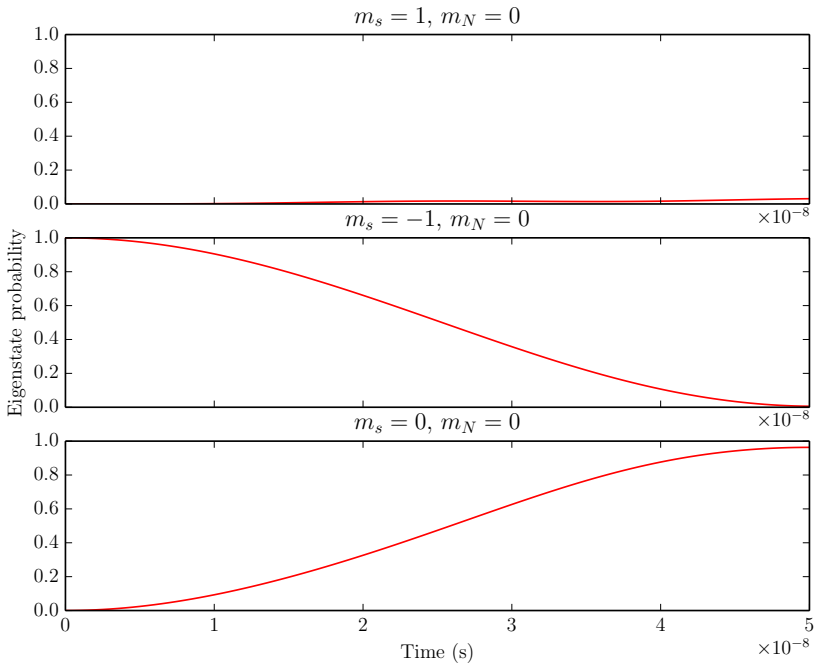

Figure 2.3: Time evolution of the eigenstate probabilities for the $m_N = 0$ states. The system is subject to an on resonant 50 ns pulse of 10 MHz (π-pulse). Note the transition from the initial state $m_s = -1$, $m_N = 0$ to the target state $m_s = 0$, $m_N = 0$. The final fidelity equals $F = 0.96325$.

the approximation that the electron driving amplitude Ω_e does not couple to the nuclear spins. Note that, since this is not a two-level system, the $m_s = 1$, $m_N = 0$ state gains significant eigenstate probability. Clearly, perfect control of a multilevel system by means of Rabi oscillations is out the question. In order to quantify the closeness of two quantum states, the notion of a density state fidelity is introduced. Although many definitions exist in the field of quantum computation and quantum information, we present the following definition to

be used throughout this work. The fidelity[4] of density states ρ and σ is defined as [4]

$$F(\rho,\sigma) := \mathrm{Tr}\left(\rho^{1/2}\sigma\rho^{1/2}\right), \tag{2.10}$$

where $F \in [0,1]$. The definition can now be applied to the example above by replacing σ and ρ with the target and generated state respectively, yielding a final state fidelity of $F = 0.96325$. This value, essentially a benchmark of Rabi oscillation based control, provides a method of comparison to optimized control of the NV electron-nuclear spins, which shall be discussed under section 2.2.

Synthesis of unitary transformations

Instead of driving a specific state transition, implementing a unitary transformation enables state-independent control, i.e. driving an arbitrary initial state with a fixed control pulse in order to perform elementary operations on a quantum spin system. These so-called quantum gates are essential to design quantum circuits that process quantum information. Throughout this work, two important quantum gates shall be discussed: the controlled-NOT (C-NOT) gate and the unconditional π - pulse. The former performs an electron spin rotation between $m_s = 0 \leftrightarrow -1$ that is conditional on the nuclear spin state $m_N = -1$. The corresponding control of a C-NOT gate is a 395 ns pulse of 1.26 MHz. The unconditional π - pulse performs these same, yet unconditional[5] rotations through the application of the exact π - pulse as discussed in the preceding example. Analogous to the quantum state fidelity, it is useful to define a measure of quantum gate closeness. The gate overlap fidelity[6] between a generated transformation X and a target transformation Y is given by [5]

$$F(X,Y) := \left|\frac{\mathrm{Tr}\left(X^\dagger Y\right)}{\mathrm{Tr}\left(Y^\dagger Y\right)}\right|. \tag{2.11}$$

This definition yields gate fidelities of the C-NOT and unconditional π - gate that equal $F = 0.21884$ and $F = 0.42702$ respectively.

[4] Please note that another measure of density state closeness shall be introduced under Section 2.2. This is due to mathematical convenience during the implementation of the GRAPE algorithm.

[5] Unconditional on the nuclear spin.

[6] Another measure of quantum gate closeness shall be introduced next.

2.2 Implementation of GRAPE

2.2.1 Introduction

Steering the dynamics of the electron-nuclear spin generally entails quantum state preparation and the synthesis of unitary transformations. Both of these operations have previously been discussed in the context of non-optimized or Rabi oscillation based control. This section describes a numerical optimization of microwave and radio-frequency control pulses that constitute these elementary operations. The algorithm, which is often referred to as gradient ascent pulse engineering (GRAPE) has recently been introduced in the context of optimally controlling coupled spin dynamics [6]. Whereas an overview containing the basic concept is introduced first, subsequent sections provide a mathematical description of the algorithm.

2.2.2 Overview

Prior to providing a thorough description of the exact implementation of GRAPE, we shall first outline the algorithm.

(i) Depending on the desired operation, a suitable performance function (functional) Φ must be defined. In the case of state preparation, the problem is to find the optimal amplitudes $u_k(t)$ of the MW and RF fields that steer a given initial density matrix ρ_0 in a specified time T to some desired target state ρ_F. The performance function is based on the Frobenius norm to measure the 'overlap' of these states. Engineering a desired unitary transformation U_F is done in a similar way, yet independently of the initial quantum state.

(ii) Once a performance function is defined, the transfer time T is discretized into N equal steps, such that the time-dependent amplitude $u_k(t)$ during the j^{th} step is given by the time-independent value $u_k(j)$ for each $k \in \{1, 2\}$. We define $\Delta t = T/N$.

(iii) In order to maximize the performance function, we approximate the partial derivatives $\frac{\partial \Phi}{\partial u(j)}$ of Φ with respect to each $u_k(j)$ to first order in Δt, where the initial control pulse is guessed.

(iv) We update all values of $u_k(j)$ such that $\tilde{u}_k(j) = u_k(j) + \epsilon \frac{\partial \Phi}{\partial u_k(j)}$, where $\tilde{u}_k(j)$ represents the updated value and ϵ denotes a proper step size. The algorithm now returns to step (iii) and terminates after M iterations.

What follows next is a more thorough and mathematical approach to the implementation of GRAPE.

2.2.3 Frobenius norm

The analysis of matrix-based algorithms often requires the use of matrix norms. In the context of quantum computation, the Frobenius norm is a common measure that quantifies the 'distance' between unitary gates or density matrices. In fact, it is the naturally induced norm of the Frobenius inner product. The definitions are given below. [7]

Definition 2.2.1. The Frobenius inner product of two complex $n \times n$ matrices X and Y containing the elements x_{ij} and y_{ij} respectively, is

$$\langle X|Y\rangle_F := \sum_i \sum_j \overline{x_{ij}} y_{ij}.$$

Definition 2.2.2. The naturally induced Frobenius norm of a complex square matrix X containing the elements x_{ij} is given by

$$||X||_F := \sqrt{\sum_i \sum_j |x_{ij}|^2} = \sqrt{\langle X|X\rangle_F}.$$

The next theorem presents three useful characteristics of the Frobenius inner product.

Theorem 2.2.1. *For two complex $n \times n$ matrices X and Y containing the elements x_{ij} and y_{ij} respectively, the following statements are valid:*

(i) $\langle X|Y\rangle_F = \mathrm{Tr}\left(X^\dagger Y\right)$

(ii) $\langle X|Y\rangle_F = \overline{\langle Y|X\rangle}_F$

(iii) $\langle X^\dagger|Y\rangle_F = \langle Y^\dagger|X\rangle_F$

(iv) $X = X^\dagger \wedge Y = Y^\dagger \implies \langle X|Y\rangle_F \in \mathbb{R}$

Proof.

(i) $\langle X|Y\rangle_F = \sum_i \sum_j \overline{x_{ij}} y_{ij} = \sum_i \sum_j \left(X^\dagger\right)_{ji} (Y)_{ij} = \sum_i \left(X^\dagger Y\right)_{ii} = \mathrm{Tr}\left(X^\dagger Y\right)$.

(ii) By definition, a complex inner product must satisfy this property. A simple proof: $\langle X|Y\rangle_F = \sum_i \sum_j \overline{x_{ij}} y_{ij} = \sum_i \sum_j \overline{x_{ij} \overline{y_{ij}}} = \overline{\sum_i \sum_j x_{ij} \overline{y_{ij}}} = \overline{\langle Y|X\rangle}_F$.

(iii) Using the first property: $\langle X^\dagger|Y\rangle_F = \mathrm{Tr}\,(XY) = \sum_i (XY)_{ii} = \sum_i \sum_j x_{ij} y_{ji} = \sum_j \sum_i y_{ji} x_{ij} = \sum_j (YX)_{jj} = \mathrm{Tr}\,(YX) = \langle Y^\dagger|X\rangle_F$.

(iv) Using properties (ii), (iii) and the fact that X and Y are hermitian, it follows that: $\langle X|Y\rangle_F = \overline{\langle Y|X\rangle}_F = \overline{\mathrm{Tr}\,(Y^\dagger X)} = \overline{\mathrm{Tr}(XY^\dagger)} = \overline{\mathrm{Tr}(X^\dagger Y)} = \overline{\langle X|Y\rangle}_F \implies \langle X|Y\rangle_F \in \mathbb{R}$.

□

2.2.4 Defining a performance function

A proper definition of the Frobenius norm enables us to define a suitable performance function. As mentioned previously, the main goal is to minimize the 'overlap' between density states or unitary gates. A reasonable performance function for a target matrix X and a matrix Y that is to be optimized, would therefore be:

$$\Phi = ||X - Y||_F^2. \tag{2.12}$$

For reasons that will become clear, it is more convenient to define Φ in a slightly different way, depending on its application. In order to do so, the following results are required.

Theorem 2.2.2. *If X and Y are two complex $n \times n$ matrices, then minimizing $\Phi_0 = ||X - Y||_F^2$ is equivalent to maximizing $\Phi_1 = \mathrm{Re}\langle X|Y\rangle_F$.*

Proof. Due to the linearity of the complex inner product: $||X - Y||_F^2 = \langle X - Y|X - Y\rangle_F = \langle X - Y|X\rangle_F - \langle X - Y|Y\rangle_F = \langle X|X\rangle_F - \langle Y|X\rangle_F - \langle X|Y\rangle_F + \langle Y|Y\rangle_F = ||X||_F^2 - \overline{\langle X|Y\rangle}_F - \langle X|Y\rangle_F + ||Y||_F^2 = ||X||_F^2 - 2\,\mathrm{Re}\,\langle X|Y\rangle_F + ||Y||_F^2$. Since $||X||_F^2$ and $||Y||_F^2$ are non-negative terms and since -2 is a constant factor, minimization of Φ_0 is equivalent to maximization of Φ_1. □

Lemma 2.2.3. *If two complex $n \times n$ matrices X and Y are hermitian, then minimizing $\Phi_0 = ||X - Y||_F^2$ is equivalent to maximizing $\Phi_1 = \langle X|Y\rangle_F$.*

Proof. Since X and Y are hermitian, it follows from the last property of Theorem 2.2.1 that $\mathrm{Re}\,\langle X|Y\rangle_F = \langle X|Y\rangle_F$, so minimization of Φ_0 is equivalent to maximization of Φ_1. □

It is now possible to define proper performance functions suited for both quantum state preparation and synthesis of unitary transformations.

Quantum state preparation

In the case of quantum state preparation, X and Y represent density matrices and are therefore hermitian. Lemma 2.2.3 provides a suitable performance function ($\Phi = \langle X|Y\rangle_F$) that is to be maximized. At this moment, it is convenient to replace X and Y by ρ_F and ρ_T respectively, where ρ_T denotes the density state of the system subject to a time-dependent hamiltonian H after a specified time T. Its time evolution is governed by the Liouville-von Neumann equation, which is difficult to solve analytically for time-dependent hamiltonians. However, a simple solution exists if H is time-independent (see Appendix B). By discretizing the time interval $[0, T]$ and assuming that the MW and RF field amplitudes $u_k(t)$ are constants $u_k(j)$ during the j^{th} interval, we satisfy the condition of a time-independent hamiltonian on each interval. The time

evolution of the spin system during a time step $j \in \mathbb{N}^+$ is therefore given by the propagator:

$$U_j = e^{-i\Delta t\left(H_0 + \sum_{k=1}^{2} u_k(j) H_k\right)}. \tag{2.13}$$

Thus, the density state ρ_j after the j^{th} interval can be calculated from the recurrence relation:

$$\rho_j = U_j \rho_{j-1} U_j^\dagger, \tag{2.14}$$

where ρ_0 conveniently denotes the initial density state at $t = 0$. In order to calculate ρ_T, we simply set $\rho_T = \rho_N$ to apply Equation 2.14:

$$\rho_T = U_N \cdots U_1 \rho_0 U_1^\dagger \cdots U_N^\dagger. \tag{2.15}$$

The performance function now equals

$$\Phi = \langle \rho_F | U_N \cdots U_1 \rho_0 U_1^\dagger \cdots U_N^\dagger \rangle_F. \tag{2.16}$$

Synthesis of unitary transformations

In the case of optimizing a unitary transformation, X and Y need not be hermitian matrices. Theorem 2.2.2 therefore provides the function $\Phi = \mathrm{Re}\langle X|Y\rangle_F$ that can now further be adapted for this purpose. As in the preceding derivation, we replace X and Y by U_F (target transformation) and U_T respectively, where U_T denotes the unitary transformation that results from the effect of a certain time-dependent hamiltonian after a specified time T. The difference regarding the quantum state preparation is that no initial density state is required. The current function would therefore be: $\Phi = \mathrm{Re}\langle U_F|U_T\rangle_F$. However, multiplying U_T by any phase factor $e^{i\phi}$ will not affect its operation on a density state ρ, since

$$e^{i\phi} U_T \rho \left(e^{i\phi} U_T\right)^\dagger = e^{i\phi} e^{-i\phi} U_T \rho U_T^\dagger = U_T \rho U_T^\dagger. \tag{2.17}$$

It is therefore equivalent to maximize $\Phi = \mathrm{Re}\langle U_F|e^{i\phi}U_T\rangle_F = \mathrm{Re}\left(e^{i\phi}\langle U_F|U_T\rangle_F\right)$. Denoting $\langle U_F|U_T\rangle_F$ by $re^{i\theta}$ with $r = |\langle U_F|U_T\rangle_F|$ and $\theta = \arg\langle U_F|U_T\rangle_F$ enables us to write $\Phi = \mathrm{Re}\left(e^{i\phi} r e^{i\theta}\right)$, so choosing $\phi = \theta$ yields

$$\Phi = \mathrm{Re}\,(r) = r = |\langle U_F|U_T\rangle_F|. \tag{2.18}$$

Due to the monotonicity of the quadratic function, this is equivalent to the maximization of the performance function:

$$\Phi = |\langle U_F|U_T\rangle_F|^2. \tag{2.19}$$

The final step is to express U_T in terms of known propagators. Note that Equation 2.15 already contains the transformation U_T that is to be optimized:

$$U_T = \prod_{j=N}^{1} U_j = U_N \cdots U_1. \tag{2.20}$$

The performance function is therefore equal to

$$\Phi = |\langle U_F|U_N \cdots U_1\rangle_F|^2. \tag{2.21}$$

2.2.5 Gradient approximation

The next step in the algorithm is to approximate the gradient of the performance function. Once again, we shall distinguish between both applications, since they require a different approach. The state preparation shall be discussed first.

Quantum state preparation

Calculating the gradient of Φ involves determining $\frac{\partial \Phi}{\partial u_k(j)}$ for each time step j and control field k. Its derivation requires several lemmas that shall be presented first.

Lemma 2.2.4. *Let X denote a constant complex $n \times n$ matrix, let $Y(t)$ be a complex $n \times n$ matrix depending on $t \in \mathbb{R}$ and let the elements of X and $Y(t)$ be denoted by x_{ij} and $y_{ij}(t)$ respectively. Then the following statement holds:*

$$\frac{\partial}{\partial t} \operatorname{Tr}(XY) = \operatorname{Tr}\left(X \frac{\partial Y}{\partial t}\right).$$

Proof.

$$\begin{aligned}
\frac{\partial}{\partial t} \operatorname{Tr}(XY) &= \frac{\partial}{\partial t} \sum_i (XY)_{ii} \\
&= \frac{\partial}{\partial t} \sum_i \sum_j x_{ij} y_{ji} \\
&= \sum_i \sum_j x_{ij} \frac{\partial y_{ji}}{\partial t} \\
&= \sum_i \sum_j x_{ij} \left(\frac{\partial Y}{\partial t}\right)_{ji} \\
&= \sum_i \left(X \frac{\partial Y}{\partial t}\right)_{ii} \\
&= \operatorname{Tr}\left(X \frac{\partial Y}{\partial t}\right).
\end{aligned}$$

□

The following lemma requires a result from matrix analysis [8] [9] [10], which is an expression for the derivative of a matrix exponential $e^{X(t)}$ with respect to t. It is given by:

$$\frac{d}{dt} e^{X(t)} = \int_0^1 e^{\alpha X(t)} \frac{dX(t)}{dt} e^{(1-\alpha)X(t)} d\alpha. \tag{2.22}$$

Lemma 2.2.5. *Let U_j denote the propagator as defined in Equation 2.15. Then its partial derivative with respect to $u(j)$ and to first order in Δt equals*

$$\frac{\partial U_j}{\partial u_k(j)} = -i\Delta t H_k U_j + \mathcal{O}(\Delta t^2).$$

Proof. For the sake of readability in this proof, $H_0 + \sum_{k=1}^{2} u_k(j) H_k$ is denoted by H. First, Equation 2.22 is applied to the left-hand side of the equation above, followed by the substitution $\alpha = \tau/\Delta t$. The integral can then be approximated to first order in Δt by subsequently rewriting the exponential function as a truncated power series and performing term-by-term integration. The derivation is given below.

$$\begin{aligned}
\frac{\partial U_j}{\partial u_k(j)} &= \frac{\partial}{\partial u_k(j)} e^{-i\Delta t H} \\
&= \int_0^1 e^{\alpha(-i\Delta t H)} \frac{d\left(-i\Delta t H\right)}{du_k(j)} e^{(1-\alpha)(-i\Delta t H)} d\alpha \\
&= \int_0^1 e^{\alpha(-i\Delta t H)} (-i\Delta t H_k) e^{(1-\alpha)(-i\Delta t H)} d\alpha \\
&= \int_0^{\Delta t} \frac{1}{\Delta t} e^{\frac{\tau}{\Delta t}(-i\Delta t H)} (-i\Delta t H_k) e^{\left(1-\frac{\tau}{\Delta t}\right)(-i\Delta t H)} d\tau \\
&= \int_0^{\Delta t} e^{-i\tau H} (-iH_k) e^{i(\tau-\Delta t)H} d\tau \\
&= -i \int_0^{\Delta t} e^{-i\tau H} H_k e^{i\tau H} d\tau \cdot e^{-i\Delta t H} \\
&= -i \int_0^{\Delta t} e^{-i\tau H} H_k e^{i\tau H} d\tau \cdot U_j \\
&= -i \int_0^{\Delta t} \left(\mathbb{I} - i\tau H + \mathcal{O}(\tau^2)\right) H_k \left(\mathbb{I} + i\tau H + \mathcal{O}(\tau^2)\right) d\tau \cdot U_j \\
&= -i \int_0^{\Delta t} \left(H_k - i\tau H H_k + \mathcal{O}(\tau^2)\right) \left(\mathbb{I} + i\tau H + \mathcal{O}(\tau^2)\right) d\tau \cdot U_j \\
&= -i \int_0^{\Delta t} \left(H_k + i\tau H_k H - i\tau H H_k + \mathcal{O}\left(\tau^2\right)\right) d\tau \cdot U_j \\
&= -i \left(\Delta t H_k + \frac{i\Delta t^2}{2} \left(H_k H - H H_k\right) + \mathcal{O}\left(\Delta t^3\right)\right) U_j \\
&= -i\Delta t H_k U_j + \mathcal{O}(\Delta t^2)
\end{aligned}$$

□

By introducing the following lemma, we can extend the previous result of Lemma 2.2.5 to approximate $\frac{\partial U_j^\dagger}{\partial u_k(j)}$.

Lemma 2.2.6. *If $X(t)$ is a complex $n \times n$ matrix depending on $t \in \mathbb{R}$ and containing the elements $x_{ij}(t)$, then the following statement holds:*

$$\frac{\partial X^\dagger}{\partial t} = \left(\frac{\partial X}{\partial t}\right)^\dagger.$$

Proof. We shall prove the statement by showing that the elements on both sides of the equation are equal. Starting from the left-hand side:

$$\left(\frac{\partial X^\dagger}{\partial t}\right)_{ij} = \frac{\partial \left(X^\dagger\right)_{ij}}{\partial t} = \frac{\partial \overline{x_{ji}}}{\partial t} = \overline{\frac{\partial x_{ji}}{\partial t}} = \overline{\left(\frac{\partial X}{\partial t}\right)_{ji}} = \left(\left(\frac{\partial X}{\partial t}\right)^\dagger\right)_{ij}.$$

□

Corollary 2.2.6.1. *Let U_j denote the propagator as defined in Equation 2.15. Due to the hermiticity of H_k and by combining Lemma 2.2.5 and 2.2.6 it follows that the partial derivative of its conjugate transpose with respect to $u_k(j)$ and to first order in Δt equals*

$$\frac{\partial U_j^\dagger}{\partial u_k(j)} = i\Delta t U_j^\dagger H_k + \mathcal{O}(\Delta t^2).$$

The final lemma that is required entails the conjugate transpose of a matrix product of length $N \in \mathbb{N}$.

Lemma 2.2.7. *Let $X_1, \ldots, X_N$ denote a sequence of N complex $n \times n$ matrices. Then the following statement holds:*

$$\left(\prod_{i=1}^{N} X_i\right)^\dagger = \prod_{i=N}^{1} X_i^\dagger.$$

Proof. The statement is proved by mathematical induction. For $N = 1$, the result is trivial, so assume that $\left(\prod_{i=1}^{n} X_i\right)^\dagger = \prod_{i=n}^{1} X_i^\dagger$ for a certain $n \in \mathbb{N}$. Since $(AB)^\dagger = B^\dagger A^\dagger$ for two complex matrices A and B, the statement can proved to be true for $n+1$:

$$\left(\prod_{i=1}^{n+1} X_i\right)^\dagger = (X_1 \cdots X_{n+1})^\dagger = X_{n+1}^\dagger (X_1 \cdots X_n)^\dagger = X_{n+1}^\dagger \cdots X_1^\dagger = \prod_{i=n+1}^{1} X_i^\dagger.$$

□

We are now ready to calculate the gradient of the performance function for each time step using the preceding lemmas. The result is given by the following theorem.

Theorem 2.2.8. *Let Φ be the performance function as defined in Equation 2.16 and let U_j denote the propagator as defined in Equation 2.15. If we define $\lambda_j := U_{j+1}^\dagger \cdots U_N^\dagger \rho_F U_N \cdots U_{j+1}$ and $\rho_j := U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger$, then the partial derivative of Φ with respect to $u_k(j)$ and to first order in Δt is given by*

$$\frac{\partial \Phi}{\partial u_k(j)} = -\langle \lambda_j | i\Delta t [H_k, \rho_j] \rangle_F.$$

Proof. Starting on the left-hand side of the equation above, we apply Theorem 2.2.1 and Lemma 2.2.7 to rewrite the expression in terms of λ_j (constant) and ρ_j ($u_k(j)$ dependent). Interchanging differentiation and the trace function by using Lemma 2.2.4 and applying Lemma B.0.1 (Appendix B) yields an expression containing the partial derivatives of U_j and $U_j^\dagger$, which are explicitly given to first order in Δt by Lemma 2.2.5 and Corollary 2.2.6.1, yielding the result. The derivation is given below.

$$\begin{aligned}
\frac{\partial \Phi}{\partial u_k(j)} &= \frac{\partial}{\partial u_k(j)} \langle \rho_F | U_N \cdots U_1 \rho_0 U_1^\dagger \cdots U_N^\dagger \rangle_F \\
&= \frac{\partial}{\partial u_k(j)} \operatorname{Tr}\left(\rho_F^\dagger U_N \cdots U_1 \rho_0 U_1^\dagger \cdots U_N^\dagger \right) \\
&= \frac{\partial}{\partial u_k(j)} \operatorname{Tr}\left(\rho_F^\dagger U_N \cdots U_{j+1} U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger U_{j+1}^\dagger \cdots U_N^\dagger \right) \\
&= \frac{\partial}{\partial u_k(j)} \left\langle \left(\rho_F^\dagger U_N \cdots U_{j+1} U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger \right)^\dagger \middle| U_{j+1}^\dagger \cdots U_N^\dagger \right\rangle_F \\
&= \frac{\partial}{\partial u_k(j)} \left\langle \left(U_{j+1}^\dagger \cdots U_N^\dagger \right)^\dagger \middle| \rho_F^\dagger U_N \cdots U_{j+1} U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger \right\rangle_F \\
&= \frac{\partial}{\partial u_k(j)} \operatorname{Tr}\left(\underbrace{U_{j+1}^\dagger \cdots U_N^\dagger \rho_F^\dagger U_N \cdots U_{j+1}}_{\lambda_j^\dagger} \underbrace{U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger}_{\rho_j(u_k(j))} \right) \\
&= \operatorname{Tr}\left(\lambda_j^\dagger \frac{\partial}{\partial u_k(j)} \left(U_j U_{j-1} \cdots U_1 \rho_0 U_1^\dagger \cdots U_{j-1}^\dagger U_j^\dagger \right) \right) \\
&= \operatorname{Tr}\left(\lambda_j^\dagger \left(U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_{j-1}^\dagger \frac{\partial U_j^\dagger}{\partial u_k(j)} + \frac{\partial U_j}{\partial u_k(j)} U_{j-1} \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger \right) \right) \\
&= \operatorname{Tr}\left(\lambda_j^\dagger i\Delta t \left(U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger H_k - H_k U_j \cdots U_1 \rho_0 U_1^\dagger \cdots U_j^\dagger \right) \right) \\
&= \operatorname{Tr}\left(\lambda_j^\dagger i\Delta t \left(\rho_j H_k - H_k \rho_j \right) \right) \\
&= -\langle \lambda_j | i\Delta t [H_k, \rho_j] \rangle_F
\end{aligned}$$

□

Synthesis of unitary transformations

Analogous to the preceding approach, the gradient corresponding to the synthesis of unitary transformations can be approximated. Since the required lemmas have already been discussed, the result is presented straightaway.

Theorem 2.2.9. *Let Φ be the performance function as defined in Equation 2.21 and let U_j denote the propagator as defined in Equation 2.15. If we define $P_j := U_{j+1}^\dagger \cdots U_N^\dagger U_F$ and $X_j := U_j \cdots U_1$, then the partial derivative of Φ with respect to $u_k(j)$ and to first order in Δt is given by*

$$\frac{\partial \Phi}{\partial u_k(j)} = -2 \operatorname{Re}\left(\langle X_j | P_j \rangle_F \langle P_j | i \Delta t H_k X_j \rangle_F\right).$$

Proof. Starting on the left-hand side of the equation above, we apply Theorem 2.2.1 and Lemma 2.2.7 to rewrite the expression in terms of P_j (constant) and X_j ($u_k(j)$ dependent). Interchanging differentiation and the trace function by using Lemma 2.2.4 and applying the product rule for differentiation yields an expression containing the partial derivatives of U_j and $U_j^\dagger$, which are explicitly given to first order in Δt by Lemma 2.2.5 and Corollary 2.2.6.1 Applying Theorem 2.2.1 once again then yields the result. The derivation is given below.

$$\begin{aligned}
\frac{\partial \Phi}{\partial u_k(j)} &= \frac{\partial}{\partial u_k(j)} |\langle U_F | U_N \cdots U_1 \rangle_F|^2 \\
&= \frac{\partial}{\partial u_k(j)} \langle U_F | U_N \cdots U_1 \rangle_F \overline{\langle U_F | U_N \cdots U_1 \rangle}_F \\
&= \frac{\partial}{\partial u_k(j)} \langle U_F | U_N \cdots U_1 \rangle_F \langle U_N \cdots U_1 | U_F \rangle_F \\
&= \frac{\partial}{\partial u_k(j)} \operatorname{Tr}\left(U_F^\dagger U_N \cdots U_{j+1} U_j \cdots U_1\right) \operatorname{Tr}\left(U_1^\dagger \cdots U_j^\dagger U_{j+1}^\dagger \cdots U_N^\dagger U_F\right) \\
&= \frac{\partial}{\partial u_k(j)} \langle \underbrace{U_{j+1}^\dagger \cdots U_N^\dagger U_F}_{P_j} | \underbrace{U_j \cdots U_1}_{X_j} \rangle_F \langle \underbrace{U_j \cdots U_1}_{X_j} | \underbrace{U_{j+1}^\dagger \cdots U_N^\dagger U_F}_{P_j} \rangle_F \\
&= \frac{\partial \langle P_j | X_j \rangle_F}{\partial u_k(j)} \langle X_j | P_j \rangle_F + \langle P_j | X_j \rangle_F \frac{\partial \langle X_j | P_j \rangle_F}{\partial u_k(j)} \\
&= \operatorname{Tr}\left(U_F^\dagger U_N \cdots U_{j+1} \frac{\partial U_j}{\partial u_k(j)} U_{j-1} \cdots U_1\right) \langle X_j | P_j \rangle_F + \\
&\quad \langle P_j | X_j \rangle_F \operatorname{Tr}\left(U_1^\dagger \cdots U_{j-1}^\dagger \frac{\partial U_j^\dagger}{\partial u_k(j)} U_{j+1}^\dagger \cdots U_N^\dagger U_F\right) \\
&= -i \Delta t \langle X_j | P_j \rangle_F \operatorname{Tr}\left(U_F^\dagger U_N \cdots U_{j+1} H_k U_j \cdots U_1\right) + \\
&\quad i \Delta t \langle P_j | X_j \rangle_F \operatorname{Tr}\left(U_1^\dagger \cdots U_j^\dagger H_k U_{j+1}^\dagger \cdots U_N^\dagger U_F\right)
\end{aligned}$$

$$
\begin{aligned}
&= -i\Delta t\langle X_j|P_j\rangle_F\langle P_j|H_kX_j\rangle_F + i\Delta t\langle P_j|X_j\rangle_F\langle H_kX_j|P_j\rangle_F \\
&= -\langle X_j|P_j\rangle_F\langle P_j|i\Delta tH_kX_j\rangle_F - \langle P_j|X_j\rangle_F\langle i\Delta tH_kX_j|P_j\rangle_F \\
&= -\langle X_j|P_j\rangle_F\langle P_j|i\Delta tH_kX_j\rangle_F - \overline{\langle X_j|P_j\rangle_F\langle P_j|i\Delta tH_kX_j\rangle}_F \\
&= -2\,\mathrm{Re}\left(\langle X_j|P_j\rangle_F\langle P_j|i\Delta tH_kX_j\rangle_F\right)
\end{aligned}
$$

□

2.2.6 Concluding remarks

The preceding derivation enables the implementation of the GRAPE algorithm to be written briefly, yet explicitly in the following way:

(i) Select the step size ϵ, the number of time intervals N and the pulse duration T.

(ii) Guess the initial control pulses $u_k(j)$.

(iii) Update all values of $u_k(j)$ according to the corresponding application:

- Quantum state preparation:

$$\tilde{u}_k(j) = u_k(j) - \epsilon\langle\lambda_j|i\Delta t[H_k,\rho_j]\rangle_F.$$

- Synthesis of unitary transformations:

$$\tilde{u}_k(j) = u_k(j) - 2\epsilon\,\mathrm{Re}\left(\langle X_j|P_j\rangle_F\langle P_j|i\Delta tH_kX_j\rangle_F\right).$$

(iv) Iterate step (iii) for a fixed number of times M.

The corresponding computer code, which was written in Python, can be found under Appendix C.

2.3 Decoherence

One problem which particularly plagues the engineering of a quantum computer is the coupling of the quantum states to an environment and the subsequent destruction of the quantum information in the computer through the process known as decoherence. [11] Regarding optimal control of the NV spin system, decoherence arises when the frequency of the control radiation is unexpectedly not on resonance with the spin transition. [5] This so-called off-resonance error will cause the spin dynamics to proceed with an unknown detuning parameter having a possible negative effect on the final density state or gate fidelity. In order to model the effect of decoherence in the case of quantum state preparation, the fixed applied magnetic field in the z-direction B_z is replaced by a randomly fluctuating field. To this end, the evolution of an initial density state is calculated for a number of magnetic field samples K from a normal distribution with $\mu = B_z$ and $\sigma = 0.005 \cdot B_z$. The average of all resulting final density state fidelities then constitutes a measure for the spin control affected by decoherence. Although the current GRAPE algorithm does not take the effect of decoherence into account, a minor adjustment enables the implementation of decoherence-based optimization. Whereas the current algorithm calculates the gradient of a performance function Φ corresponding to a fixed magnetic field B_z, the decoherence-based algorithm averages over several performance functions prior to the gradient calculation. The new decoherence-based performance function Φ_D would therefore be equal to

$$\Phi_D = \frac{1}{L} \sum_{i=1}^{L} \Phi_{B_i}, \tag{2.23}$$

where $\Phi_{B_1}, \ldots, \Phi_{B_L}$ denote the performance functions corresponding to L sampled magnetic fields from the exact distribution as mentioned above.

Chapter 3

Simulation and Discussion

3.1 Introduction

This chapter elucidates the simulations performed in this research where the emphasis is placed on finding the parameters that maximize the final fidelities in the case of both quantum state preparation and synthesis of unitary transformations. This includes the determination of a suitable step size ϵ, selecting a proper number of time intervals N and finding the optimal pulse length T, which is in itself an optimization problem. Although solutions can be found by means of exhaustive searching through a specified subset of the hyperparameter space, this method would require an excessive amount of computation time. A more practical, though less reliable approach would be solving for different parameters successively, starting from an initial guess. The resulting control pulses shall subsequently be discussed under the optimal control sections. Latter sections include decoherence-based optimization and the beneficial effects of parallel computation. All simulations throughout this work were performed with electron and nuclear spin driving frequencies of $\omega_e = D - \gamma_e B_z$ and $\omega_N = Q$ respectively.

3.2 Quantum state preparation

3.2.1 Overview

Throughout this section, constructing optimal control pulses for the exact example of quantum state preparation (as discussed under Section 2.1.3) shall be of exclusive interest, i.e. driving the spin system from $m_s = -1$, $m_N = 0$ to $m_s = 0$, $m_N = 0$. In what follows next, the π-pulse as it is applied in Rabi oscillation based control shall be used as an initial guess. Thenceforth, if a suitable step size has been chosen, a time interval selection is made. This is followed by an analysis of pulse length variation, in which the initial guess is changed accordingly, i.e. choosing a constant pulse of 10 MHz and duration T. Selections are made through the analysis of fidelity plots in which the density state fidelities are shown as a function of the number of iterations of the GRAPE algorithm.

3.2.2 Determination of a suitable step size

In order to determine a suitable step size, initial algorithm parameters must be chosen. Figure 3.1 shows fidelity curves corresponding to 5 different step sizes for $N = 10$ and $T = 50$ ns. First note that the lowest step size of $\epsilon = 10^{13}$ results in only a slight fidelity improvement after 1000 iterations. Larger step sizes lead to faster rates of convergence and higher final density state fidelities, but only up to and including $\epsilon = 10^{15}$. A step size of $\epsilon = 10^{16}$ but particularly $\epsilon = 10^{17}$ causes the algorithm to overshoot the (local) optimum, resulting in poor final density state fidelities which are even lower than the initial guess fidelity. A major weakness of the GRAPE algorithm now becomes apparent — monotone and fast convergence is limited by imposing a fixed step size. Since achieving

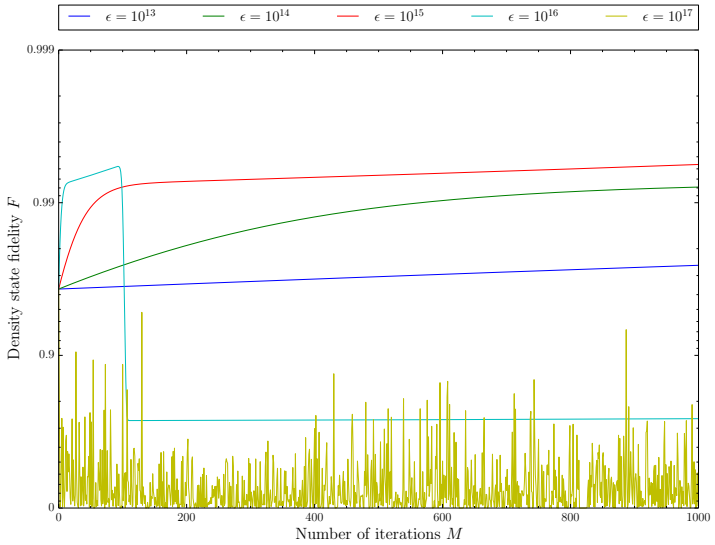

Figure 3.1: Density state fidelity plot on a single inverse logarithmic scale for 5 different step sizes with $N = 10$ and $T = 50\,\text{ns}$. Note that all graphs start at the same initial density state fidelity of $F = 0.96325$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

monotone convergence is our primary concern throughout this work, a step size of $\epsilon = 10^{15}$ appears to yield the most promising result and shall therefore be selected for the optimization of a suitable time interval.

3.2.3 Time interval selection

In order to find the optimal number of time intervals N we set $\epsilon = 10^{15}$ and $T = 50\,\text{ns}$. Once again, we emphasize the unlikeliness of finding a global optimum with this crude optimization method. It only provides the best result for a fixed step size and a number of intervals from a fixed set. However, the gradient of the performance function contains the factor $\Delta t = {}^{T}/_{N}$ (see Theorem 2.2.8). This gives reason to believe that a higher number of time intervals leads to a smaller gradient through Δt, which in turn requires a larger step size ϵ in order to achieve a similar final density state fidelity. We would therefore expect the preceding step size optimization to be specific for the number of time intervals ($N = 10$). This may cause the time interval optimization to yield the best result for $N = 10$. Figure 3.2 shows fidelity curves corresponding to 6 different numbers of intervals. Indeed, setting $N = 10$ leads to the highest final density state

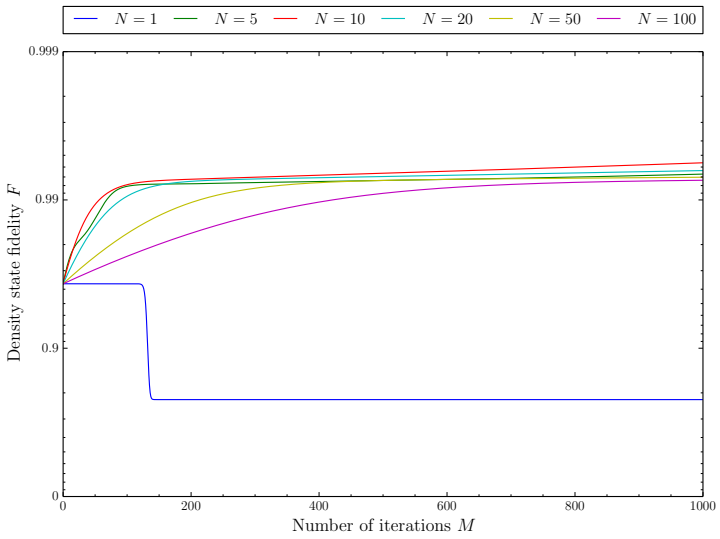

Figure 3.2: Density state fidelity plot on a single inverse logarithmic scale for 6 different numbers of time intervals with $\epsilon = 10^{15}$ and $T = 50\,\text{ns}$. Note that all graphs start at the same initial density state fidelity of $F = 0.96325$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

fidelity, whereas a larger number of time intervals leads to increasingly slow rates of convergence, confirming the hypothesis. In the case where $N = 100$, a tenfold increase compared to $N = 10$, the gradient decreases accordingly. This is analogous to a tenfold decrease in step size (see Figure 3.1 for $\epsilon = 10^{14}$). A more detailed analysis of the fidelity curves during the first few iterations leads to the conclusion that the optimization with $N = 5$ exhibits a higher rate of convergence compared to the case where $N = 10$. Although this is in accordance

with the aforementioned hypothesis, it does not explain why the former results in a lower final density state fidelity. A reasonable explanation would be the lack of degrees of freedom that inhibits convergence to a better optimum. Also note that setting $N = 1$ restricts the algorithm to only 2 degrees of freedom — two constant electron and nuclear spin driving amplitudes, resulting in no improvement at all. In fact, the initial guess outperformed the GRAPE algorithm for $N = 1$ (constant amplitude) and $\epsilon = 10^{15}$. The fidelity breakdown is probably due to the disproportional step size causing similar behaviour as can be seen in Figure 3.1 for $\epsilon = 10^{16}$ (a tenfold increase). In conclusion, $N = 10$ appears to yield the most promising result and shall therefore be selected for the optimization of a suitable pulse length.

3.2.4 Pulse length variation

For the sake of clarity, this section is divided into two parts — one for pulse lengths shorter than 50 ns and one for those longer than 50 ns. Starting with the latter, we set $N = 10$ and $\epsilon = 10^{15}$ as discussed previously. Since the initial driving amplitude of 10 MHz corresponds to an exact duration of $T = 50$ ns (π - pulse), we expect that longer durations will cause increasingly poor initial density state fidelities. A duration of $T = 100$ ns (which is in fact a 2π - pulse) is expected to rotate the spin back to its initial state, yielding an initial density state fidelity near to 0. Figure 3.3 shows fidelity curves corresponding to 6 different pulse lengths including the optimization of the 50 ns pulse that has already been shown in the preceding two figures. Note that only the first 50 iterations are shown, for this region provides interesting information on fidelity convergence. As expected, the initial density state fidelities decrease for increasing pulse durations down to a minimum near to 0 for $T = 100$ ns. Remarkably, increasing pulse durations exhibit faster rates of convergence and also yield higher final density state fidelities compared to the optimization of a 50 ns pulse within the first 20 iterations. In fact, the fidelity curves terminate in reverse order with respect to the initial density state fidelity. The longest duration of $T = 100$ ns appears to yield the most promising result within the range of pulse durations longer than or equal to 50 ns. Analogous to the clarification given in the preceding section, this behaviour is probably due to the unsuitable step size of $\epsilon = 10^{15}$ for certain pulse lengths. Since the gradient of the performance function is proportional to Δt and thus to T, longer durations result in higher convergence rates but also increase the probability of overshooting an optimum. Why is the latter not observed in this case? Apparently, for a fidelity breakdown to occur, doubling the duration T is insufficient as can be observed also in Figure 3.2 for the case where $N = 5$.

We shall continue with the analysis of shorter pulse durations, which are in fact preferable due to the reduced risk of decoherence that increases with time. Again, we set $N = 10$ and $\epsilon = 10^{15}$. Figure 3.4 shows fidelity curves corresponding to 5 different pulse lengths including the optimization of the 50 ns pulse. As predicted by the hypothesis, optimization of shorter pulses exhibits increasingly low rates of initial convergence and poor final density state fidelities.

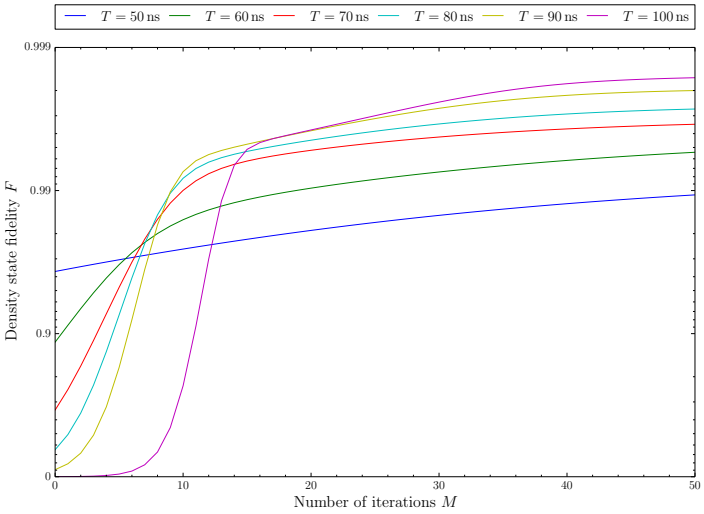

Figure 3.3: Density state fidelity plot on a single inverse logarithmic scale for 6 different pulse lengths longer than or equal to 50 ns with $N = 10$ and $\epsilon = 10^{15}$. Note that in this case, all graphs have different initial density state fidelities lower than $F = 0.96325$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 50$.

None of these optimizations outperform the preceding results of greater pulse lengths. Nevertheless, the curve corresponding to a pulse length of $T = 20\,\text{ns}$ appears to maintain its convergence rate throughout all iterations. This interesting phenomenon requires further investigation that shall be discussed under the subsequent section of optimal control.

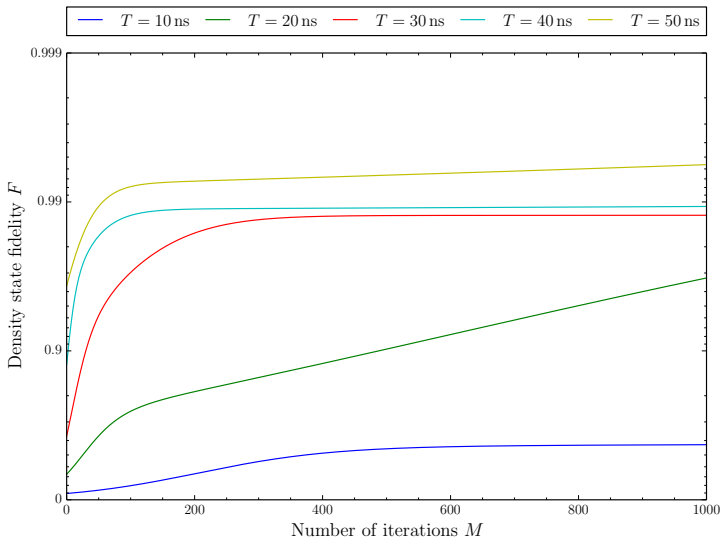

Figure 3.4: Density state fidelity plot on a single inverse logarithmic scale for 5 different pulse lengths shorter than or equal to 50 ns with $N = 10$ and $\epsilon = 10^{15}$. Note that in this case, all graphs have different initial density state fidelities lower than $F = 0.96325$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

3.2.5 Optimal control

This section continues on the preceding series of optimizations by further investigation of the 20 ns pulse improvement, since it is preferable to control pulses of longer duration and seems to have a promising convergence rate. For this reason it is interesting to verify whether the trend of constant convergence continues as the number of iterations increases or whether it exhibits alternative behaviour. Secondly, we are interested in the final density state fidelity of the acquired control pulse. Figure 3.5 shows the 50 ns pulse density state fidelity plot for a maximum of 5000 iterations. Although the constant rate of convergence is not

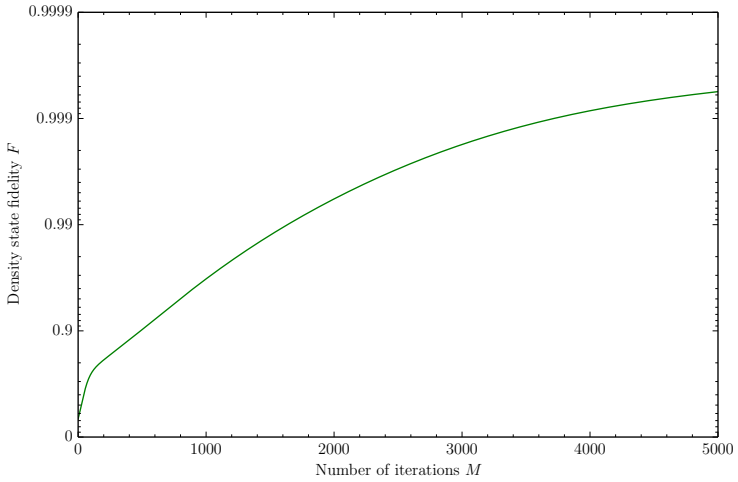

Figure 3.5: Density state fidelity plot on a single inverse logarithmic scale for a 20 ns pulse with $N = 10$ and $\epsilon = 10^{15}$. Note that the rate of convergence diminishes as the number of iterations increases. The maximum number of iterations shown is $M = 5000$ at which a final density state fidelity of $F = 0.99944$ is achieved.

longer maintained, the final density state fidelity of $F = 0.99944$ outperforms all of preceding optimization results.

Now that the optimal parameters have been determined and an optimal final density state fidelity has been obtained, we proceed with the analysis of the actual control pulses that correspond to this last optimization. Figure 3.6 shows the electron and nuclear spin driving amplitudes of the 20 ns pulse. First note that the electron spin driving amplitude reaches a maximum of approximately 70 MHz, whereas the nuclear spin driving amplitude does not exceed $5 \cdot 10^{-9}$ Hz. Apparently, strongly driving the nuclear spin transitions with the fixed driving frequency Q is not required for this specific state preparation. Indeed, in driving from the initial state $m_s = -1$, $m_N = 0$ to the target state $m_s = 0$, $m_N = 0$, the nuclear spin quantum number remains 0, so there is no need for this form

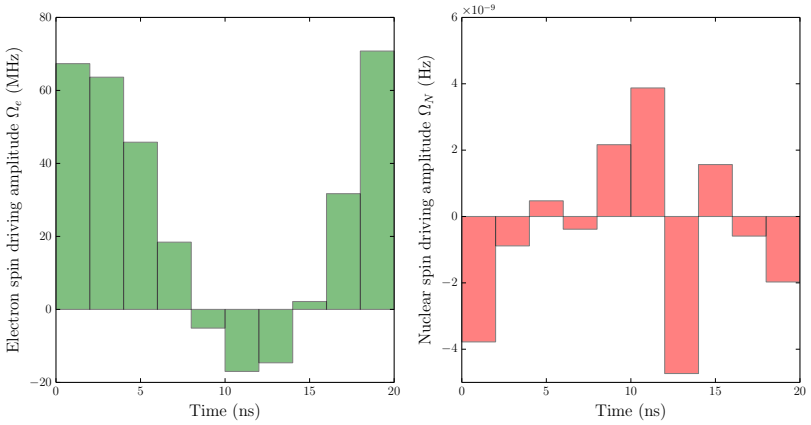

Figure 3.6: Electron and nuclear spin driving amplitudes corresponding to the optimized 20 ns pulse. Initial controls of constant amplitude were set to 10 MHz and 0 Hz for the electron and nuclear spin driving amplitude respectively.

nuclear spin control.

Of further interest is the time evolution of the eigenstate probabilities of the spin system subject to this optimized pulse, although the relatively high final density state fidelity already is a strong indication for a successful quantum state preparation. Figure 3.7 shows the time evolution for the $m_N = 0$ states of a spin system subject to the optimized 20 ns pulse. Other eigenstate probabilities, which are in fact negligible due to the lack of nuclear spin control, are not shown in this graph. Comparing this result to non-optimized Rabi oscillation based control (as discussed under Section 2.1.3) leads to the observation that the GRAPE algorithm has outperformed the former fidelity of $F = 0.96325$ with $F = 0.99944$.

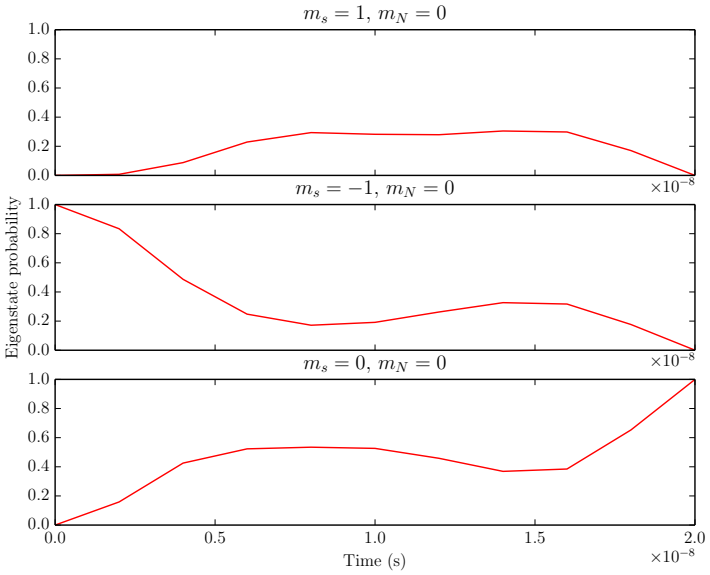

Figure 3.7: Time evolution of the eigenstate probabilities for the $m_N = 0$ states. The system is subject to an optimized 20 ns pulse with $N = 10$. Note the transition from the initial state $m_s = -1$, $m_N = 0$ to the target state $m_s = 0$, $m_N = 0$. The final density state fidelity equals $F = 0.99944$.

3.3 Synthesis of unitary transformations

3.3.1 Overview

Throughout this section, constructing optimal control pulses for the unitary transformations of a C-NOT and an unconditional π - gate (as discussed under Section 2.1.3) shall be of exclusive interest. Within the context of non-optimized control, a 395 ns pulse of 1.26 MHz and a 50 ns pulse of 10 MHz constitute these unitary transformations respectively and shall therefore be used as an initial guess. Analogous to the approach under the preceding section, determination of a suitable step size shall be of initial concern, followed by the selection of a number of time intervals and a proper pulse duration. In each of these optimizations, the C-NOT and unconditional π - gates shall be discussed separately and selections shall be made through the analysis of fidelity plots in which the gate fidelities are shown as a function of the number of iterations of the GRAPE algorithm.

3.3.2 Determination of a suitable step size

Controlled-NOT gate

In order to determine a suitable step size, initial algorithm parameters must be chosen. Figure 3.8 shows fidelity curves corresponding to 4 different step

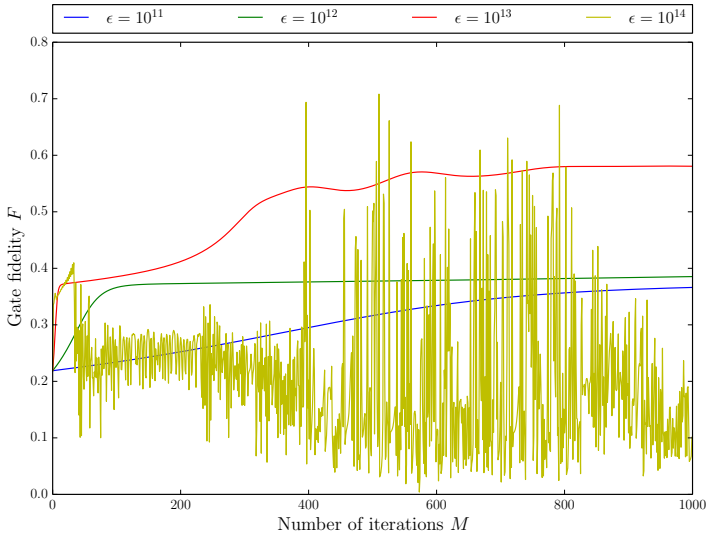

Figure 3.8: Gate fidelity plot for a C-NOT gate on a linear scale for 4 different step sizes with $N = 10$ and $T = 395$ ns. Note that all graphs start at the same initial density state fidelity of $F = 0.21884$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

sizes for $N = 10$ and $T = 395\,\text{ns}$, which is similar to the result in Figure 3.1 (quantum state preparation). Increasing step sizes lead to faster rates of convergence up to and including $\epsilon = 10^{13}$. Although a step size of $\epsilon = 10^{14}$ yields occasional optima that outperform other fidelity curves, its convergence is highly unpredictable and certainly not monotone. For the C-NOT gate, a step size of $\epsilon = 10^{13}$ appears to yield the the most promising result and shall therefore be selected for the optimization of a suitable time interval. Note that this step size differs from the one corresponding to quantum state preparation by a factor 100.

Unconditional π-gate

In the case of an unconditional π-gate, we set $N = 10$ and $T = 50\,\text{ns}$. Figure 3.9 shows fidelity curves corresponding to 3 different step sizes for $N = 10$ and $T = 50\,\text{ns}$. Again, a similar behaviour is observed. Whereas a step size of

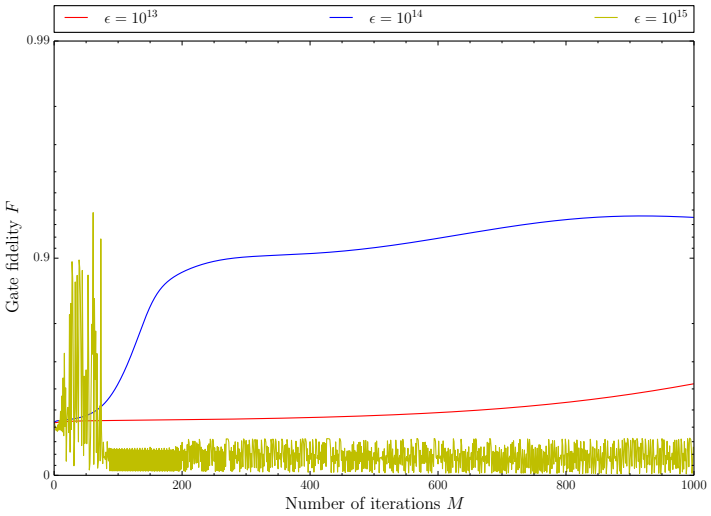

Figure 3.9: Gate fidelity plot for an unconditional π-gate on a single inverse logarithmic scale for 3 different step sizes with $N = 10$ and $T = 50\,\text{ns}$. Note that all graphs start at the same initial density state fidelity of $F = 0.42702$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

$\epsilon = 10^{15}$ gives rise to an unreliable optimization, a step size of $\epsilon = 10^{14}$ yields the most promising result. Remarkably, it differs from the C-NOT optimal step size by a factor 10. Apparently, optimizing different unitary transformations requires different step sizes.

Another difference between the two unitary transformations is the final density state fidelity. In the case of the C-NOT gate, a step size of $\epsilon = 10^{13}$ yields

at most $F = 0.58091$, whereas optimization for the unconditional π-gate with $\epsilon = 10^{14}$ yields a higher fidelity of $F = 0.936$. This is possibly due to the different properties of the transformations. As aforementioned, the unconditional π-gate swaps the $m_s = -1$ states with those of $m_s = 0$ unconditionally, i.e. without taking the nuclear spin into account. On the contrary, the C-NOT gate is conditional on the nuclear spin state, which requires a very specific rotation. Simply driving with an on resonance driving frequency corresponding to the desired rotation will affect all of nearby energy spin states with relatively small splittings — the nuclear spin splittings. From this point of view, it is not surprising that constructing a C-NOT gate is not as straightforward as is the case with an unconditional π-gate. For this reason, a higher number of iterations may be needed for a C-NOT control pulse to outperform the current optimized unconditional π-pulse. We proceed with the selection of a suitable number of time intervals.

3.3.3 Time interval selection

Controlled-NOT gate

In this part, we set $\epsilon = 10^{13}$ and $T = 395\,\text{ns}$. Figure 3.10 shows fidelity curves

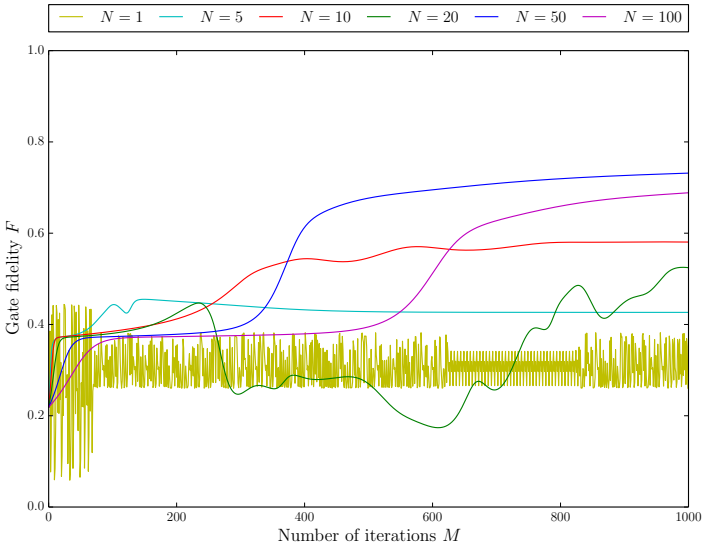

Figure 3.10: Gate fidelity plot on a linear scale for 6 different numbers of time intervals with $\epsilon = 10^{13}$ and $t = 395\,\text{ns}$. Note that all graphs start at the same initial gate fidelity of $F = 0.21884$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

corresponding to 6 different numbers of intervals. As in the preceding results

for quantum state preparation, we observe higher convergence rates for increasingly few time intervals during the first 50 iterations. This is probably due to the unitary transformation gradient that contains the same factor $\Delta t = {}^{T}/_{N}$ (see Theorem 2.2.9). An interesting difference however, is that many curves in this optimization are not monotone. This proves again that a fixed step size limits the results of the GRAPE algorithm, especially in the case of C-NOT gate optimization. Probably, the lack of an unambiguous optimum causes the performance function to be highly variable leading to strongly varying gradients that require an iteration dependent step size. The fidelity curve that both exhibits monotone convergence and yields the highest final gate fidelity results from the optimization with $N = 50$; remarkable, since the preceding step size optimization was performed for $N = 10$. This may be another result of the strongly varying gradient in this case. Before proceeding to the section of pulse length variation, we shall first discuss the time interval selection for the unconditional π - gate.

Unconditional π - gate

As concluded in the preceding section, the optimal step size for the optimization of the unconditional π - gate equals $\epsilon = 10^{14}$. Figure 3.11 shows fidelity curves

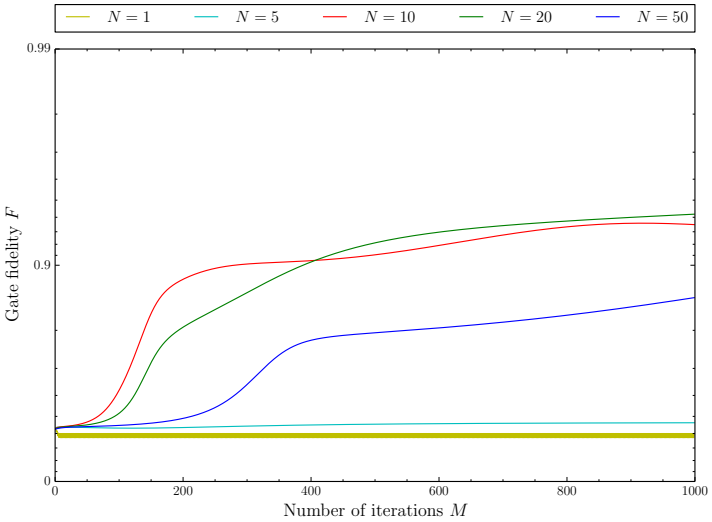

Figure 3.11: Gate fidelity plot on a single inverse logarithmic scale for 5 different numbers of time intervals with $\epsilon = 10^{14}$ and $T = 50\,\text{ns}$. Note that all graphs start at the same initial gate fidelity of $F = 0.42702$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

corresponding to 5 different numbers of intervals. An immediate observation is the monotone convergence of nearly all fidelity curves in contrast with the

optimization of the C-NOT gate. Analogous to the time interval selection of the quantum state preparation, we see that for $N > 10$ convergence rates decrease during the first 200 iterations. For $N = 1$, the curve oscillates around a constant gate fidelity, indicating that the step size is too large and for $N = 5$, the lack of degrees of freedom impedes convergence to a high final gate fidelity. Optimization with $N = 20$ appears to yield the the highest final gate fidelity and shall therefore be selected for the optimization of a suitable pulse duration.

3.3.4 Pulse length variation

Controlled-NOT gate

The forthcoming optimizations include time length variations of at most 50 ns. In the first case of the C-NOT gate, where $N = 50$ and $\epsilon = 10^{13}$, the result of optimizing 5 different pulse durations is shown in Figure 3.12. The first remark-

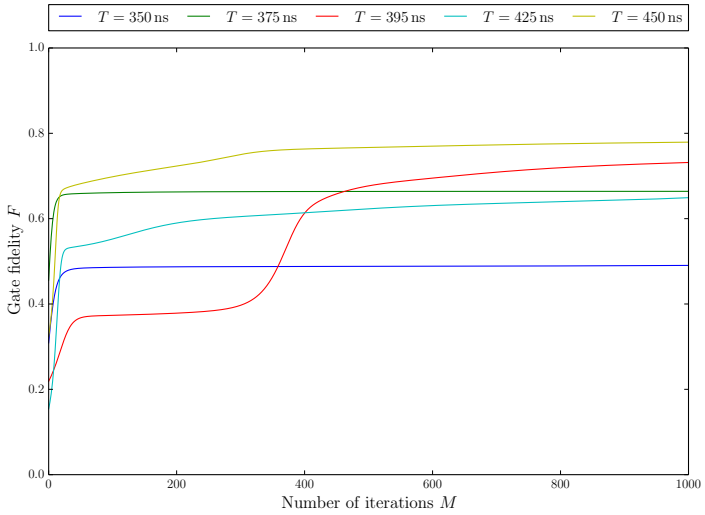

Figure 3.12: Gate fidelity plot on a linear scale for 5 different pulse lengths around 395 ns with $N = 50$ and $\epsilon = 10^{13}$. Note that almost all graphs have initial fidelities higher than $F = 0.21884$ (Rabi oscillation based control), which is shown in red. The maximum number of iterations shown is $M = 1000$.

able thing to note is that all fidelity curves (except for the case where $T = 425$) have higher initial gate fidelities than the 395 ns Rabi oscillation based control pulse. Apparently, simply altering the length of the original control pulse has a positive effect on the fidelity of the C-NOT gate. Then why is the 395 ns pulse still used? The answer lies in the definition of the gate fidelity. Unless an optimized unitary transformation has a gate fidelity equal to 1, there are always matrix elements that cause poor fidelities. However, many elements corresponding to unimportant spin state transitions of the C-NOT transformation matrix

(within $m_s = 1$ for example) are not required to perfectly equal those of the target transformation matrix. Since the performance function is defined for arbitrary transformations, it treats the optimization of all elements equally. This may cause poor gate fidelities even though the corresponding pulse performs well in actual practice. Better definitions of a performance function may thus lead to more intuitive and predictable results enabling better optimization of a specific unitary transformation, such as the C-NOT gate. We proceed with the analysis of Figure 3.12. As expected, optimization of the 425 ns and 450 ns pulses shows high rates of convergence not only within the first iterations, but throughout all numbers of iterations compared to the optimization of the two shortest pulse lengths. As aforementioned, the reason is that longer durations T lead to higher gradients through the factor Δt. Eventually, the highest final gate fidelity of $F = 0.77929$ is reached through optimizing the 450 ns pulse, which shall be discussed in greater detail under the subsequent section of optimal control.

Unconditional π - gate

Analogous to the corresponding section of quantum state preparation, this section is divided into two parts — one for pulse lengths shorter than 50 ns and one for those longer than 50 ns. Regarding the pulse length optimization of the unconditional π - gate, better insight into several phenomena can be obtained in this way. As concluded in the preceding sections, optimal parameters in this case are $\epsilon = 10^{14}$ and $N = 20$. Figure 3.13 shows fidelity curves corresponding to 5 different pulse lengths including the optimization of the 50 ns pulse that has already been shown in Figure 3.11. First note that almost all initial gate fidelities exceed that of the 50 ns pulse due to the same reason as discussed in the preceding optimization of the C-NOT gate. Secondly, all fidelity curves exhibit monotone convergence, since the shorter pulse lengths lead to smaller gradients. It is thus less likely to overshoot an optimum causing a fidelity decrease.

For pulse durations longer than or equal to 50 ns, Figure 3.14 shows 6 different gate fidelity curves. Since longer pulses lead to higher gradients, we expect to see (partially) decreasing fidelity curves. Indeed, for the two longest pulses of 90 ns and 100 ns, fidelity breakdown occurs. Another phenomenon is the highly variable rate of convergence, which is due to the fixed step size used in the GRAPE algorithm. The highest final gate fidelity found in these optimizations equals $F = 0.98414$ and is achieved through optimization of the 80 ns pulse, which shall be discussed in greater detail under the subsequent section of optimal control.

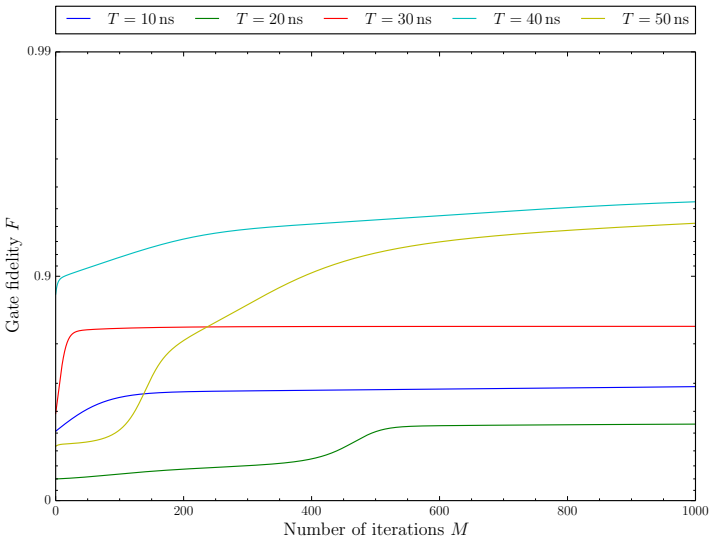

Figure 3.13: Gate fidelity plot on a single inverse logarithmic scale for 5 different pulse lengths shorter than or equal to 50 ns with $N = 20$ and $\epsilon = 10^{14}$. Note that almost all graphs have initial gate fidelities higher than $F = 0.42702$ (Rabi oscillation based control), which is shown in yellow. The maximum number of iterations shown is $M = 1000$.

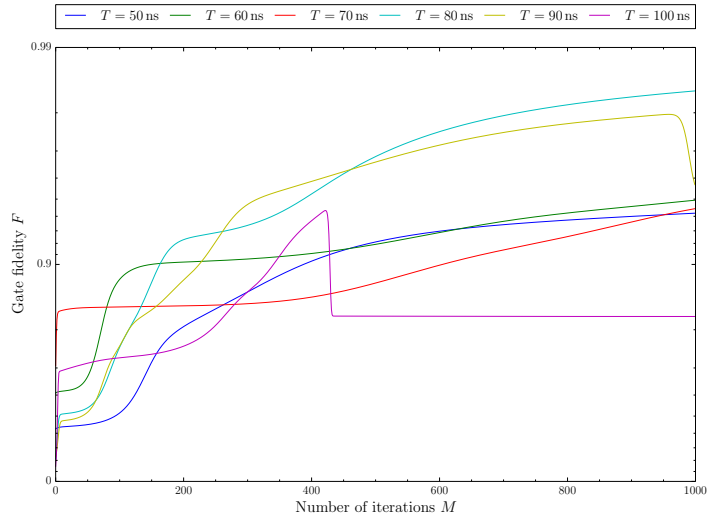

Figure 3.14: Gate fidelity plot on a single inverse logarithmic scale for 6 different pulse lengths longer than or equal to 50 ns with $N = 20$ and $\epsilon = 10^{14}$. The initial gate fidelity of the 50 ns pulse equals $F = 0.42702$ (Rabi oscillation based control). The maximum number of iterations shown is $M = 1000$.

3.3.5 Optimal control

Controlled-NOT gate

In the preceding section, the optimal control pulse for the purpose of a C-NOT gate was found using parameters of $\epsilon = 10^{13}$ and $N = 50$. Figure 3.15 shows the electron and nuclear spin driving amplitudes of the corresponding 450 ns pulse. Similar to the optimized pulse in the case of quantum state preparation, we

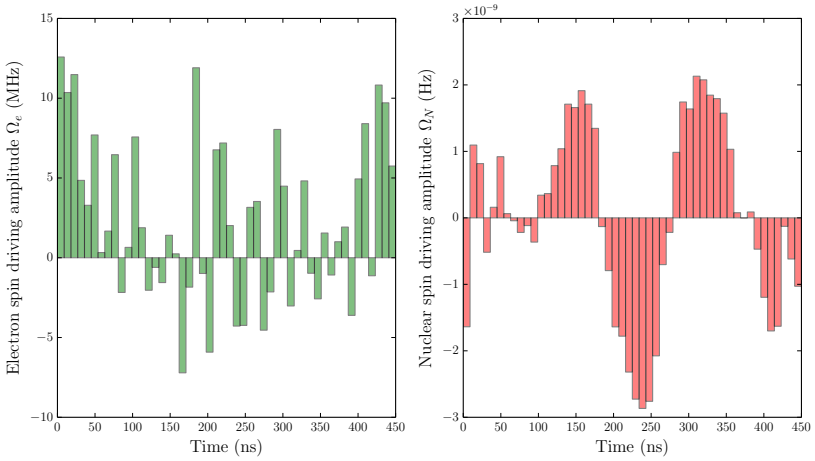

Figure 3.15: Electron and nuclear spin driving amplitudes corresponding to the optimized 450 ns pulse. Initial controls of constant amplitude were set to 1.26 MHz and 0 Hz for the electron and nuclear spin driving amplitude respectively.

observe that the nuclear spin driving amplitude remains within a range of several nanohertz. This indicates that for an optimal C-NOT pulse, strongly driving the nuclear spin transitions is unnecessary. Indeed, no nuclear spin transitions are required for the unitary transformation of a C-NOT, since the only target transitions are between the $m_s = -1$, $m_N = -1$ and $m_s = 0$, $m_N = -1$ states. As for the electron spin driving amplitude, we observe highly irregular behaviour. As could also be concluded from the fidelity plots in the preceding section, constructing an optimal C-NOT pulse is not as straightforward as is the initial guess of a constant amplitude. Nevertheless, this optimized pulse with a fidelity of $F = 0.77929$ clearly outperforms the Rabi oscillation based control pulse result of $F = 0.21884$.

As aforementioned, a gate fidelity that does not equal 1 provides no information on the transformation matrix structure. In order to gain further insight into this result, we assume that the$m_N = 0, -1$ states of $m_s = 0, -1$, which can be used as a quantum register, are of exclusive interest. By calculating the gate fidelity of the 4×4 submatrix corresponding to these transitions, more

information is obtained on the performance of this specific transformation. It appears that in this case, the corresponding fidelity of $F = 0.33274$ is lower compared to the same subspace fidelity of the Rabi oscillation based transformation matrix ($F = 0.50430$). Note again that the incompatibility of the unspecific performance function has lead to these erroneous results. Although the analysis of transformation specific performance functions is beyond the scope of this research, it would highly be recommended as a subject for future projects.

Unconditional π - gate

In the preceding section, the optimal control pulse for the purpose of an unconditional π - gate was found using parameters of $\epsilon = 10^{14}$ and $N = 20$. Figure 3.16 shows the electron and nuclear spin driving amplitudes of the corresponding 80 ns pulse. Although the nuclear spin driving amplitude has increased tenfold

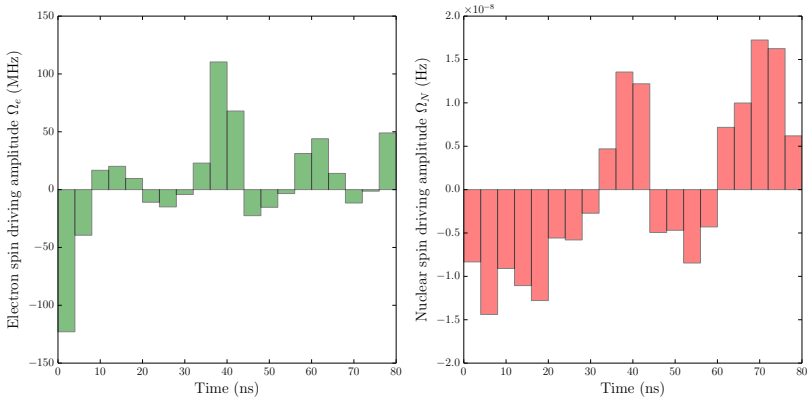

Figure 3.16: Electron and nuclear spin driving amplitudes corresponding to the optimized 80 ns pulse. Initial controls of constant amplitude were set to 10 MHz and 0 Hz for the electron and nuclear spin driving amplitude respectively.

with respect to the optimized quantum state preparation and C-NOT pulses, it is still negligible. The unconditional π - gate does not require nuclear spin state transitions. As for the electron spin driving amplitude, a periodicity can be observed. In conclusion, this optimized pulse with a fidelity of $F = 0.98414$ outperforms the Rabi oscillation based control pulse result of $F = 0.42702$.

From a theoretical point of view, there is no reason to reject this result. In practice however, there is one major downside to this control pulse that we shall try to resolve next. The excessive electron spin driving amplitude (> 15 MHz) prevents this pulse from being applied in a real experimental setting. Throughout the preceding sections, a systematic method of finding the optimal algorithm parameters was introduced and applied. In each optimization, the initial guess was set to a relatively short Rabi oscillation based control pulse and no attention was paid to the experimental applicability of the optimized result. In order

to improve this, we need to ask ourselves the following question: What may have caused this pulse to have reached these relatively high electron driving amplitudes in comparison with the optimized pulse for the C-NOT gate? It was possibly due to the higher initial guess of 10 MHz with respect to 1.26 MHz. Perhaps another reason is the limited time length (80 ns) of the pulse, forcing the algorithm to construct pulses of high amplitude. Under the preceding section, pulse length variations of at most 50 ns were discussed, where relatively long pulse lengths resulted in a fidelity breakdown due to the incompatible fixed step size of $\epsilon = 10^{14}$. For these reasons, we perform the optimization of the unconditional π - gate once more by setting a low step size of $\epsilon = 10^{12}$, a long duration of $T = 1000$ ns and a number[1] of time intervals $N = 100$ with initial electron and nuclear spin driving amplitudes of 0 Hz. The number of iterations was set to $M = 2000$. The resulting electron and nuclear spin driving amplitudes of the corresponding optimal control pulse are shown in Figure 3.17. Interestingly, the

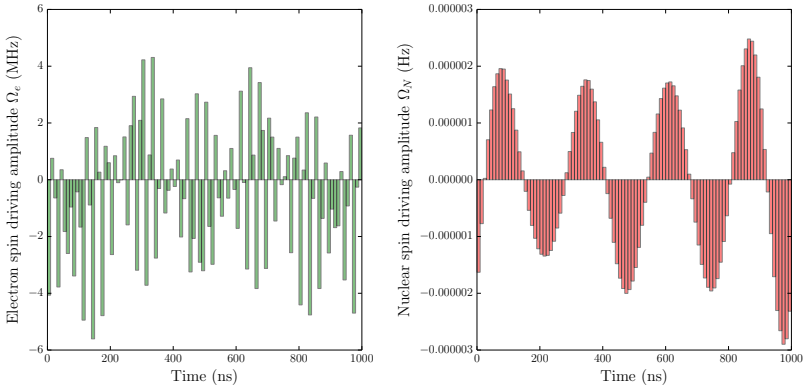

Figure 3.17: Electron and nuclear spin driving amplitudes corresponding to the optimized 1000 ns pulse. Initial controls were set to 0 Hz for both electron and nuclear spin driving amplitudes.

problem of excessive driving amplitudes is solved, as predicted by the hypothesis. Furthermore, the nuclear spin driving amplitude has increased by a factor 100 compared with the nuclear driving amplitude of the C-NOT gate, although it is still negligible. Perhaps the best result however, is the gate fidelity that corresponds to this pulse. During the optimization, the gate fidelity exhibited monotone convergence and reached a final value of $F = 0.99413$, which even outperforms the previous result for the unconditional π - gate.

[1]Such that the time interval length of 10 ns approximately equals the one of the C-NOT gate, which is 9 ns.

3.4 Decoherence-based optimization

The effect of decoherence and a method for its simulation were previously discussed under Section 2.3. This section elucidates the effect of decoherence on a standard optimized pulse as well as on a pulse constructed through decoherence-based optimization for the case of quantum state preparation. Consider the optimized pulse of the quantum state preparation as discussed under Section 3.2.5. Without taking into account the effect of decoherence, the final density state fidelity of this pulse was found to be $F = 0.99944$. Applying the decoherence-based optimization with the same parameters of $\epsilon = 10^{15}$, $N = 10$, $T = 20\,\text{ns}$, $M = 5000$ and $L = 200$ (number of performance functions) yields the pulse as shown in Figure 3.18. Remarkably, the density state fidelity corresponding to

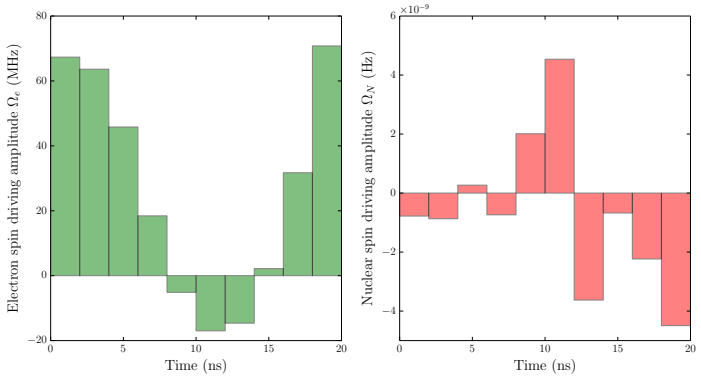

Figure 3.18: Electron and nuclear spin driving amplitudes corresponding to the decoherence-based optimization of the 20 ns pulse. Initial controls of constant amplitude were set to 10 MHz and 0 Hz for the electron and nuclear spin driving amplitude respectively. The corresponding fidelity equals $F = 0.99944$.

this pulse is $F = 0.99944$, which is equal to the original optimization fidelity. However, in comparison with the original pulse in Figure 3.6, the nuclear spin driving amplitude appears to be different. Nevertheless, it remains within the order of $10^{-9}\,\text{Hz}$ and therefore negligible. Moreover, differences of this magnitude cannot even be seen for the electron driving spin amplitude, which appears to be very similar to the original pulse. When subjecting both control pulses to the effect of decoherence, the differences become clear. Figure 3.19 shows three histograms in which the final eigenstate probabilities of 1000 evolutions based on randomly sampled magnetic fields are shown for the $m_N = 0$ states in the case of the standard and decoherence-optimized pulse. First note the spread of eigenstate probabilities due to the effect of decoherence. Although both methods show a great amount of overlap, the decoherence-optimized pulse consistently results in less spread distributions of eigenstate probabilities. As expected, this indicates that its result is less sensitive to the effect of decoher-

ence. Secondly, if we consider again the evolution of the originally optimized pulse without the effect of decoherence, i.e. an evolution subject to a magnetic field of B_z, the final eigenstate probabilities equal $3.171 \cdot 10^{-5}$, $5.268 \cdot 10^{-4}$ and 0.9994 for $m_s = 1$, $m_s = -1$ and $m_s = 0$ respectively. In the histograms of Figure 3.19, these values lie in the outer left bar ($m_s = 1$), in the centre of the distribution ($m_s = -1$) and in the outer right bar ($m_s = 0$) respectively. From this observation, we can conclude that the average fidelities must have decreased due to the effect of decoherence, which is the case indeed. The mean fidelities of these distributions equal $F = 0.99939$ and $F = 0.99940$ for the standard and decoherence-optimized pulse respectively, both lower than $F = 0.99944$. Due to the relatively high fidelity of these pulses, the final eigenstate probabilities lie close to either 0 or 1. Decoherence will almost certainly have a negative effect on the eigenstate probabilities, since there is simply only one way of deterioration — for the probabilities close to 0 in the positive direction and for the probabilities close to 1 in the negative direction. Returning to the comparison of both methods, we conclude that the decoherence-optimized pulse reaches a higher final density state fidelity, although the decisiveness of this result is speculative. Errors due to the finite sampling from normal distributions may also have caused this fidelity difference. Furthermore, if this minor fidelity improvement were not due to these errors, but to the actual beneficial algorithm adjustment, it cannot outweigh the undesirable time cost of constructing these decoherence-optimized control pulses.

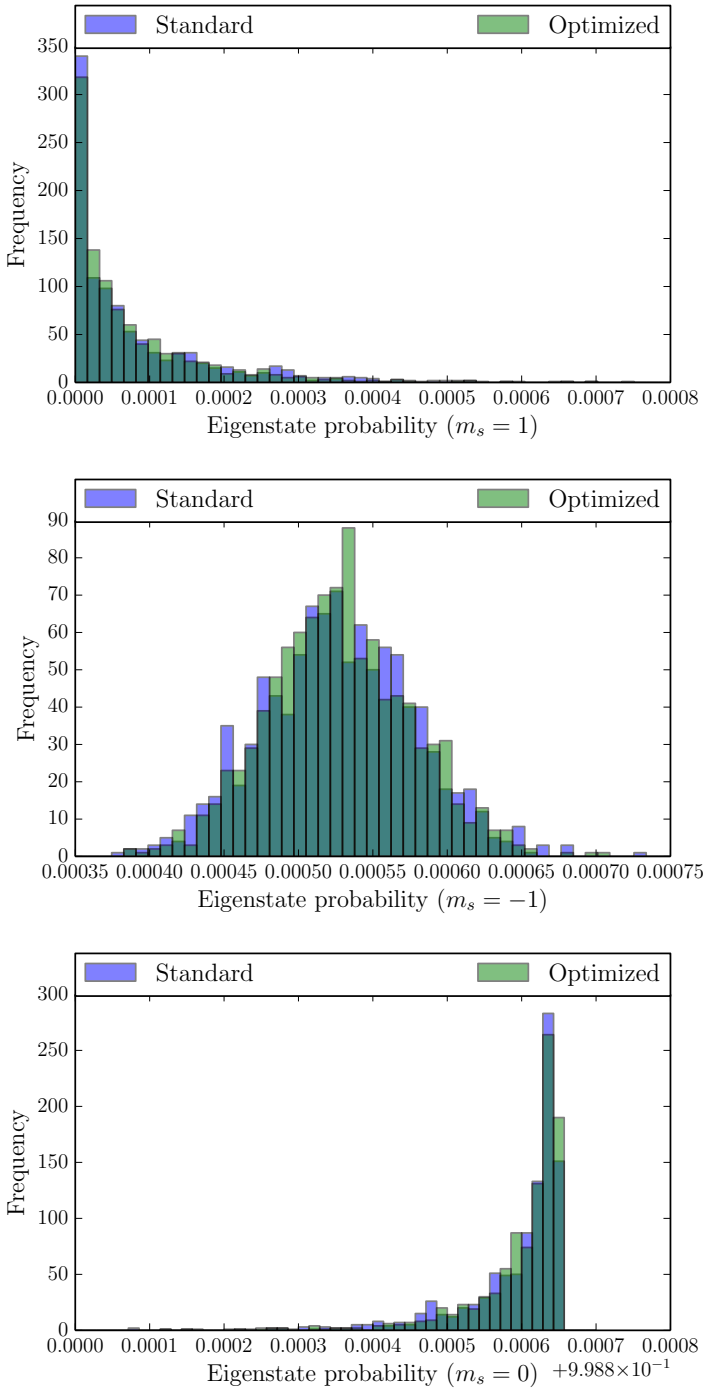

Figure 3.19: Histograms of final eigenstate probabilities of 1000 evolutions based on randomly sampled magnetic fields for the $m_N = 0$ states in the case of the standard (purple) and decoherence-optimized pulse (green). Note the spread due to the effect of decoherence.

3.5 Parallel computation

Parallel computation entails the simultaneous use of more than one CPU or processor core to execute a program or multiple computational threads. Many simulations performed in this research are time-consuming due to extensive calculations and iterative processes. For this reason, the principles of parallel computation are applied to calculations that do not necessarily require sequential processing in order to reduce the computation time.

Although the iterative GRAPE algorithm itself must be executed sequentially, processes such as the calculation of independent propagators (Equation 2.15) and the computation of multiple evolutions for the purpose of decoherence-based optimization (Section 3.4) provide the opportunities for this parallelization. An important variable concerning parallel computation is the number of parallel processes or threads, which needs to be determined experimentally in order to maximally reduce the computation time. As an example, we consider the computation of 1000 independent time evolutions for corresponding magnetic fields and a varying number of processes. Figure 3.20 shows a significant decrease of computation time as the number of processes increases up to and in-

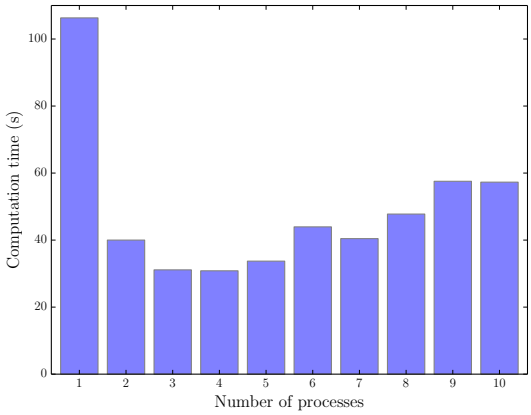

Figure 3.20: Computation time as a function of the number of processes corresponding to the calculation of 1000 independent time evolutions. For the optimal number of processes (4), the computation time is reduced by 71% (compared with sequential processing).

cluding 4, at which the algorithm runs 71% faster in comparison with sequential processing. Interestingly, the optimal number of processes equals the number of cores in the Intel® Core™ i7-2630QM processor on which all simulations were performed. The use of more than 4 processes leads to increasing computation times, probably due to the fact that additional processes are virtual and run on one of the same four processor cores, causing a slowdown in comparison with the optimal parallel computation.

Chapter 4

Conclusions

4.1 General remarks

Throughout this work, the GRAPE algorithm was applied to three examples of electron-nuclear spin control — a quantum state preparation and the implementation of a controlled-NOT and unconditional π - gate. We firstly conclude that the fixed step size in the GRAPE algorithm limits the monotone and fast convergence of optimized control pulses. A gradient- or time-dependent step size may resolve this issue and thus lead to higher final fidelities. Perhaps equally as important is the unspecific performance function for the synthesis of unitary transformations which may also lead to erroneous optimization results. Furthermore, the number of degrees of freedom in the algorithm, i.e. the number of time intervals, may not be too low if a high final fidelity is to be achieved. Whereas the crude optimization of algorithm parameters throughout this work may have yielded local optima, better results can almost surely be found by more systematic search methods. As for all optimized control pulses, which will be mentioned subsequently, the nuclear spin driving amplitude was negligible. The following sections summarize the conclusions for quantum state preparation, synthesis of unitary transformations, decoherence-based optimization, parallel computation and the outlook concerning this research.

4.2 Quantum state preparation

For the case of quantum state preparation, an optimal pulse was acquired with parameters of $\epsilon = 10^{15}$, $N = 10$, $T = 20\,\text{ns}$ and $M = 5000$, yielding a final density state fidelity of $F = 0.99944$, which outperforms the Rabi oscillation based control fidelity of $F = 0.96325$.

4.3 Synthesis of unitary transformations

For the case of a controlled-NOT gate, an optimal pulse was acquired with parameters of $\epsilon = 10^{13}$, $N = 50$, $T = 450\,\text{ns}$ and $M = 1000$, yielding a final density state fidelity of $F = 0.77929$, which outperforms the Rabi oscillation based control fidelity of $F = 0.21884$. For the case of an unconditional π - gate, an optimal pulse was acquired with parameters of $\epsilon = 10^{12}$, $N = 100$, $T = 1000\,\text{ns}$ and $M = 2000$, yielding a final density state fidelity of $F = 0.99143$, which outperforms the Rabi oscillation based control fidelity of $F = 0.42702$. The problem of excessive driving amplitudes was solved by setting low initial controls and a relatively long pulse duration.

4.4 Decoherence-based optimization

Implementing the effect of decoherence in the GRAPE algorithm yields no significant average fidelity improvement, but has a positive effect on reducing the spread of final density states for the case of quantum state preparation.

4.5 Parallel computation

The principles of parallel computation that were applied to suitable calculations in the GRAPE algorithm have proved to yield positive results. Compared with sequential processing, the computation time of a single characteristic calculation reduced by 71% for the optimal number of processes (4).

4.6 Outlook

Future research on this subject should primarily include the implementation of a variable step size in order to improve the monotone convergence of fidelity curves. As aforementioned, more systematic search methods of algorithm parameters would also improve the results. Obviously, a tremendous amount of research could be performed on the optimization of many unitary transformations and the experimental implementation of them in an operational NV centre. Remaining problems that require light to be shed on include the adaptation of performance functions to a specific target transformation and the analysis of strong decoherence effects on final density state distributions and fidelities.

Bibliography

[1] Bernien, H. (2014). *Control, measurement and Entanglement of Remote Quantum Spin Registers in Diamond*. Delft University of Technology.

[2] Bland, J. (2002). *A Mössbauer Spectroscopy and Magnetometry Study of Magnetic Multilayers and Oxides*. University of Liverpool.

[3] Le Bellac, M. (2006). *A Short Introduction to Quantum Information and Quantum Computation*. Cambridge University Press, Cambridge.

[4] Nielsen, M. A. and Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge.

[5] Said, R. S. and Twamley, J. (2009). *Robust control of entanglement in a nitrogen-vacancy center coupled to a ^{13}C nuclear spin in diamond*. Physical Review A 80, 032303.

[6] Khaneja, N. et al. (2005). *Optimal control of coupled spin dynamics: design of NMR pulse sequences by gradient ascent algorithms*. Journal of Magnetic Resonance 172, 296-305.

[7] Horn, R. A. and Johnson, C. R. (1990). *Norms for Vectors and Matrices*. Ch. 5 in Matrix Analysis. Cambridge University Press, Cambridge.

[8] Bellman, R. (1960). *Introduction to Matrix Analysis*. McGraw-Hill, New York.

[9] Snider, R. F. (1964). *Perturbation variation methods for a quantum Boltzmann equation*. Journal of Mathematical Physics, 5, 1580-1587.

[10] Wilcox, R. M. (1967). *Exponential operators and parameter differentiation in quantum physics*. Journal of Mathematical Physics, 8, 962-982.

[11] Bacon, D. M. (1997). *Decoherence, Control, and Symmetry in Quantum Computers*. California Institute of Technology.

Appendices

Appendix A

Spin-1 operators

The spin-1 operators are a set of three 3×3 complex matrices which are Hermitian and unitary. Each of these operators corresponds to an observable describing the spin of a spin-1 system in each of the three spatial directions. The formal definitions are given below[1].

$$S_x = \frac{\hbar}{\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$S_y = \frac{\hbar}{\sqrt{2}} \begin{pmatrix} 0 & -i & 0 \\ i & 0 & -i \\ 0 & i & 0 \end{pmatrix}$$

$$S_z = \hbar \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

[1]For the sake of readability and numerical simulation, the reduced Planck constant $\hbar$ is usually set to 1. As a consequence, energy is measured in terms of frequency, which is an established custom within the field of quantum nanoscience.

Appendix B

Time evolution of a density operator

Whereas the Schrödinger equation describes the time evolution of pure states, the time evolution of a density operator is given by the Liouville-von Neumann equation:

$$i\hbar\frac{\partial\rho}{\partial t} = [H, \rho]. \tag{B.1}$$

In fact, the two equations are equivalent. Solving the equation above for a time-independent hamiltonian requires the use of the following two lemmas on the subject of matrix derivatives and commutators.

Lemma B.0.1. *If $X(t)$ and $Y(t)$ are two complex $n \times n$ matrices depending on $t \in \mathbb{R}$ and containing the elements $x_{ij}(t)$ and $y_{ij}(t)$ respectively, then*

$$\frac{\partial(XY)}{\partial t} = X\frac{\partial Y}{\partial t} + \frac{\partial X}{\partial t}Y.$$

Proof. It is sufficient to prove that the matrix elements on both sides of the equation are equal:

$$\begin{aligned}
\left(\frac{\partial(XY)}{\partial t}\right)_{ij} &= \frac{\partial}{\partial t}(XY)_{ij} \\
&= \frac{\partial}{\partial t}\left(\sum_k x_{ik}y_{kj}\right) \\
&= \sum_k \left(x_{ik}\frac{\partial y_{kj}}{\partial t} + \frac{\partial x_{ik}}{\partial t}y_{kj}\right) \\
&= \sum_k \left(x_{ik}\left(\frac{\partial Y}{\partial t}\right)_{kj}\right) + \sum_k \left(\left(\frac{\partial X}{\partial t}\right)_{ik} y_{kj}\right) \\
&= \left(X\frac{\partial Y}{\partial t}\right)_{ij} + \left(\frac{\partial X}{\partial t}Y\right)_{ij} \\
&= \left(X\frac{\partial Y}{\partial t} + \frac{\partial X}{\partial t}Y\right)_{ij}.
\end{aligned}$$

□

Lemma B.0.2. *If X is a complex $n \times n$ matrix and $a \in \mathbb{R}$, then*

$$[H, e^{aiH}] = 0.$$

Proof. Since H obviously commutes with itself: $He^{aiH} = H\sum_{n=0}^{\infty}\frac{(aiH)^n}{n!} = \sum_{n=0}^{\infty}\frac{(ai)^n HH^n}{n!} = \sum_{n=0}^{\infty}\frac{(ai)^n H^n H}{n!} = \sum_{n=0}^{\infty}\frac{(aiH)^n}{n!}H = e^{aiH}H.$ □

The following theorem now provides the solution of $\rho(t)$ to the Liouville-von Neumann equation.

Theorem B.0.3. *Considering a time-independent hamiltonian and given the initial density state ρ_0, the solution to the Liouville-von Neumann equation equals*[1]

$$\rho(t) = e^{-iHt/\hbar}\rho_0 e^{iHt/\hbar}.$$

Proof. Starting with the left-hand side of Equation B.1, we substitute the solution $\rho(t)$, applying Lemma B.0.1 twice and Lemma B.0.2 once:

$$\begin{aligned}
i\hbar\frac{\partial\rho}{\partial t} &= i\hbar\frac{\partial}{\partial t}\left(e^{-iHt/\hbar}\rho_0 e^{iHt/\hbar}\right) \\
&= i\hbar\left(e^{-iHt/\hbar}\frac{\partial}{\partial t}\left(\rho_0 e^{iHt/\hbar}\right) + \frac{\partial}{\partial t}\left(e^{-iHt/\hbar}\right)\rho_0 e^{iHt/\hbar}\right) \\
&= i\hbar\left(e^{-iHt/\hbar}\left(\rho_0\frac{\partial}{\partial t}e^{iHt/\hbar} + \frac{\partial\rho_0}{\partial t}e^{iHt/\hbar}\right) + \frac{\partial}{\partial t}\left(e^{-iHt/\hbar}\right)\rho_0 e^{iHt/\hbar}\right) \\
&= i\hbar\left(e^{-iHt/\hbar}\left(\frac{\rho_0 iH}{\hbar}e^{iHt/\hbar}\right) - \left(\frac{iH}{\hbar}e^{-iHt/\hbar}\right)\rho_0 e^{iHt/\hbar}\right) \\
&= -\left(e^{-iHt/\hbar}\rho_0 H e^{iHt/\hbar} - He^{-iHt/\hbar}\rho_0 e^{iHt/\hbar}\right) \\
&= -\left(e^{-iHt/\hbar}\rho_0 e^{iHt/\hbar}H - He^{-iHt/\hbar}\rho_0 e^{iHt/\hbar}\right) \\
&= H\rho - \rho H \\
&= [H,\rho].
\end{aligned}$$

□

In general, it is more convenient to write the solution in terms of the propagator $U = e^{-iHt/\hbar}$:

$$\rho(t) = U\rho_0 U^\dagger. \tag{B.2}$$

[1] For the sake of readability and numerical simulation, the reduced Planck constant $\hbar$ is usually set to 1. As a consequence, energy is measured in terms of frequency, which is an established custom within the field of quantum nanoscience.

The following lemma shows that propagators of hermitian matrices are unitary.

Lemma B.0.4. *If H is a hermitian $n \times n$ matrix, then its propagator $U = e^{-iHt/\hbar}$ is unitary.*

Proof. Using the properties of the hermitian adjoint on bounded operators $(X+Y)^\dagger = X^\dagger + Y^\dagger$ and $(XY)^\dagger = Y^\dagger X^\dagger$, we conclude that:

$$\begin{aligned}
U^\dagger U &= \left(e^{-iHt/\hbar}\right)^\dagger e^{-iHt/\hbar} \\
&= \left(\sum_{n=0}^{\infty} \frac{(-iHt/\hbar)^n}{n!}\right)^\dagger e^{-iHt/\hbar} \\
&= \sum_{n=0}^{\infty} \frac{((-iHt/\hbar)^n)^\dagger}{n!} e^{-iHt/\hbar} \\
&= \sum_{n=0}^{\infty} \frac{\left((-iHt/\hbar)^\dagger\right)^n}{n!} e^{-iHt/\hbar} \\
&= \sum_{n=0}^{\infty} \frac{\left(iH^\dagger t/\hbar\right)^n}{n!} e^{-iHt/\hbar} \\
&= \sum_{n=0}^{\infty} \frac{(iHt/\hbar)^n}{n!} e^{-iHt/\hbar} \\
&= e^{iHt/\hbar} e^{-iHt/\hbar} \\
&= I.
\end{aligned}$$

□

Appendix C

GRAPE algorithm Python code

```
import numpy as np
import pickle

# Functions
def build_UF():

    # Unconditional pi gate
    """
    UF = np.eye(9, 9)
    UF[3, :] = np.eye(9, 9)[6, :]
    UF[4, :] = np.eye(9, 9)[7, :]
    UF[5, :] = np.eye(9, 9)[8, :]
    UF[6, :] = np.eye(9, 9)[3, :]
    UF[7, :] = np.eye(9, 9)[4, :]
    UF[8, :] = np.eye(9, 9)[5, :]

    # CNOT gate
    """
    UF = np.eye(9, 9)
    UF[5, :] = np.eye(9, 9)[8, :]
    UF[8, :] = np.eye(9, 9)[5, :]


    return UF

def build_rho():

    p0_s = [0, 0, 0, 1, 0, 0, 0, 0, 0]
    pT_s = [0, 0, 0, 0, 0, 0, 1, 0, 0]

    p0   = np.array([p0_s[1], p0_s[0], p0_s[2], p0_s[7], p0_s[6],
        p0_s[8], p0_s[4], p0_s[3], p0_s[5]]) # probabilities (
        initial)
    pT   = np.array([pT_s[1], pT_s[0], pT_s[2], pT_s[7], pT_s[6],
        pT_s[8], pT_s[4], pT_s[3], pT_s[5]]) # probabilities (
```

```
        target)

    rho0 = np.zeros((9, 9))
    rhoT = np.zeros((9, 9))

    for i in range(0, 9):
        rho0 = np.matrix(rho0 + p0[i] * np.outer(np.eye(9)[:, i],
            np.eye(9)[:, i]))   # density matrix (initial)
        rhoT = np.matrix(rhoT + pT[i] * np.outer(np.eye(9)[:, i],
            np.eye(9)[:, i]))   # density matrix (target)

    return (rho0, rhoT)

# Constants of relevance
# ------------------------------------------------------
D       = 2 * np.pi * 2.878e9       # Zero field splitting (Hz)
Q       = 2 * np.pi * 4.946e6       # Quadrupole splitting (Hz)
gamma_e = 2 * np.pi * 2.802e6       # Gyromagnetic ratio e (Hz/G)
gamma_N = 2 * np.pi * 0.3e3         # Gyromagnetic ratio N (Hz/G)
AN      = 2 * np.pi * 2.186e6       # Coupling constant (Hz)

# Spin matrices
# ------------------------------------------------------
x = np.matrix([[0,   1, 0], [1,  0,   1], [0,  1,  0]]) * np.sqrt
    (2) / 2
y = np.matrix([[0, -1j, 0], [1j, 0, -1j], [0, 1j,  0]]) * np.sqrt
    (2) / 2
z = np.matrix([[1,   0, 0], [0,  0,   0], [0,  0, -1]])

# Simulation constants
# ------------------------------------------------------
J    = 1        # Number of iterations
K    = 1000     # Number of magnetic fields
L    = 1        # Number of processes
N    = 10       # Number of data points
M    = 5000     # Number of iterations
eps  = 1e15     # Stepsize
tmin = 0        # Minimum time
tmax = 20e-9    # Maximum time

# Building initial and target state
# ------------------------------------------------------
(rho0, rhoT) = build_rho()
UF           = build_UF()

# Set time scale
# ------------------------------------------------------
tlist = np.linspace(tmin, tmax, N + 1)

# Magnetic field initialization
# ------------------------------------------------------
Bz = 100e6 / gamma_e
Bz_list = np.random.normal(Bz, 0.005 * Bz, K)
```

```
import scipy.linalg
import numpy as np
import multiprocessing as mp
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from scipy.optimize import minimize
from IPython.parallel import Client
from pylab import *
from settings import *
import pickle
import pprint
import time
import sys
import dill

plt.close('all')

# Functions
# ------------------------------------------------------------
def build_H(Bz):

    w_e     = D - gamma_e * Bz      # Driving frequency (electron) (
        Hz)
    w_N     = Q                     # Driving frequency (nitrogen) (
        Hz)

    H0 = D * kron(z ** 2, eye(3)) + gamma_e * Bz * kron(z, eye(3))
        - \
         Q * kron(eye(3), z ** 2) + gamma_N * Bz * kron(eye(3), z)
             - \
         AN * kron(z, z) - \
         w_e * kron(z ** 2, eye(3)) + \
         w_N * kron(eye(3), z ** 2)

    H    =  2 * [0]
    H[0] = sqrt(2) / 2 * kron(x, eye(3))  # First control
        hamiltonian
    H[1] = sqrt(2) / 2 * kron(eye(3), x)  # Second control
        hamiltonian

    return (H0, H)

def plot_rho(tlist, rho, color):

    RHO = array(rho)
    N   = shape(rho)[0] - 1

    """
    for i in range(0, 9):
        plt.subplot(9, 1, i + 1)
        if i == 0:
            plt.title('Eigenstate probabilities')
        if i == 8:
            plt.xlabel('Time (s)')
        plot(tlist, RHO[0 : N + 1, i, i], color=color)
        plt.axis([tlist[0], tlist[N], 0, 1])
```

```
    """

    f, axarr = plt.subplots(3, 1)

    plt.xlabel(r'Time (s)')
    axarr[1].set_ylabel(r'Eigenstate probability')

    plt.rc('font', **{'family': 'serif', 'serif': ['Computer Modern
        ']})
    plt.rc('text', usetex=True)

    axarr[0].plot(tlist, RHO[0 : N + 1, 1, 1], color=color)
    axarr[0].set_title(r'$m_s = 1$, $m_N = 0$')
    axarr[1].plot(tlist, RHO[0 : N + 1, 7, 7], color=color)
    axarr[1].set_title(r'$m_s = -1$, $m_N = 0$')
    axarr[2].plot(tlist, RHO[0 : N + 1, 4, 4], color=color)
    axarr[2].set_title(r'$m_s = 0$, $m_N = 0$')

    # Fine-tune figure
    plt.setp([a.get_xticklabels() for a in f.axes[:-1]], visible=
        False)

    for i in range(3):
            axarr[i].axis([tlist[0], tlist[N], 0, 1])

def expdiag(A):

    if shape(A)[0] != shape(A)[1]:
        print 'Error in expdiag: input should be an nxn matrix'
        return
    else:
        res = (1 + 0j) * eye(shape(A)[0])
        for i in range(0, shape(A)[0]):
            res[i, i] = exp(A[i, i])
    return res

def expm(A, dt):

    A = matrix(A)
    if shape(A)[0] != shape(A)[1]:
        print 'Error in expm: input should be an nxn matrix'
        return
    elif any(A.H != A):
        print 'Error in expm: matrix is not hermitian'
    else:
        L = eye(shape(A)[0]) * (-1j * dt) * eigh(A)[0]
        U = eigh(A)[1]
        res = dot(U, dot(expdiag(L), inv(U)))
    return res

def gate(tmin, tmax, N, H0, H, u, n):

    dt      = (tmax - tmin) / N
    U       =  N * [0]
    gate    = eye(9, 9)

    for j in range(0, N): # Calculation of propagators
```

```
                SUM  = 0
                for i in range(0, shape(u)[0]):
                    SUM = SUM + u[i, j, n] * H[i]
                U[j] = expm(H0 + SUM, dt)

    for j in range(0, N):
        gate = dot(U[j], gate)

    return gate

def build_prop(N, H, H0, dt, u, n, j):

    U   = eye(9, 9)
    SUM = 0

    for i in range(0, shape(u)[0]):
        SUM = SUM + u[i, j, n] * H[i]
    U = expm(H0 + SUM, dt)

    return U

def fidelity_G(x, y):

    X = matrix(x)
    Y = matrix(y)

    return abs(trace(dot(X.H, Y)) / trace(dot(Y.H, Y)))

def fidelity_S(x, y):

    # x represents a pure state, y represents an arbitrary state

    X   = matrix(x)
    phi = array([ X[i, i] for i in range(len(X)) ])
    Y   = matrix(y)

    return abs(sum(dot(conjugate(phi), dot(Y, phi).T)))

def gaussian(x, mu, sig):

    return exp(-1 / 2 * ((x - mu) / sig) ** 2) / (sig * sqrt(2 * pi
        ))

def single_evolution(Bz):

    f = open('U_CALC.p', "r")
    u_calc = pickle.load(f)

    (H0, H) = build_H(Bz)

    return evolution(tmin, tmax, N, rho0, rhoT, H0, H, u_calc, M)
```

```
from dGRAPE import *

rc = Client()
dview = rc[:]
dview.use_dill()

with dview.sync_imports():
        import scipy.linalg
        import numpy as np
        from scipy.sparse import csr_matrix
        from scipy.optimize import minimize
        from dGRAPE import build_prop

#-----------------------------------------------------------
def evolution(tmin, tmax, N, rho0, rhoT, H0, H, u, n):

        def single_build_prop_par(j):

                return build_prop(N, H, H0, dt, u, n, j)

        dt      = (tmax - tmin) / N
        rho     = (N + 1) * [0]
        lam     = (N + 1) * [0]
        rho[0] = rho0
        lam[N] = rhoT

        dview.push(dict(N=N, H=H, H0=H0, dt=dt, u=u, n=n))

        U = dview.map_sync(single_build_prop_par, range(0, N))

        for j in range(0, N): # Calculation of rho and lambda for
            each j
                rho[j + 1]      = dot(U[j], dot(rho[j], U[j].H))
                lam[N - j - 1] = dot(U[N - j - 1].H, dot(lam[N - j
                    ], U[N - j - 1]))

        return (rho, lam)

def build_U(tmin, tmax, N, H0, H, u, n):

        def single_build_prop_par(j):

                return build_prop(N, H, H0, dt, u, n, j)

        dt = (tmax - tmin) / N

        dview.push(dict(N=N, H=H, H0=H0, dt=dt, u=u, n=n))

        U = dview.map_sync(single_build_prop_par, range(0, N))

        return U

def grape_S(tmin, tmax, M, N, rho0, rhoT, u, eps, BZ):

        dt      = (tmax - tmin) / N
        tlist   = linspace(tmin, tmax, N + 1)
        F       = zeros(M)
```

```
for n in range(0, M):

        gradient = zeros((shape(u)[0], N))

        for b in range(0, size(BZ)):

                (H0, H) = build_H(BZ[b])
                (rho, lam) = evolution(tmin, tmax, N, rho0,
                     rhoT, H0, H, u, n)

                for i in range(0, shape(u)[0]):
                        for j in range(0, N):
                                gradient[i, j] = gradient[i
                                    , j] + trace( -1j * dt
                                    * dot(lam[j].H, dot(H[i
                                    ], rho[j]) - dot(rho[j
                                    ], H[i])))

        F[n] = fidelity_S(rhoT, rho[-1])
        print ' State fidelity: ' + str(F[n])

        for i in range(0, shape(u)[0]):
                for j in range(0, N):
                        u[i, j, n + 1] = u[i, j, n] + eps *
                             gradient[i, j] / size(BZ)

        ETA = 'GRAPE-algorithm in progress: ' + str(n * 100
             / M) + '%'
        sys.stdout.write('\r'+ '                          
            ------------------------------------------')
        sys.stdout.write('\r'+ ETA)

sys.stdout.write('\r'+ '                                  
    ----------------------------------')
sys.stdout.write('\r'+ 'Calculation complete: 100%')

"""
plot_rho(tlist, rho, 'r')
plt.rc('font', **{'family': 'serif', 'serif': ['Computer
    Modern']})
plt.rc('text', usetex=True)

figure(1)
colormap = plt.cm.jet
plt.gca().set_color_cycle([colormap(i) for i in linspace(0,
     0.9, M)])
plot(tlist[0 : -1], u[0, :, :])

figure(2)
colormap = plt.cm.jet
plt.gca().set_color_cycle([colormap(i) for i in np.linspace
    (0, 0.9, M)])
plot(tlist[0 : -1], u[1, :, :])

show()
```

```
        """
        maximum = max(F)
        index = [i for i, j in enumerate(F) if j == maximum][0]

        return u, index, F

def grape_G(tmin, tmax, M, N, UF, u, eps, Bz):

        (H0, H) = build_H(Bz)
        dt      = (tmax - tmin) / N
        tlist   = linspace(tmin, tmax, N)
        X       = (N + 1) * [0]
        P       = (N + 1) * [0]
        X[0]    = matrix(eye(9, 9))
        P[N]    = matrix(UF)
        F       = zeros(M)

        for n in range(0, M):

                U = build_U(tmin, tmax, N, H0, H, u, n)

                for j in range(0, N): # Calculation of X and P for
                    each j
                        X[j + 1]     = dot(U[j], X[j])
                        P[N - j - 1] = dot(U[N - j - 1].H, P[N - j
                            ])

                for i in range(0, shape(u)[0]):
                        for j in range(0, N):
                                u[i, j, n + 1] = u[i, j, n] + eps *
                                    -2 * real(trace(1j * dt * dot(
                                    P[j].H, dot(H[i], X[j])))) *
                                    trace(dot(X[j].H, P[j])))

                ETA = 'GRAPE-algorithm in progress: ' + str(n * 100
                    / M) + '%'
                sys.stdout.write('\r'+ '                            
                    ')
                sys.stdout.write('\r'+ ETA)

                F[n] = fidelity_G(gate(tmin, tmax, N, H0, H, u, n),
                    UF)
                print ' Gate fidelity: ' + str(F[n])

        sys.stdout.write('\r'+ '                                    
            ')
        sys.stdout.write('\r'+ 'Calculation complete: 100%')

        GATE = array(X)

        """
        plot_rho(tlist, rho, 'r')
        plt.rc('font', **{'family': 'serif', 'serif': ['Computer
            Modern']})
        plt.rc('text', usetex=True)

    figure(1)
```

```
    colormap = plt.cm.jet
    plt.gca().set_color_cycle([colormap(i) for i in linspace(0,
        0.9, M)])
    plot(tlist, u[0, :, :])

    figure(2)
    colormap = plt.cm.jet
    plt.gca().set_color_cycle([colormap(i) for i in np.linspace(0,
        0.9, M)])
    plot(tlist, u[1, :])

    show()

        """
        maximum = max(F)
        index = [i for i, j in enumerate(F) if j == maximum][0]

        return u, index, F
# ------------------------------------------------------------

# Initial controls
# ------------------------------------------------------------
u = np.zeros((2, N, M + 1))

# driving amplitude constant (Hz) (omega_e)
u[0, :, 0] = 2 * np.pi * 10e6 + 0 * np.linspace(0, tmax, N)

# driving amplitude constant (Hz) (omega_N) unew1[::10, -1]
u[1, :, 0] = 0 * np.pi * 10e3 + 0 * np.linspace(0, tmax, N)


# Testing procedures
# ------------------------------------------------------------

# TESTING DECOHERENCE OPTIMIZED GRAPE
(H0, H) = build_H(Bz)

BZ = np.random.normal(Bz, 0.005 * Bz, 200)
BZ = [Bz]

(u, index, F) = grape_S(tmin, tmax, M, N, rho0, rhoT, u, eps, BZ)

(rho, lam) = evolution(tmin, tmax, N, rho0, rhoT, H0, H, u, index)

# TESTING SYNTHESIS OF UNITATRY TRANSFORMATIONS
(u, index, F) = grape_G(tmin, tmax, M, N, UF, u, eps, Bz)

print ' State fidelity: ' + str(F[index]) + ' Index: ' + str(index)

plot(range(M), F)
show()
```