



# Gate Set Tomography for Nitrogen-Vacancy Systems

David M. Vos

Supervisor: Johannes Borregaard

EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,  
In Partial Fulfilment of the Requirements  
For the Bachelor of Computer Science and Engineering

### Abstract

Quantum computing is an emerging field with many promising future applications. These include, but are not limited to, quantum machine learning, quantum cryptography and quantum chemical engineering.

Before these can be realised, obstacles, which arise due to scaling, need to be overcome. To accomplish this, quantum processors must be built with low rates of noise and high rates of target gate fidelity. By characterising quantum systems, possible sources of noise can be identified, and consequently, systems can be designed which effectively suppress noise.

Gate Set Tomography (GST) is a protocol developed for the characterisation of quantum processors. In this research, we apply GST to nitrogen-vacancy (NV) centre systems. We also construct a widget meant to visualise GST results in an intuitive manner.

Our research is based on twelve models, varying in the number of gates used, the initial state of the nitrogen nucleus, and whether an XY4 echo was applied after gates. We use simulated and experimental models, and analyse these using the error generator, diamond norm and  $N_\sigma$  metrics.

We conclude that our simulated models capture sources of noise, as the proportion of stochastic errors shifts to Hamiltonian errors, when we change our target model from the ideal model to simulated ones. We also conclude that the XY4 echo significantly reduces non-Markovianity, which arises due to coupling. Furthermore, evidence which points to calibration faults is discovered within certain models. Finally, we present the visualisation widget and show how it can be used to interpret results.

**Keywords:** quantum computing, gate set tomography, NV systems, system characterisation

# Contents

<b>1</b>	<b>Introducton</b>	<b>1</b>
1.1	Related Work . . . . .	1
1.2	Motivation and Research Objectives . . . . .	1
1.3	Research Question . . . . .	2
1.4	Summary Of Contents . . . . .	2
<b>2</b>	<b>Theoretical Background</b>	<b>2</b>
2.1	Definitions . . . . .	3
2.2	Gate Set Tomography . . . . .	4
2.2.1	Fiducials . . . . .	4
2.2.2	Germes . . . . .	5
2.2.3	Algorithm . . . . .	6
2.2.4	Parameterisation . . . . .	6
2.2.5	Gauge Optimisation . . . . .	6
2.2.6	Non-Markovianity . . . . .	7
2.2.7	Error Generator . . . . .	7
2.2.8	Diamond Norm . . . . .	8
2.3	Nitrogen-Vacancy Quantum Bits . . . . .	8
<b>3</b>	<b>Python GST Implementation</b>	<b>9</b>
3.1	Basic objects . . . . .	9
3.2	Algorithms . . . . .	11
3.3	Protocol . . . . .	11
<b>4</b>	<b>Methodology</b>	<b>12</b>
4.1	Model and Experiment Design Construction . . . . .	12
4.2	Experimental Data . . . . .	12
4.3	Simulated Data . . . . .	13
<b>5</b>	<b>Results and Discussion</b>	<b>13</b>
5.1	Error Generator Metric . . . . .	13
5.1.1	Trends and Outliers . . . . .	14
5.1.2	Effects of Nitrogen Initialisation . . . . .	14
5.2	Diamond Norm . . . . .	15
5.3	Non-Markovianity . . . . .	15
5.3.1	CPTP vs TP . . . . .	15
5.3.2	Effects of XY4 Echo . . . . .	16
5.4	GSTBlochWidget . . . . .	16
5.4.1	Development of GSTBlochWidget . . . . .	16
5.4.2	Application of GSTBlochWidget . . . . .	17
<b>6</b>	<b>Responsible Research</b>	<b>17</b>
<b>7</b>	<b>Conclusion</b>	<b>18</b>
7.1	Summary . . . . .	18
7.2	Future Work . . . . .	18
7.3	Acknowledgements . . . . .	19
<b>A</b>	<b>Quantum Computer Science</b>	<b>19</b>
A.1	The Quantum Bit . . . . .	19
A.2	The Quantum Gate . . . . .	21
A.3	The Quantum Circuit . . . . .	21
<b>B</b>	<b>GSTBlochWidget Additional Data</b>	<b>22</b>
B.1	UML Diagram . . . . .	22
B.2	Example Code . . . . .	22
<b>C</b>	<b>Tables of Results</b>	<b>23</b>
C.1	Error Generators and Diamond Norm . . . . .	23
C.2	$N_\sigma$ metric . . . . .	24
	<b>References</b>	<b>25</b>

# 1 Introduction

## 1.1 Related Work

**Quantum computer science** is a new and emerging field combining quantum mechanics with computer science. Quantum computer science was first proposed in 1980 by physicist Paul Benioff who envisioned a quantum mechanical version of a Turing machine [1]. Richard Feynman and Yuri Manin later proposed that a quantum computer could execute algorithms which were non-feasible for classical computers [2][3]. In 1986 Richard Feynman introduced an early version of the quantum circuit notation used to this day [4], turning abstract ideas into an exact science. Shor's algorithm [5] was developed in 1994, which theoretically has the power to break RSA encryption. After this, the first physical implementation of a quantum algorithm on a two-qubit system was implemented in 1998 [6].

In the years since then, landmark achievements have been accomplished. In 2016 Chinese researchers entangled particles 1200 km away using a satellite [7]. In 2019 Google claimed to have reached quantum supremacy [8], meaning they outperformed a classical computer with a quantum computer. In 2021 IBM launched a 127 qubit quantum processor, a doubling in size compared to their earlier model launched in 2020 [9], which begs the question if Moore's law applies to both classical and quantum computers. Today quantum computing is a fast-paced and quickly developing research field and industry. An introduction to the concepts behind the field of quantum computer science can be found in appendix A.

**The motivations** for enabling quantum computing are manifold. Quantum computers carry the possibility to revolutionise computation, by making some problems which are intractable on classical computers, solvable. For example, quantum machine learning aims to optimise certain AI problems by outsourcing computationally difficult subroutines to quantum computers [10], while quantum finance seeks to solve finance problems by running intractable simulations on quantum computers [11]. The research fields of chemistry and biology can also be revolutionised by quantum computers. These fields seek to solve problems regarding the simulation of molecular quantum systems, such as proteins [12]. Quantum computers present themselves as the perfect machines on which to run these simulations [13]. Lastly, but perhaps most importantly, the field of cybersecurity has an immense interest in quantum computer science. As mentioned, quantum computers theoretically have the power to break RSA encryption, and informationally secure channels can be constructed, which use the laws of quantum mechanics to make it noticeable when a channel has been compromised [14].

## 1.2 Motivation and Research Objectives

**Before all these amazing advancements** can be made a reality, quantum computers of a certain size and robustness need to be built. Running complicated quantum algorithms requires a large number of qubits. For example, the 127 qubit processor available at this time is a far cry from the 10,000 qubits that have been predicted to be required for breaking 2048-bit RSA encryption [15]. Furthermore, quantum systems tend to increase in noise and complexity when scaled, pitting this fast-growing field against enormous obstacles to overcome.

**Nitrogen-Vacancy (NV) centre systems** provide one possible way to implement quantum computers. NV systems, due to the nature of their structure, exhibit properties which are extremely sought after in the quest for scalable quantum computation. In 2021 Taminiau et al showed that it was possible to implement fault-tolerant operations on an NV centre system. However, they also state: "future improvements in fidelity and the number of qubits will be required" [16]. Characterisation of NV centre systems could greatly increase the target gate fidelity of these systems. Understanding where noise appears to exhibit itself can lead to discovering what possible sources of noise there might be, and consequently to designing systems with reduced noise and increased target gate fidelity.

**Gate Set Tomography (GST)** is a framework which was developed to characterise quantum systems. The framework incorporates the principle that our assumptions regarding what can be known of a system are incomplete. GST is calibration-free, meaning that it does not take into account perfect preparations or measurements of a system, and only relies on the data presented to it.

**The motivations** as to why one should use NV systems as a target system when performing GST are as follows. Firstly, the fact that this system utilises a point defect in a crystal lattice structure translates to extremely high coherence times, making it a preferable candidate for certain applications, and by extension optimisation. This means that the quantum information stored within NV systems has a lifespan which is larger than comparable quantum information systems. Secondly, and more importantly,

the free electron in the vacancy is surrounded by carbon nuclei, every one of these nuclei has its spin and magnetic field. The nitrogen nucleus couples to the electron, and if not initialised properly it may produce a lot of non-Markovian noise. As a result, our system has many possible sources of noise. Therefore, NV systems are of particular interest to a protocol like GST, which has been designed to characterise and quantify noise.

**In this research** we apply GST to NV centre systems using the open-source framework pyGSTi. Note that we will focus on one qubit systems in this paper. Several models and data sets are analysed, and discernible discrepancies will be discussed in order to characterise NV centre systems. The results from this research can then be used for further research to design improved NV centre quantum computers. Furthermore, we will construct a widget to visualise the results obtained by applying GST. This widget serves to improve upon the interpretation of the observed errors and can be used by research groups to build an intuition for characterised systems.

### 1.3 Research Question

**As established** the main aim of this project will be to apply GST, by use of pyGSTi, to data produced by a simulated and experimental NV system. Therefore the main research question of this project is:

- **How can pyGSTi be employed to characterise operations in an NV-based quantum device when comparing simulated data with experimental data?**

**For this project** both experimental data from a physical NV system and simulated data could be accessed. The data sets could be generated with the nitrogen nucleus in differing initialisation states, data sets could also be produced employing the XY4 echo mentioned in 2.4. Furthermore, it may be hard to interpret GST results using standard metrics.

**Our sub-questions** are defined as follows:

- What does pyGSTi tell us when we compare experimental models with simulated models with differing nitrogen initialisation states?
- What does pyGSTi tell us when we analyse models which employ an XY4 echo?
- How can GST results be visualised in a way that is intuitive and simple to understand?

### 1.4 Summary Of Contents

**A theoretical background** required for interpreting future results will be given in section 2. Various concepts are defined, the inner workings of GST are explained and information is presented regarding NV systems. Section 3 introduces pyGSTi, and explains the different objects, algorithms and protocols used within this framework. The methodology applied in this project is described in section 4, the workflow and constructed models are presented, and it is clarified how experimental and simulated data were collected in order to run GST. Section 5 presents the results and discussion of this project. It uses the error generator, diamond norm and  $N_\sigma$  metric to accomplish this. Comparisons between models are drawn, and our results are interpreted in order to characterise the system. It also details how the visualisation widget is implemented, and the code which underlies it. We also examine how the visualisation widget may be used to interpret results. Section 6 considers the reproducibility of this research and explains why responsible research is important. Finally, in section 7 we summarise our research and present questions which may be researched in the future. We then give credit where credit is due in the acknowledgements subsection. Furthermore, appendix A offers introductory information regarding quantum computer science. Appendix B presents the UML diagram and a code example of the visualisation widget. Finally, appendix C summarises the metrics collected by GST runs in the form of tables.

## 2 Theoretical Background

In the following subsections, both GST and NV centre qubits will be briefly explained. However, to properly understand the gist of GST, some background knowledge regarding quantum computer science is needed. If the reader has no former experience with quantum computer science, we refer the reader to appendix A, which offers an introductory section by explaining the basics of quantum computer science.

In this section, we will define the density matrix, superoperator, Bloch sphere, measurement and a gate set, after which we move on to explain GST. Finally, we will conclude by briefly explaining what an NV centre is, and how it can be controlled and measured.

## 2.1 Definitions

**The density matrix** is a matrix that describes a quantum state, it can describe a "non-pure" quantum state known as a mixed state. Mixed states generally arise in three different situations. Firstly when the preparation of the system is not fully known. Secondly, when one wants to describe an entangled system, which will not be the case for this project as one needs more than a single qubit for entanglement. Finally, when noise or decoherence transforms a pure state into a mixed state, by "leaking" out the quantum information of the system.

The density operator is mathematically defined as:

$$\rho = \sum_j p_j |\psi_j\rangle \langle \psi_j| \quad (1)$$

Where  $p_j$  is the probability of quantum state  $|\psi_j\rangle$ . The density matrix must conform to three requirements: it must be Hermitian, have a trace of 1 and be positive semi-definite.

A density matrix can also be written in the superket notation, which is defined as:  $|\rho\rangle\rangle = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$

where  $c_i = \text{tr}(\sigma_i \cdot \rho)$  and  $\sigma_i$  is one of the Pauli matrices or the Identity matrix.

**The superoperator** is a linear operator which acts upon a superket to transform the quantum state, that is:  $|\rho'\rangle\rangle = G|\rho\rangle\rangle$ . To be considered a physically valid superoperator it should follow two constraints. Firstly, it should preserve the trace of the density matrix. The trace of the density matrix encodes the measurement probabilities, and  $\text{tr}(\rho) \neq 1$  is considered nonphysical. The second constraint is that the density matrix should be completely positive, as negative probabilities make no sense physically. Together we call these constraints completely positive and trace-preserving (CPTP).

**The Bloch sphere** is an intuitive way to visualise a quantum state. The Bloch sphere is defined as a three-dimensional sphere, of which one dimension is complex. A quantum state is defined by the Bloch vector, if the Bloch vector has a length of 1 it resides on the surface of the Bloch sphere, and describes a pure state. Consequently, mixed states are described by a Bloch vector with a length smaller than 1. In the context of superkets we can find a corresponding Bloch vector by using the formula  $\rho = \frac{1}{2}(I + r_x\sigma_x + r_y\sigma_y + r_z\sigma_z)$ . Where  $r_x$ ,  $r_y$  and  $r_z$  are the coordinates of the Bloch vector. The superket vector can easily be rewritten to the notation above by setting  $c_0 = 1$ , in this case  $c_1 = r_x$ ,  $c_2 = r_y$  and  $c_3 = r_z$ . The inverse operation can be accomplished using:  $\rho' = \sum_{i=0}^3 c_i' \cdot \sigma_i$ .

**Measurement** is an operation applied to a quantum state that "collapses" the wave function and forces it into a classical state. Given an input state  $|\rho\rangle\rangle$  the probability of measuring an effect  $i$  is:  $\text{prob}(i|\rho) = \langle\langle E_i|\rho\rangle\rangle$ . An effect is simply a possible outcome for a quantum system, for example, if we measure an NV centre system, an effect would be to measure the  $|1\rangle$  state by observing a photon. The set of effects  $E_i$  completely describes all possible measurements and is also known as a positive operator-valued measure (POVM).

**The Gate Set**  $\mathcal{G}$  will now be defined, that is the mathematical definition of the set that GST aims to characterise.

Given  $G_i : \mathcal{B}(\mathcal{H}) \rightarrow \mathcal{B}(\mathcal{H})$  for  $i = 1, \dots, N_G$ , where  $G_i$  is a quantum gate and  $N_G$  is the total number of distinct quantum gates.

$|\rho^{(i)}\rangle\rangle$  for  $i = 1, \dots, N_\rho$ , where  $N_\rho$  is the total number of distinct state preparations.

$\langle\langle E_i^{(m)}|$  for  $m = 1, \dots, N_M$  and  $i = 1, \dots, N_E^{(m)}$ , where  $N_M$  is the total number of distinct measurements, and the  $m$ -th measurement has  $N_E^{(m)}$  distinct outcomes.

A gate set is defined by:

$$\mathcal{G} = \left\{ \{|\rho^{(i)}\rangle\rangle\}_{i=1}^{N_\rho}; \{G_i\}_{i=1}^{N_G}; \{\langle\langle E_i^{(m)}|\}_{m=1, i=1}^{N_M, N_E^{(m)}} \right\} \quad (2)$$

This definition encompasses all the state preparations, gates and measurements on a quantum processor, which provides a complete description of our operations on it.

## 2.2 Gate Set Tomography

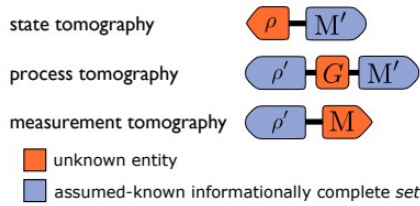


Figure 1: Tomography protocols preceding GST. State tomography seeks to characterise a quantum state  $\rho$ . Process tomography seeks to characterise a set of gates  $G$ . Measurement tomography seeks to characterise a set of measurement operations  $M$ . Figure adapted from [17].

**GST** is a protocol for detailed, predictive characterisation of gate sets on quantum computing processors. Early versions of GST emerged around 2012-13, and since then it has been refined, demonstrated, and used in a large number of experiments.

When a quantum gate is implemented in physical reality, the superoperator of said implementation will never exactly mirror the gate matrix. A fidelity of exactly 1 is next to impossible, due to a plethora of reasons including background noise, pulse spillover and gate-induced heating.

**Preceding tomography protocols** suffered from the fact that some point of the process was assumed to be completely known (pre-calibrated). An overview of earlier tomography protocols can be found in figure 1. State tomography, while trying to characterise a quantum state  $\rho$ , assumed there was an informationally complete set of fiducial measurements  $M'$  [18]. Fiducial in this context means "accepted as a fixed basis of reference" [17] and we will see it reappear in GST. On the

other hand, process tomography tries to characterise a gate set  $G$  while assuming the quantum state  $\rho'$  and the measurement set  $M'$  are completely known [19]. Lastly, measurement tomography seeks to characterise  $M$  while assuming  $\rho'$  is known [20].

GST aims to alleviate the practical problems introduced by these discrepancies. By mathematically analysing classical data produced by a substantial amount of different quantum circuits, a gate set is found which has the highest likelihood to describe the real-life superoperators. Furthermore, GST also provides us with a plethora of metrics to further analyse, in order to characterise the system.

**The motivation for GST** is two-fold:

1. An ability to accurately assess the capabilities of quantum processors.
2. An ability to characterise and understand noise, so quantum processors can be designed and built, which effectively suppress noise.

We will now further elaborate on the inner workings of GST. Note that to make this paper understandable for computer scientists mathematical details will largely be omitted. For further readings, we refer to the original 2021 GST paper [17].

The first step of GST is to calculate what data is needed for the characterisation. To this end, we create an "experiment design", which in its most bare form is simply a list of quantum circuits to be run on the target system. These circuits are composed of "fiducials" and "germs", which will now be explained in more detail.

### 2.2.1 Fiducials

**The first step in running the GST protocol** is to select fiducials. Fiducials are an informationally complete (IC) set of state preparation and measurement (SPAM) operators. These are the gates that will be put at the beginning and the end of every circuit needed for GST to calculate the estimated superoperators. Informationally complete in this context means that the possible measurements must span the complete set of effects, that is: it must be possible to measure at every possible quantum state of the quantum system in question. This is needed because generally, a quantum processor has one native state preparation (usually  $|0\rangle$ ) and one measurement basis (usually  $|0\rangle, |1\rangle$ ). Therefore, to get to the complete gate set from these initial states and measurements, a set of single-qubit rotations is needed.

**To give an example:** We are given a simple model of a quantum processor which can perform the X gate, Y gate and the I gate. In this model, it is in theory only possible to visit the  $|0\rangle$  and  $|1\rangle$  state.

The X gate and Y gate, describe a rotation of  $\pi$  radians around the x and y-axis of the Bloch sphere, respectively. See appendix A for a detailed explanation.

An example of a viable set of preparation fiducials would then be:  $(((), (X), (Y), (XX))$  An example of a viable set of measurement fiducials would then be:  $(((), (X), (Y), (XX))$  Note that the  $()$  is the Null gate, and differs from the identity gate in the sense that it does not have a duration associated with it.

### 2.2.2 Germs

**The second step in running the GST protocol** is to select a set of germs. These are the gates which will be sandwiched between the fiducials in an experiment. The constraint we have for a germ set is that it must be amplificationally complete (AC). This is defined as the set of gates that amplify every error that can be amplified [17]. This definition leaves a lot of room open for possible germ sets, as there exists an infinity of possible sets which adhere to this condition. A considerable part of pyGSTi is to find the sub-optimal germ set.

In an experiment, these germs are repeated  $p$  times to amplify these errors. For example, if one element of our germ set would be the  $X$  gate, which might e.g in physical reality over-rotate our qubit by  $\frac{1}{32}\pi$ , then this error is so small that it is generally hard to detect. If our  $p$ -value then is 16, we would apply the  $X$  gate 16 times resulting in a displacement of  $\frac{1}{2}\pi$ , which is an error magnitude which we confidently can identify.

It was found that logarithmically spaced values of  $p$  are optimal as this problem can be reduced to a binary search algorithm [17]. For example, for a circuit depth of 64, our  $p$  values would be [1, 2, 4, 16, 32, 64]. Choosing a larger circuit depth should translate to estimated superoperators with a better fit (as the errors are amplified more). The downside of a large circuit depth is that running the experiment takes more time and that after a certain amount of time the quantum information will be lost to decoherence.

**A point of notice** is that GST also offers the choice to apply fiducial pair reduction (FPR), as some of the circuits constructed for the experiment design are redundant.

As the authors of the original GST paper write: "The redundancy stems from the fact that (as observed previously) each germ  $g$  only amplifies certain "directions" in error space" and: "So if a germ  $g$  amplifies  $m$  directions in parameter space, it is sufficient to select  $m$  fiducial pairs that measure linearly independent probabilities sensitive to those  $m$  variations" [17].

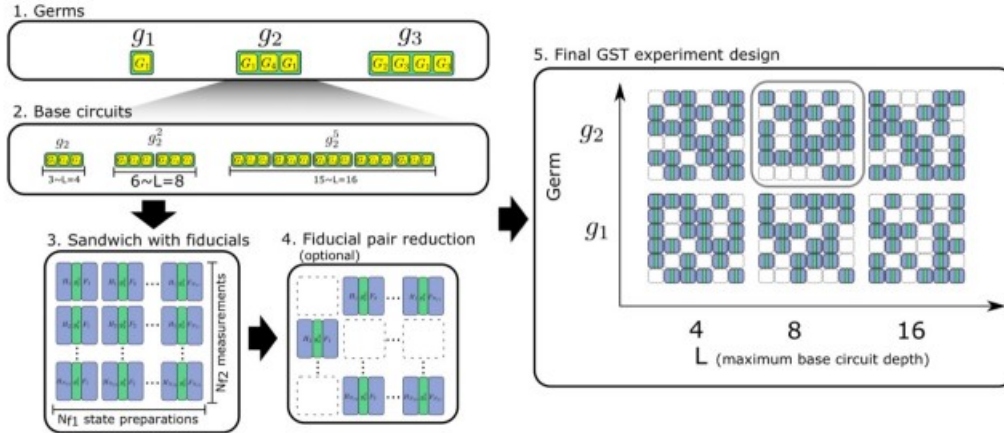


Figure 2: Overview of experiment design construction. 1) A set of AC germs is found. 2) base circuits are composed by repeating germs  $p$  times. 3) The resulting circuits are sandwiched between fiducials. 4) FPR may optionally be applied. 5) Our final result is a circuit list, varying in fiducials, germs and germ power  $p$ . Figure adapted from [17]

**Calculating an experiment design** becomes straightforward now that we know what fiducials and germs are. It is simply every possible combination of a preparation fiducial + germ <sup>$p$</sup>  + measurement fiducial, where  $p$  is the power of the germ. See figure 2 for a simple overview of a GST experiment design construction, which is as follows.

First, a set of AC germs  $g_i$  is constructed, then we choose a set of germ powers  $p_j$  and create our base circuits by repeating every  $g_i$  by  $p_j$  times. We then sandwich our base circuits with our previously selected fiducials and, optionally apply FPR to reduce the number of circuits within our experiment design. Our final result is a circuit list with a variation in fiducials, germs and circuit depth (how often the germ is repeated). We then run this circuit list on our physical quantum processor and feed the data back into the GST algorithm.



### 2.2.3 Algorithm

**After collecting the required data** it is time to run the GST protocol itself. A gate set can be regarded as a parameterised statistical model, therefore fitting a gate set to collected data is a form of parameter estimation. There are many ways to achieve parameter estimation, but GST generally uses maximum likelihood estimation (MLE). MLE varies the parameters of the gate set, to maximise the probability that the estimated superoperator fits the data.

The loglikelihood function is constructed as follows:

$$\log \mathcal{L}_s = N_s \sum_{\beta_s} f_{s,\beta_s} \log(p_{s,\beta_s}) \quad (3)$$

Where  $s$  is the index of a circuit in the experiment design,  $N_s$  is the amount of times  $s$  was repeated,  $m_s$  is the number of outcomes of  $s$ ,  $N_{s,\beta_s}$  is the the number of times outcome  $\beta_s$  was observed,  $p_{s,\beta_s}$  is the probability predicted by  $\mathcal{G}$  of getting outcome  $\beta_s$  from circuit  $s$  and  $f_{s,\beta_s}$  is the corresponding observed frequency:  $N_{s,\beta_s}/N_s$ .

Note that this formula describes the loglikelihood of a single circuit, to get the loglikelihood for the total GST experiment we simply sum them:  $\log \mathcal{L} = \sum_s \log \mathcal{L}_s$

The high-level algorithm used by the GST protocol to maximise the loglikelihood is seen in figure 3.

---

#### Algorithm 1 Gate Set Tomography

---

```

 $\vec{\theta} \leftarrow \vec{\theta}_0$ 
for  $L \in 1, 2, 4, 8, \dots$  do
     $\mathcal{D} \leftarrow \text{Truncate}(\mathcal{D}_0, L)$ 
     $\vec{\theta} \leftarrow \text{Argmin}(\chi^2, \mathcal{G}, \mathcal{D}, \vec{\theta})$ 
end for
 $\vec{\theta} \leftarrow \text{Argmin}(-\log \mathcal{L}, \mathcal{G}, \mathcal{D}_0, \vec{\theta})$ 

```

---

$\mathcal{D}_0$  is the full data set and  $\vec{\theta}_0$  is the initial vector of the model parameters.  $\vec{\theta}$  is the vector of the parameters of  $\mathcal{G}$ , which we are optimising.  $\text{Truncate}(\mathcal{D}_0, L)$  is a function which returns a data set  $\mathcal{D}$ , which is a subset of  $\mathcal{D}_0$ , obtained by pruning away data sets with germ power  $> L$ .

$\text{Argmin}(\mathcal{S}, \mathcal{G}, \mathcal{D}, \vec{\theta}_1)$  is a function which returns the  $\vec{\theta}$  at which statistic  $\mathcal{S}(\mathcal{G}(\vec{\theta}), \mathcal{D})$  is a minimal, using a local optimiser seeded at  $\vec{\theta}_1$ . The optimiser can be called with a variety of minimisation methods, these include: simplex, brute, evolve, swarm and multiple others.

Figure 3: Pseudocode of GST algorithm

**No deep knowledge** regarding the mathematics behind this algorithm is required to run a GST experiment. It may greatly help to know the background when interpreting the results, but to run an experiment it is sufficient to know that GST is a protocol which fits a model to experimental data using some constraints.

### 2.2.4 Parameterisation

**A superoperator** has to meet two constraints to be considered physically valid. It must be trace-preserving (TP) and completely positive (CP). The GST protocol considers these constraints on  $\mathcal{G}$  parameters which can be used to define a smaller gate set model.

In order to enforce TP, GST locks the first row of every superoperator to be  $[1, 0, \dots, 0]$  and sets the first element of every state preparation vector to  $\frac{1}{\sqrt{2}}$  [17]. Enforcing TP is therefore quite trivial.

The CP constraint is harder to define and impose than the TP constraint, as the TP constraint is a constant equality and the CP constraint is a nonlinear equality. [17] defines several ways to impose the CP constraint, which all require mathematical definitions outside the scope of this paper.

Usually, it is preferred to run GST with both of these constraints (CPTP).

### 2.2.5 Gauge Optimisation

**The "catch" of GST** is that it returns an estimated gate set that is unique up to gauge transformations. These are the model parameters, which can be altered without changing the outcome of circuit probabilities. Albeit still altering the results of GST, like estimated superoperators and fidelity, as these are gauge-dependant. Gauge freedom exists because of the lack of a reference frame, meaning that the strength of GST, is also its weakness. As the authors of Gate Set Tomography write: "Gauge optimisation is the act of reporting GST estimate, in a gauge that optimises some gauge-variant metric of closeness to the ideal target gate set. In other words, we choose the gauge to make the gates look as good as possible" [17]. The final step of GST is called gauge optimisation, which optimises these gauge parameters using the squared Frobenius distance. The theory behind gauge optimisation is outside of the scope of this paper but can be found in section 6.2.2 of [17].

### 2.2.6 Non-Markovianity

The metric used to analyse the results of a GST run is the *goodness of fit* of the estimated gate set. If the estimated gate set fails to fit the data produced it is an indication that some assumptions of the model were violated. We will call such violations "non-Markovianity". However, GST does not focus on finding out where this non-Markovian noise is coming from, it focuses on detecting when the GST model has failed to fit the data. In this sense, it is one of the many tools in the toolbox of the quantum engineer who wants to characterise a quantum system in order to optimise it. In order to quantify the non-Markovianity GST uses Wilk's theorem [21] which, when applied to GST, states that: "if the gate set model is valid, then the loglikelihood ratio statistic between them is a  $\chi_2^k$  random variable" [17], that is:  $2(\log \mathcal{L}_{max} - \log \mathcal{L}) \sim \chi_2^k$

Where  $\log \mathcal{L}$  is the likelihood of the calculated model and  $\mathcal{L}_{max}$  is the likelihood of a trivial "maximal" model which assigns one probability for each independent outcome. As  $\chi_2^k$  has mean  $k$  and standard deviation  $\sqrt{2k}$  we can now quantify the model violation by using:

$$N_\sigma \equiv \frac{2(\log \mathcal{L}_{max} - \log \mathcal{L}) - k}{\sqrt{2k}} \quad (4)$$

$N_\sigma \leq 1$  indicates a good fit, while  $N_\sigma \gg 1$  indicates a model violation, meaning no gate set can explain all the data [17].

### 2.2.7 Error Generator





Sector	Dimension	Action	Example effect (Bloch sphere)	$\epsilon_j$ (Error Probability)	$\theta_j$ (Error Amplitude)
Hamiltonian $\mathbb{H}$	$d^2 - 1$	$H_P[\rho] = -i[P, \rho]$		0	1
Stochastic (Pauli) $\mathbb{S}$	$d^2 - 1$	$S_P[\rho] = P\rho P - \mathbb{I}\rho\mathbb{I}$		1	0
Stochastic (Pauli-correlation) $\mathbb{C}$	$\binom{d^2 - 1}{2}$	$C_{P,Q}[\rho] = P\rho Q + Q\rho P - \frac{1}{2}\{\{P, Q\}, \rho\}$		0	0
					1
Active $\mathbb{A}$	$\binom{d^2 - 1}{2}$	$A_{P,Q}[\rho] = i\left(P\rho Q - Q\rho P + \frac{1}{2}\{\{P, Q\}, \rho\}\right)$		0	1
					0

Figure 4: Properties of subspaces of error generator. First column shows subspace in question. Second column shows dimension of subspace, third column shows physical action. Fourth column shows effects of error on the Bloch sphere. Finally, fifth and sixth column show error probability and amplitude. Figure adapted from [22].

Another important metric in the context of Gate Set Tomography is the gate error generator. This metric is less used in literature, but offers more insight, as it can quantify the error within an operation relative to an ideal target operation. When constructing the error generator we assume  $G = e^{\mathbb{L}}G_0$ , where  $G$  is our physical noisy superoperator,  $G_0$  is our ideal target superoperator and  $\mathbb{L}$  is the error generator [22]. That is, we are modelling our errors by applying a second "noise" gate after our perfect gate. We can then project the resulting error generator matrix onto subspaces of error generator space, to analyse the different components of the error generator.

[22] defines three classes, and 4 subspaces, of elementary errors, from which these subspaces are derived. These can be seen in figure 4, with additional information.

1. **Hamiltonian generators**, which are described by unitary error processes in subspace  $\mathbb{H}$ . These can be seen as over or under rotations. These occur, for example, if a control mechanism is faulty, which results in our qubit missing its target state. These errors are well understood, and can often be solved by dynamical decoupling, or re-calibrating control. It can be visualised by a rotation of the Bloch Sphere, resulting in displaced axes.
2. **Stochastic generators**, which are described by convex mixtures of Hamiltonian generators. Subspace  $\mathbb{S}$  and  $\mathbb{C}$  are defined, describing Pauli-stochastic and Pauli-correlation sectors respectively. An example of stochastic errors is minimally disturbing measurements. These can be seen as the environment "measuring" the system, which leads to quantum information leaking out, resulting in decoherence. These can be visualised as the axes of the Bloch sphere contracting or expanding, resulting in depolarisation.
3. **Active generators**, which consider "everything that is left"[22], that is these generators are achieved by defining the complement of the subspaces of  $\mathbb{H}$ ,  $\mathbb{S}$  and  $\mathbb{C}$ . These are "relatively mysterious", and the only example the authors of [22] give are  $T_1$  decay processes. They can be visualised by rotations, contractions, expansions or shifts of the Bloch sphere axes.

In this research, we will swap our target ideal superoperator with a superoperator obtained by running GST on simulated data. That is, we assume that our simulation is our ideal system, and we use this system as a reference frame when calculating this metric. We do this to evaluate if our simulated models capture the noise generated within our physical system. If our simulation succeeds in capturing sources of noise, we expect a lowered proportion of stochastic errors, as this subspace captures errors due to decoherence.

### 2.2.8 Diamond Norm

**The final metric** we will introduce is the diamond norm. The diamond norm is a completely bounded trace norm in the space of quantum operations. It compares superoperators by measuring the "single-use distinguishability". It can informally be described as follows: Assume we want to know how similar superoperator A and superoperator B are. We present an agent randomly with either A or B, we then permit the agent to apply the given superoperator on one quantum state. He may then measure the state, and attempt to determine which superoperator he was given. The maximal probability of success, of the agent, is then determined by the diamond norm of the difference between the two superoperators.

The diamond norm is mathematically defined as:

$$\|G - G_0\|_\diamond = \sup_\rho \|(G \otimes \mathbb{1})(\rho) - (G_0 \otimes \mathbb{1}_d)(\rho)\|_1 \quad (5)$$

Where  $G$  and  $G_0$  are the gates we are comparing and  $\rho$  ranges over all valid quantum states.

Solutions to this problem are hard to obtain analytically but can be computed efficiently with semi-definite programming, as is done in pyGSTi [23].

## 2.3 Nitrogen-Vacancy Quantum Bits

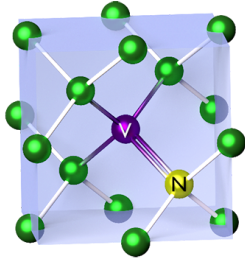


Figure 5: Structure of NV centre qubit. Green spheres are carbon nuclei, yellow sphere is the nitrogen nucleus and purple sphere is a vacancy.

**One possible way** to implement a qubit is by using an electron spin inside an NV centre in a diamond lattice [24]. These qubits are realised by replacing a carbon atom in these lattices with a nitrogen atom, as seen in figure 5. This creates a vacancy in which an electron can then be manipulated using microwave (MW) waves enabling quantum computing.

These qubits have a long coherence time ( $T_2^* \approx 1.5\text{ms}$ ) [26]. They can also work at a large variety of temperatures (up to room temperature  $\approx 290\text{ K}$ ) [27]. The key features of NV centre qubits are quite handy in, for example, quantum network applications.

An NV centre system can be regarded as a single qubit system, in which one only considers the electron spin. But it can also be seen as a two-qubit system as this electron spin may couple to the nucleus spin of the Nitrogen atom. Applying single qubit GST to the NV system, where the Nitrogen nucleus is not properly initialised, may produce faulty results, as the coupling of the spins produces non-Markovian behaviour.

**Control** of the qubit is obtained by a sequence of MW pulses with well-defined amplitude, frequency and phase. By carefully controlling the pulse the qubit can be initialised to the  $|0\rangle$ ,  $|1\rangle$  or a superposition state. This works, because of the Zeeman effect which states that the spin-up and spin-down states of systems will have different resonance frequencies under the influence of a static magnetic field [28]. This enables us to "address" the different states the spin can be in, and therefore control it.

**An echo** can also be utilised, which is a sequence of pulses applied after every quantum gate to minimise decoherence. It can be seen as a "kicking" the quantum system to prevent it from "relaxing". In the results section, we will compare data where echos are applied to data where this is not the case. Echos are especially interesting in the context of NV centre systems, as the surrounding nuclei spins couple to the system. Therefore, there exist a myriad of noise sources within these systems, which threaten the lifespan of the quantum information. We will use the XY4 pulse sequence described in [25]. This echo consists of the X and Y gate, and a wait duration  $\tau$ . The pulse sequence can be seen in figure 6 and is applied after every gate pulse.

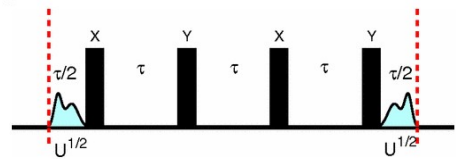


Figure 6: The pulse sequence of an XY4 echo. Adapted from [25]

**Measurement** of the quantum state is achieved by an optical laser pulse of green light (532nm). The reason why this works is that there are different optical transitions, which are associated with different spin states, as seen in figure 7. If a laser pulse is applied which is only resonant with the transition to spin-up, then only when the NV centre is in the spin-up state will it be excited and photons will be detected. That is, we are using state-dependent fluorescence, only if the electron is in a specific state will light be emitted at a specific frequency. This enables us to detect which state the electron is in. The problem is often that at least one photon needs to be detected, which is not a trivial problem. In an experimental setup, there is often a plethora of equipment available to improve the light extraction and shield the system from thermal bombardment and electromagnetic interference.

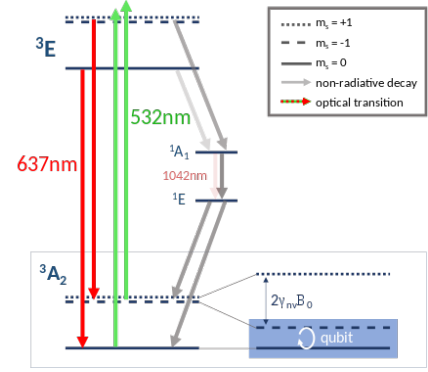


Figure 7: Energy levels of NV system. The spin states of the electron is shown, and the transitions with corresponding wavelengths. See [29] for more information.

### 3 Python GST Implementation

The primary tool used for this project is the open-source framework python GST implementation (pyGSTi). We will explain the way pyGSTi is constructed, and how it can be used.

On 05-10-2021 a detailed paper discussing GST was published [17] alongside an open-source Python framework implementing GST techniques [23]. In the short span since it has been introduced, pyGSTi has already been adopted and used by research groups worldwide. It has been described as: "emerging as a gold standard for detecting and describing problems inside quantum computing hardware" [30].

pyGSTi has amongst others been used to:

- Calculate the fidelity of two qubit gates for semi-conductor spin qubits. [31]
- Calculate the fidelity of three qubits gates for donor quantum processor in silicon. [32]
- Detecting crosstalk errors in quantum information processors. [33]

We will now delve deeper into the basic building blocks of the pyGSTi framework.

#### 3.1 Basic objects

**Circuits** are the most basic building blocks within pyGSTi. A `Circuit` object implements a quantum circuit by representing it as a sequence of layers. Circuits consist of two parts; The gate label parameter defines the order of the gates within the circuit, and the label allows one to address a qubit. This is an optional parameter and if set to empty will default to 0, 1, ..., n - 1 for n qubits. The circuit as a whole is represented by a list, where every element of the list is a circuit layer. If one then wants to apply multiple gates one simply makes that layer a list, creating a nested list.

**A Model** in the simplest form represents the physical capabilities of a physical quantum processor, and can therefore predict outcome probabilities when applied to quantum circuits [23]. Within pyGSTi they are used to run the different algorithms like GST, Randomized Benchmarking (RB) and Robust Phase Estimation (RPE). A `Model` object encapsulates a set of quantum gates along with state preparation and measurement (POVM) operations, and after running a characterisation protocol also encapsulates the results of said protocol. Models come in two types: `ExplicitOpModel` and `ImplicitOpModel`.

**The `ExplicitOpModel` object** is usually used for quite simple models that can often be defined in a few lines of code. It takes three inputs: a list of labels of the available qubits, a list of labels of the available gates and a list of strings describing the gate operations. It can then be further elaborated, for example by specifying the germs and fiducials used.

**The `ImplicitOpModel` object** can be seen as an extension of the `ExplicitOpModel` object in the sense that when a program is run it constructs an `ExplicitOpModel` object on the fly. The layers the `ImplicitOpModel` constructs are based upon the building blocks and layer rules of the model. pyGSTi currently implements two types of `ImplicitOpModel` objects:

1. The `LocalNoiseModel` objects are noise models where "noise" (the departure or deviance from perfection) of a given gate is localised to only the qubits where that gate is intended to act [23].

2. The `CloudNoiseModel` objects allow imperfections in a gate to involve qubits in a neighbourhood of or cloud around the gate's target qubits [23].

As seen in the type of implementations implicit models are mainly used for complicated multi-qubit systems with stringent noise requirements.

**The ModelPack** is a set of pre-constructed models offered by pyGSTi. These different models vary in the number of qubits, gates and parameters. They are developed to enhance the usability of pyGSTi, as for most research projects one of the example models can be used directly, or tweaked to fit the project, which is the case for this project.

**The ExperimentDesign object** is used by a characterisation protocol to specify what data it needs. In its simplest form, it is composed of a list of `Circuit` objects. However in GST, for example, it requires additional data, such as the germs and fiducials used and the circuit depth.

For the purposes of GST there exists the `GateSetTomographyDesign` object, which inherits from the `ExperimentDesign` object. If this model does not contain or has the wrong, germ and fiducial circuits, the resulting object will not be able to run GST, as it will throw an exception.

**The Dataset object** is the main input/output (IO) entity found within pyGSTi. It is constructed as a dictionary in which the key is a circuit and the values are the counts of the measurements of said circuit.

The `Dataset` object itself is quite simple. Although the true strength of pyGSTi exhibits itself in the various modules used to interact with the `Dataset` object. PyGSTi offers support for: custom text format, JSON format, experiment design directory and Pickle format.

In this project, the JSON format will be used to construct the circuit lists to be run on the system. The outcome counts of the circuit list are then delivered back in NumPy arrays by experimentalists using .npy files. Finally, it is written into a `Dataset` object using a custom parser.

**The Results object** is what pyGSTi delivers after running a characterisation protocol. In short, a `Results` object holds a given experiment design together with the outcome of the characterisation algorithm. The outcome is stored in a `Model` object, and also includes a dictionary of gauge-optimisation parameter dictionaries. One thing to take into account is that GST is an iterative protocol. This has as result that the results object stores a model for every iteration. After generating a `Results` object it can either be written to a directory for later retrieval or be further processed into a `Report` object.

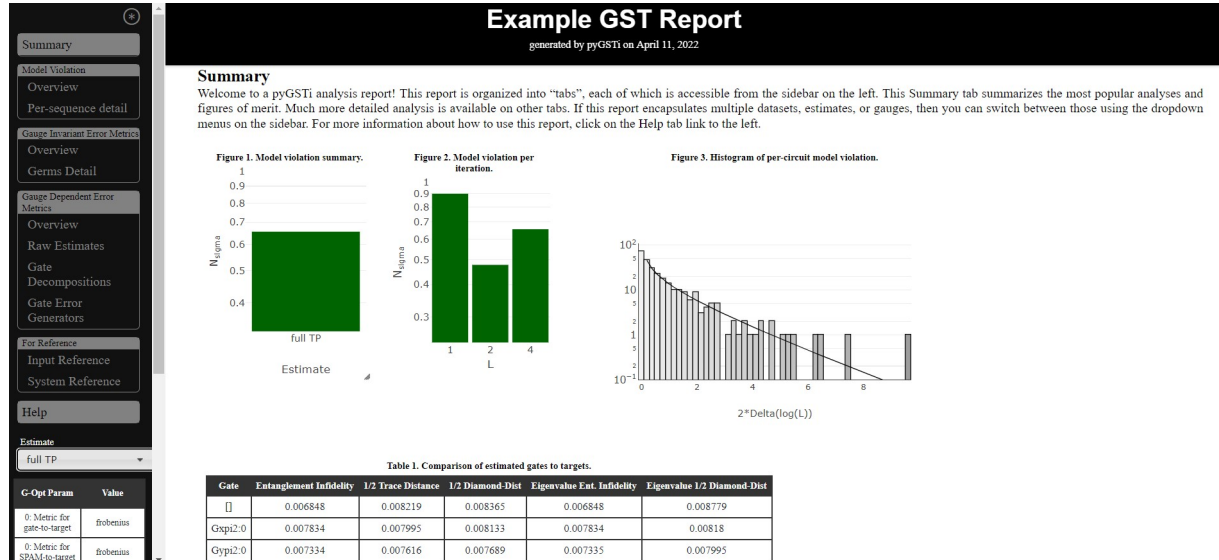


Figure 8: An example of a Report in HTML format. This figure displays the summary tab, which shows the  $N_{\sigma}$ , diamond norm, trace distance and various other metrics. Various other tables, figures, plots and metrics can be found on other tabs.

**The Report object** is the main way in which pyGSTi communicates its results to a user. The report contains different metrics and data about the characterisation of the quantum system. In figure 8 a screenshot of an example report is displayed.

## 3.2 Algorithms

**Fiducial selection** is often the second step when running a GST experiment (after model construction). As explained in 2.2.1 the purpose of preparations and measurement fiducials is to prepare an adequately diverse set of input states and measurements to completely explore our operations of interest. Thus we can create estimations of the superoperators with a high goodness of fit.

pyGSTi abstracts a lot of complicated physics and mathematics down to a function with several parameters. The function used for fiducial selection is:

```
pygsti.algorithms.fiducialselection.find_fiducials(target_model, omit_identity=True,
eq_thresh=1e-06, ops_to_omit=None, force_empty=True, max_fid_length=2, algorithm='grasp',
algorithm_kwargs=None, verbosity=1)
```

Out of all these parameters only the `target_model` is necessary (which we constructed in step one). The other parameters have default settings, which is sufficient in most cases. The only parameter that had to be overridden was the `max_fid_length`, as more powerful models might need larger fiducials.

Note that this method returns both the preparation and measurement fiducials.

**Germ selection** is usually the step performed after fiducial selection. As explained in 2.2.2 the defining property of a germ set is that it is AC, which means that it amplifies all possible gate errors. Or more specifically: the repetition of a set of germs sandwiched between IC fiducials will yield a sensitivity that scales with the germ power  $p$  (the amount of times the germ is repeated) to any direction in the model space.

The function signature of the germ selection function is:

```
pygsti.algorithms.germselection.find_germs(target_model, randomize=True,
randomization_strength=0.01, num_gs_copies=5, seed=None, candidate_germ_counts=None,
candidate_seed=None, force='singletons', algorithm='greedy', algorithm_kwargs=None,
mem_limit=None, comm=None, profiler=None, verbosity=1)
```

This function takes more parameters than the `find_fiducials` function, although once again only the `target_model` parameter is required.

## 3.3 Protocol

**GST** is the main functionality and motivation for pyGSTi.

To run GST one needs three inputs:

1. A target model, which describes the operations our quantum processor can perform.
2. A list of circuits which our quantum processor needs to perform note that this is simply our germs sandwiched  $p$  times between our fiducials. This is also known as the experiment design.
3. A data set in the form of experimental measurement counts for each circuit in the circuit list.

When we have collected our required inputs pyGSTi offers two main `Protocol` objects with which to run GST:

- **GateSetTomography** which runs a single model optimisation on an initial model with any parameterisation the user likes. This protocol can run on any `GateSetTomographyDesign` experiment design.
- **StandardGST** which runs multiple model optimisations based on an `ExplicitModel` target model by parameterising this model in different ways. This protocol can only run on the `StandardGSTDesign` experiment designs since the usual germs and fiducials structure is expected.

Usually, the `GateSetTomography` protocol is more flexible than the `StandardGST` protocol, as the input of this protocol is more complicated. However, this has as a result that it generally takes more effort to use this object.

**Other characterisation protocols** are also contained within pyGSTi, but will not be explained in detail as they were not used for this project. It suffices to say that pyGSTi can perform: circuit simulation, drift characterisation, mirror circuit benchmarks, randomised benchmarking and robust phase estimation.



## 4 Methodology

The **workflow** when conducting a GST experiment can be seen in figure 9.

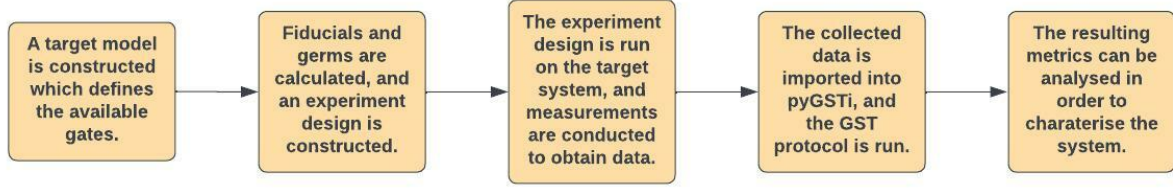


Figure 9: Flowchart of GST workflow.

We will now explain in detail how these different steps were undertaken in this research. Note that all GST runs were conducted with the CPTP and TP constraints described in section 2.2.4.

### 4.1 Model and Experiment Design Construction

3 gate model	
fiducials	germs
$X^{\frac{1}{2}}$	I
$Y^{\frac{1}{2}}$	$X^{\frac{1}{2}}$
$(X^{\frac{1}{2}} X^{\frac{1}{2}})$	$Y^{\frac{1}{2}}$
$(X^{\frac{1}{2}} X^{\frac{1}{2}} X^{\frac{1}{2}})$	$(X^{\frac{1}{2}} X^{\frac{1}{2}})$
$(Y^{\frac{1}{2}} Y^{\frac{1}{2}} Y^{\frac{1}{2}})$	$(I I I I Y^{\frac{1}{2}})$
	$(I I I I X^{\frac{1}{2}} Y^{\frac{1}{2}})$
	$(I I I I Y^{\frac{1}{2}} X^{\frac{1}{2}})$
	$(I I Y^{\frac{1}{2}} X^{\frac{1}{2}} X^{\frac{1}{2}} X^{\frac{1}{2}})$
	$(X^{\frac{1}{2}} X^{\frac{1}{2}} Y^{\frac{1}{2}} Y^{\frac{1}{2}} X^{\frac{1}{2}} Y^{\frac{1}{2}})$
	$(I I X^{\frac{1}{2}} X^{\frac{1}{2}} X^{\frac{1}{2}} Y^{\frac{1}{2}})$

Table 1: Fiducials and germs of the 3 gate model

**Although multiple models were constructed,** two primary models have been used for the final results of this project. The first model used was provided by Dr Yun and Mr Bartling together with experimental data sets. This model was a modified version of the `smq1Q_XYI.py` model from pyGSTi's `ModelPack`. It describes a one-qubit system with an X, Y and I gate. The experimentalists had extended this model by adding the  $X^{1/2}$  and  $Y^{1/2}$  gates. A point of notice is that the germs and fiducials used were all five individual gates. This germ set is in theory not AC, therefore it is a sub-optimal solution. Due to the structure of GST, germs can not be modified after data has been collected, as the germs make up the centre of the circuits used to collect the data. Furthermore, the results obtained by running this experiment design are still valid, albeit that they are less sensitive to certain errors.

**The second model** is a three-gate model utilising the X, Y and I gates. The `find_fiducials` and `find_germs` methods were used to find a set of appropriate fiducials and germs. We also constructed a three-gate model which uses germs and fiducials found in "Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography" [34] to compare the effectiveness of GST on these varying fiducial and germ sets.

Due to a mix up the experimental data was generated by a model which uses the fiducials of [34] and the germs which we computed ourselves. This should have no consequences for the results as the fiducials set is IC, and the germ set is AC.

Both models use the same set of fiducials for both preparation and measuring, the only difference being that the 3 gate model does not use the I gate as a fiducial. The fiducials and germs of said models are summarised in Tables 1 and 2.

After constructing these models, `ExperimentDesign` objects were generated, and the resulting circuit lists were sent to the experimentalists in JSON format to receive measurement data.

5 gate model	
fiducials	germs
I	I
$X^{\frac{1}{2}}$	$X^{\frac{1}{2}}$
$Y^{\frac{1}{2}}$	$Y^{\frac{1}{2}}$
$(X^{\frac{1}{2}} X^{\frac{1}{2}})$	X
$(X^{\frac{1}{2}} X^{\frac{1}{2}} X^{\frac{1}{2}})$	Y
$(Y^{\frac{1}{2}} Y^{\frac{1}{2}} Y^{\frac{1}{2}})$	

Table 2: Fiducials and germs of the 5 gate model

### 4.2 Experimental Data

**Six experimental data sets** were received during this research. This data was generated by an NV system under the care of the Taminiau group at QuTech. Two of these data sets were obtained by running the 5 gate model experiment design, and the other four with the 3 gate model. The differing factor between these data sets is the initialisation state of the nitrogen nucleus.

For the 5 qubit model, one of the data sets has the nitrogen nucleus initialised to the 0 state and the other one to the mixed state. The mixed state means that we are not initialising the nitrogen nucleus, it is therefore a mix of the three possible states it can be in. However, the initialisation procedure is not perfect and has a fidelity of approximately 90-95%.

The 3 qubit model is run with four different states: p1, m1, 0 and mixed. The p1, m1 and 0 states refer to the energy levels of the nitrogen nucleus spin, for this research we will say that in these cases the nitrogen has been initialised, as these three states should deliver (roughly) the same results.

One other important differing factor is that the 5 gate model was run using bare gates, while the 3 gate model runs employ an XY4 echo, which in principle should lead to lower decoherence (as mentioned in section 2.3). The data was delivered as a NumPy array in a .npy file and was loaded into pyGSTi using a custom parser.

### 4.3 Simulated Data

**The experiment designs** were also run on a simulator, as opposed to the physical system for three reasons. Firstly, running these experiments on a simulator lets us cross-reference the results with experimental data. Secondly, a simulator is less of a black box than a physical system, and can easily be tweaked (for example the nitrogen initialisation fidelity). Thirdly, the physical system was in the use of the research group and at the time not available.

The simulator in question is a numerical solver written in the QuTiP framework and implemented by the Taminiau research group of QuTech. QuTiP stands for Quantum Toolbox in Python and is a widely used open-source framework for simulating the open quantum dynamics of a system. The mechanics behind the simulator, which was used for this research, can be found in [35].

**To generate simulated data** two experiment designs were used. The models used to construct these experiment designs are the two models described in 5.1. All of these experiment designs were run under three different nitrogen initialisation parameters. The first data set was generated by setting the parameter to 0, for the second data set the parameter was set to 1, and for the third data set the parameter was set to 0.95 (to mimic the physical system). All simulated data sets of the 3 qubit model were run with an XY4 echo, to analyse the effects of the echo, when compared with the physical system. These data sets were then fed back into pyGSTi, and reports were constructed to analyse the results.

## 5 Results and Discussion

**The main functionality of GST** is to characterise a gate set by applying the steps explained in section 2.2. The resulting report carries a plethora of information, with many different metrics. For this project, a total of twenty-nine GST reports were generated. However, we will only report on twelve to limit the scope of this project. All data sets and accompanying reports can be found at <https://github.com/davidvos99/GST-for-NV-systems>.

In this section, we will discuss the results presented by pyGSTi, and try to find causes for any discernible discrepancies. We will use the error generator, diamond norm and  $N_\sigma$  metrics to discuss differences observed between models where the nitrogen nucleus was initialised, as opposed to not initialised. After this, we will report on the effects of the XY4 echo that the 3 gate models are implicitly using. We will then discuss the discrepancy found in the CPTP parameterised models as opposed to TP parameterised models. Evidence, which alludes to calibration faults, is also discovered within the diamond norm. Finally, we will discuss the GSTBlochWidget in the context of visualising errors and possible applications.

### 5.1 Error Generator Metric

**The error generator** is one of the most useful metrics pyGSTi has to offer, especially in the context of comparing models. Projecting the error generator on different subspaces enables us to make statements as to the nature of these errors.

An example of an error generator report can be seen in figure 15 in appendix C. This specific figure shows the error generators of the 3\_gate\_exp\_init\_state=m1 comparative to the ideal target model. The first column displays a heat map of the estimated error generator, that is it shows the error generator  $\mathbb{L}$  as explained in section 2.2.7. As  $G = e^{\mathbb{L}}G_0$ ,  $\mathbb{L} = 0$  means a perfect fit, the heat map shows how much the estimated values deviate from 0.

In the corresponding columns, the error generator is projected onto different subspaces of error generator space. These projections are divided into the X, Y and Z basis, which enables us to see how



much each basis has been affected by this type of error. The report also displays how much of the error generator is captured within each subspace, allowing us to quantify the proportions of the types of errors observed.

**For this project** we use the standard error generator which pyGSTi constructs when delivering a report. In these error generators, the default target model is the ideal model with perfect gates. Furthermore, pyGSTi code was edited to construct error generators where the target model has been replaced by a simulated model. See appendix C for an overview of the error generator results.

### 5.1.1 Trends and Outliers

We will now state some points of notice regarding the used models, we will then observe general trends and finally report on outliers of these trends.

**One point of notice** is that the experimental 3 gate model where the nitrogen was initialised to the 0 state has extremely high non-Markovianity.

Also, note that the 3 gate models utilise XY4 echos. Finally, we remember that the 5 gate model uses single gate germs, which consequently means that these results will be less sensitive to errors. Nevertheless, the results are still valid and might point us in interesting directions to explore in further research.

**General trends** seem to be well established in the data. We will summarise these below:

1. Most of the errors seem to be captured in the  $\mathbb{H}$  subspace. This seems especially the case for the 5 gate models and the models where the nitrogen nucleus has been initialised.
2. When the nitrogen nucleus is not initialised, that is `init_state=mixed`, a higher proportion of the errors seem to come from the  $\mathbb{S}$  subspace.
3. The proportion of errors from the  $\mathbb{S}$  subspace goes down when the target model is a simulated model.
4. When the nitrogen nucleus is not initialised, the proportion of  $\mathbb{A} + \mathbb{C}$  errors is lower when the target model is the `ideal_model` (as opposed to the simulated model) for the 3 gate model, and vice versa for the 5 gate model.

**Outliers** are also present within the data:

1. The I gate of the experimental 3 gate model with `init_state=0` has a high proportion of errors from subspace  $\mathbb{S}$  when the target model is the ideal model, these shift to  $\mathbb{H}$  when the target model is the simulated model.
2. The  $Y^{\frac{1}{2}}$  gate of the experimental 3 gate model with `init_state=0` is almost all because of stochastic errors, this does not change when changing the target model.
3. The  $X^{\frac{1}{2}}$  gate of the experimental 3 gate model with `init_state=mixed` has a high proportion of errors from subspaces  $\mathbb{S}$  and  $\mathbb{A} + \mathbb{C}$  when compared to the ideal model, these shift to subspace  $\mathbb{H}$  when compared to the simulated model.
4. The  $Y^{\frac{1}{2}}$  gate of the experimental 5 gate model with `init_state=mixed` has a high proportion of errors from subspaces  $\mathbb{A} + \mathbb{C}$  when compared to the simulated model.

### 5.1.2 Effects of Nitrogen Initialisation

**The biggest effect** that not initialising the nitrogen nucleus seems to have on the error generator, is that errors proportionally seem to be found more in the  $\mathbb{S}$  subspace (trend 2). This is within our realm of expectations, as not initialising the nitrogen leads to unaccounted coupling, which has decoherence as a consequence. Subspace  $\mathbb{S}$  represents these kinds of errors in the sense that decoherence leads to a shrinking Bloch sphere. Also, trend 1 comes into consideration here, as consequently the proportion of errors are transferred from the Hamiltonian subspace.

Especially the 3 gate model shows higher proportions of stochastic errors when compared to the 5 gate model. This might be because the germ set of the 5 gate model is not AC, meaning errors are less likely to be amplified and discovered. This partly explains trend 1, as this translates to the 5 gate models having a higher proportion of Hamiltonian errors.

Furthermore, as trend 3 states, the proportion of stochastic errors goes down when replacing the ideal model with our simulated model. This is a good sign, as this hints that our model captures some of the noise sources correctly, meaning that our errors shift from the stochastic realm to the Hamiltonian one. An interesting continuation would be to tweak the simulator and then repeat this research with new data. By tweaking the simulator, and looking at the shifts in error generator subspace proportion, we could better understand where the noise is coming from.

## 5.2 Diamond Norm

**The diamond norm** metric enables us to quantify how "close" two given superoperators are. That is, it gives us the probability that they can be differentiated when randomly applied. The diamond norm results can be found in the tables within appendix C. We observe that for all models the diamond norms are comparatively the same when ideal models are switched to simulated models. Although the values differ somewhat, the change is in most cases lower than 20%. This means that the gates of our experimental models are approximately as close to the ideal gates, as the gates of our simulated models.

**One point of notice** is the diamond norm of the  $X^{\frac{1}{2}}$  and  $Y^{\frac{1}{2}}$  gates of the 3 gate models with `init_state=m1` and `init_state=p1`. The diamond norm for the  $X^{\frac{1}{2}}$  gate is approximately twice as large for the `init_state=m1` model compared to the `init_state=p1` model, and vice versa for the  $Y^{\frac{1}{2}}$  gate. This is also reflected in the error generator and estimated superoperators of these models.

This could be explained by the fact that the m1 and p1 states have a slight difference in energy. When controlling an NV centre MW pulses calibrated to a specific frequency is used, which correlates with these energy levels. If these pulses are not calibrated correctly, a possible outcome could be this discrepancy of a factor 2, in the  $X^{\frac{1}{2}}$  and  $Y^{\frac{1}{2}}$  gates, which we observe.

The error generator also supports this theory, as virtually all errors are Hamiltonian. This means that these are coherent errors, which mostly occur due to calibration and control faults. To conclusively prove this hypothesis, research would need to be conducted regarding the pulse calibration of this system, and more data collected where the system has been initialised to the p1 and m1 state.

## 5.3 Non-Markovianity

**The  $N_\sigma$  metric** discussed in 2.2.6 is the primary metric of pyGSTi, as it can quantify how much a model has been violated. The  $N_\sigma$  has been calculated for both CPTP and TP parameterised models, as any discrepancies might be points of interest. The tables showing the  $N_\sigma$  metric can be found in appendix C.

### 5.3.1 CPTP vs TP

**Comparing CPTP results with TP results** has an interesting application as a "sanity check". As the authors of [17] put it: "In practice, we have found it useful to perform longsequence GST using both the CPTP and TP gate set models, and compare the results. If a significantly better non-CP fit is found, then the two estimates should be examined carefully: either the CPTP fit got trapped in a local maximum, or significant non-Markovian dynamics occurred during the experiment."

Therefore, we will briefly compare the  $N_\sigma$  metric of our models when ran under CPTP and TP parameters.

In table 3 no noteworthy differences appear to exhibit themselves. The biggest difference, proportionally, is 12.7%, therefore, the differences in  $N_\sigma$  are not significant. We do observe that the TP runs generate a slightly lower  $N_\sigma$ . This could be explained by the fact that having fewer constraints allows the algorithm to find a better fit, albeit with the chance that the outcome is nonphysical.

In table 4 we see similar results to table 3, albeit with one big exception. The experimental model with `init_state=0` has an enormous difference in  $N_\sigma$ , which is multiple orders of magnitudes larger for the CPTP parameterised model.

Also, note that outliers 1 and 2 are from this model. On closer inspection, we find that the non-Markovianity finds its origin in the  $Y^{\frac{1}{2}}$  gate. The errors produced by this gate are almost entirely stochastic, and the estimated superoperator of said gate fails to fit the target superoperator. We also observe this in the diamond norm, which has an extremely big value, when compared to the other gates. As mentioned before, any discrepancies might hint at a local maximum or significant non-Markovian dynamics. To pinpoint which one of these causes it might be further research is needed, this will be expanded upon in section 7.

### 5.3.2 Effects of XY4 Echo

**The biggest effect** that the XY4 echo seems to have, is also found in the  $N_\sigma$  metric. Although the error generators have somewhat differing proportions, there are little to no significant differences. However, when we look at the  $N_\sigma$  metric, we instantly notice that it is extremely high for the 5 gate model with `init_state=mixed`, while the values for the 3 gate model with `init_state=mixed` are similar to the other 3 gate model values. This is extremely interesting, as the echo seems to combat the non-Markovian consequences of coupling well. This can also be seen in the diamond norm metric, where the `init_state=mixed` has a lower diamond norm than the other 3 gate models, meaning the gates in this model are "closer" to the target gates. The XY4 echo could be a powerful tool in constructing future quantum processors and algorithms with higher fidelity. Further research would be needed to characterise and understand the implications of this operation better.

## 5.4 GSTBlochWidget

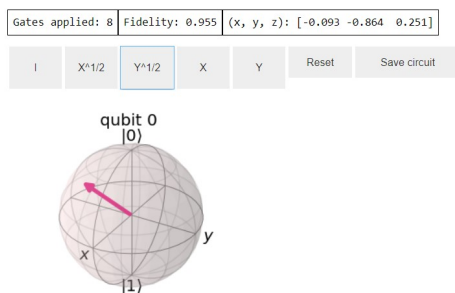


Figure 10: A screenshot of the GSTBlochWidget. Upper textboxes show additional data. Buttons can be used to apply gates, reset the quantum state and save applied circuit to a text file.

Furthermore, the `plot_bloch_multivector` function of the qiskit library is used to render the Bloch sphere.

**The back-end** of the GSTBlochWidget is a home-written Python script. Although the widget itself can only be properly used within a notebook, all of the functionalities are imported. The widget was developed in such a manner to make it simple to swap out results, so GST runs can easily be compared to each other. In the Jupyter notebook itself only the result data from a GST run is loaded, the `GSTBlochWidget` object is declared and the `render_widget` function is called. The `render_widget` function implements the `on_button_click`, `apply_gates` and `update_output` functions inline. These are called when a button is clicked, and make the widget interactive.

It takes as initialisation parameters a list of gate labels, the resulting estimated superoperators, the ideal superoperators and a file path to a directory where circuits may be stored. It stores all of these parameters and then keeps track of a current density matrix and a target pure quantum state.

The density matrices are changed by calling the `evolve` function. The `evolve` function uses the identity  $|\rho\rangle\rangle = G|\rho\rangle\rangle$ . Note that the conversion from density matrix to superket and vice versa can be accomplished using the formulas given in Appendix A. Also note that calculating the Bloch vector coordinates using these formulas is trivial, and is also done by the `GSTBlochWidget` to display these as additional data.

The `GSTBlochWidget` also calculates the circuit fidelity using the formula  $F = \langle\psi|\rho|\psi\rangle$ , where  $F$  is the fidelity,  $|\psi\rangle$  is the target ideal state and  $\rho$  is the actual density matrix. Lastly, the `GSTBlochWidget` saves all gates applied, intermediate fidelity and density matrices, to print these to a text file once the save circuit button is clicked. See the UML diagram or code example in appendix B for a concise overview.

After running pyGSTi on the first data sets, it was in some cases challenging to interpret the results. Although pyGSTi offers an accessible HTML report, with a plethora of data, it can be somewhat overwhelming. For this reason, an interactive widget was developed, which lets the user apply one of the estimated gates, and shows the resulting state on the Bloch sphere. An example of this widget can be found in figure 10.

### 5.4.1 Development of GSTBlochWidget

**The front-end** of the widget is rendered using the `ipywidgets` framework. This has as a result that some elements of the widget will only work within a Jupyter notebook. It uses the `Button` object to receive input, and the `Output` object to show the number of gates applied, the fidelity relative to an ideal state and the current Bloch vector coordinates.

### 5.4.2 Application of GSTBlochWidget

The **GSTBlochWidget** was developed to visualise gate errors. Although this tool is not able to quantitatively provide new metrics, it serves a purpose in the pedagogical sense. This tool enables us to easily see the actual effects of an estimated gate on the Bloch sphere. To give an example: we know from our results that the experimental 3 gate model with `init_state=0` has a high non-Markovianity. We also know that the errors in  $Y^{\frac{1}{2}}$  gate are almost entirely stochastic errors. After loading the results from this model and applying this gate we get the Bloch sphere presented in figure 11. We notice that our Bloch vector is almost at the centre, meaning that there is an extreme amount of decoherence. We also see this in the fidelity metric, which after one gate is as low as 0.511, meaning that we barely have a 50% chance of attaining our target state.

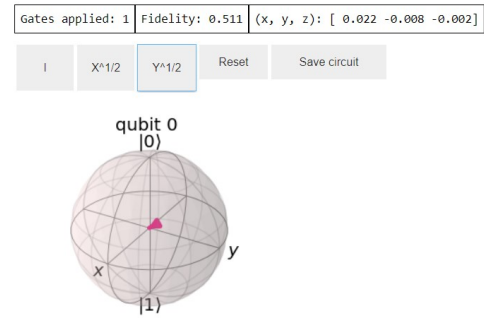


Figure 11: Widget of experimental 3 gate model with `init_state=0`, after one  $Y^{\frac{1}{2}}$  gate has been applied

**We knew from the other metrics** that this gate has a bad performance, but the GSTBlochWidget enables us to see this bad performance in action. The code for this widget has been shared with the research group at the Taminiau lab to assist with interpreting GST results.

## 6 Responsible Research

With negative attitudes and distrust towards science and research on the rise and a digital environment which facilitates the spread of misinformation, it is more important than ever that research be conducted ethically and responsibly. The scientific method itself is merely an empirical method designed to obtain knowledge. This method does not define concepts like ethics or responsibility, therefore it is up to the researcher to understand why this is important and to incorporate it in their research.

**Ethical consequences of quantum computer science** are far-reaching, albeit far in the future. The big challenges in quantum computer science today are to scale existing systems and to make them fault-tolerant. Without solving these challenges complex quantum algorithms will stay abstract conjecture without any real-world application. Before these are solved, it is hard to envision what the precise ethical consequences will be.

Although, some ethical consequences of the field are already clear. For i.e privacy, quantum computer science could have big ethical consequences. As mentioned in section 2.1, the Deutsch-Jozsa algorithm, does theoretically have the power to break RSA encryption, which is widely used for secure data transmission. Quantum cryptography also has the power to establish information-theoretically secure communication channels [14]. These are communication channels, which use the laws of quantum information to detect eavesdroppers, making the channel in theory unhackable. Being able to break existing encryption schemes and offer unhackable channels, this technology could revolutionise cybersecurity, thereby having major ethical consequences. But this scenario is still far off, as today's challenges are to prove that these systems can be scaled efficiently.

**Reproducibility** stands at the core of conducting research responsibly. The philosopher of science Karl Popper defined falsifiability as a standard of evaluation for scientific work. Falsifiability states that a theory is falsifiable if it can be contradicted by an empirical test on existing technologies [36]. Adhering to falsifiability makes a theory predictive and testable, thus ensuring that science has the tools to compare and favour different theories.

Of course, to falsify a theory the underlying experiments would need to be reproduced. Therefore reproducibility is, by extension, an integral part of falsifiability, which stands at the centre of scientific philosophy to this day. To make this research easy to reproduce, all used code and data sets have been uploaded to Github under the following link: <https://github.com/davidvos99/GST-for-NV-systems>. Furthermore, this report was written to explain the results as well as possible. The author can always be reached at the email address on the title page for further questions or explanations regarding this research.

## 7 Conclusion

### 7.1 Summary

**At the beginning of this research** we asked ourselves what the  $N_\sigma$ , diamond norm and error generator metrics might tell us when regarding a simulated model as an ideal model. We collected simulated and experimental data, from a 3 gate model and a 5 gate model. These models differed in nitrogen initialisation states, and the 3 gate model has XY4 echos applied. We then analysed this data and generated said metrics.

**We found that** initialising the nitrogen nucleus lowers the proportion of stochastic errors, while it expands the proportion of Hamiltonian errors. We also discovered that when we used the simulated models as target models, the proportion of stochastic models went down, meaning that the simulation captured a significant amount of the noise produced. A discrepancy was also observed in the  $X^{\frac{1}{2}}$  and  $Y^{\frac{1}{2}}$  gates of the 3 gate models where the nitrogen nucleus was initialised to the m1 and p1 state. This discrepancy hints at calibration faults. These results could be used in further research to pinpoint sources of noise and calibration errors within NV centre systems

**The XY4 echo** seemed to have no big influence on the proportions of errors. We did discover that the XY4 echo combats non-Markovianity extremely well by considering the  $N_\sigma$  metric. We see that by applying this echo we can attain low levels of non-Markovianity, without having to initialise the nitrogen nucleus. This produces a lot of questions for further research, as this mechanism could be refined and optimised to build quantum processors with higher target gate fidelity rates. For example, studies could be conducted into the nature of the surrounding noise, and what echo parameters best combat these.

**We also asked ourselves** how GST results could be presented in an intuitive visual manner. To this end, we developed the GSTBlochWidget, which proved its worth when visualising gates, that fell outside of our realm of expectations. The GSTBlochWidget enabled us to quickly and easily apply a certain quantum circuit with gates estimated by GST, while at the same time reporting useful metrics, like fidelity.

### 7.2 Future Work

**The GSTBlochWidget** could be further developed, and new interesting features could be added. For example, the number and complexity of metrics used could be further expanded upon, to deliver a more complete oversight of the system. One could also add options to easily run whole quantum circuits, instead of using buttons to apply gates. Frankly put, the amount of features one could imagine the GSTBlochWidget to incorporate in the future is overwhelming. The author also hopes that development on the GSTBlochWidget will not cease, and it will continue improving upon its capability to easily convey complex results.

**XY4 echos** were applied to the 3 gate model in this project. However, this echo was not made explicit, that is in the corresponding experiment designs the gates this echo consists of were not present. In further research these GST results would be rerun with the echos made explicit, so the gates the echo consists of can be characterised themselves.

**The XY4 echo and simulation** could also be refined. A study could be conducted where GST is performed on models with differing echo and simulation parameters. These results could then be used to optimise the echo, and better understand where the noise in NV centre systems is coming from.

**The evidence of calibration faults** found in the diamond norm of the models, where the nitrogen is initialised in the m1 and p1 state, should also be investigated further. More data can be collected where the NV system is initialised in these states, to conclusively show if these findings are correct, or if there might be some other reason for these discrepancies.

**Proper fiducial and germ selection** is an important part of GST. However, in this project, the 5 gate model uses single-gate germs on the experimental data. In further research more complete germ sets should be generated, leading to bigger experiment designs, and a higher sensitivity to errors.

**Two qubit GST** is the logical next step in the context of NV centre systems. The models could be expanded to regard the NV centre as a two-qubit system. In this case, the gate set produced should fit well, as the electron spin and nitrogen nucleus spin are both contained within the model. Therefore, the issue of non-Markovian behaviour due to unaccounted coupling is negated. Interesting quantum phenomena, like entanglement, could then also be characterised.

**Calculating Markovianity by deconstructing two-qubit GST** is an interesting continuation of the project mentioned above. Once two-qubit GST has been correctly implemented an analysis of Markovianity could be done. By deconstructing the data set into single qubit data set and applying single qubit GST to them two gate sets would be produced. In an ideal world, the two-qubit gate set would then be the tensor product of these gate sets. Of course, any discrepancy would hint at non-Markovianity and could be further analysed to find the causes.

**A pyGSTi pipeline** is also a project which could provide valuable speed-ups for future research. The author experienced that conducting a lot of GST runs can be time-consuming and mind-numbing work. Building a pipeline, which automatically performs multiple pyGSTi runs at once, would be a trivial software engineering project. The results of this project could then be enjoyed by quantum engineers worldwide, reducing the amount of unproductive time and increasing the time spent analysing a system.

### 7.3 Acknowledgements

This research could not have been conducted without the advice and guidance of Dr Johannes Borregaard. Seeing a project take shape, from abstract discussions to real-life implementation, is extremely fulfilling. Therefore, the author would like to express his utmost gratitude to Dr Borregaard, for making this project take shape and presenting this project.

Furthermore, this research was conducted together with the applied mathematics and applied physics student Zoya Polshchikova. Without her mathematical insight and physical intuition, interpreting many of the results and understanding the inner workings of GST would be challenging.

Lastly, the author would also like to express his gratitude to PhD student Hans Bartling and postdoctoral researcher Jiwon Yun for an introductory meeting regarding GST, and for sharing their experimental data. Mr Yun also offered continued support in the form of simulated data.

## A Quantum Computer Science

This explanatory introduction will focus on one qubit systems, as this research also focuses on one qubit systems. Note that the true magic of quantum computer science exhibits itself in multi-qubit systems, where entanglement is possible. For the interested reader we refer you further to [37] or [38].

### A.1 The Quantum Bit

In this subsection we will first introduce the quantum bit, after this, we will show how a quantum state can be visualised on the Bloch sphere. We will then introduce the density matrix, which is required to describe mixed states. Finally, we will explain the measurement operation, and what an effect is.

**The quantum bit** (qubit) is the quantum analogy of the binary digit (bit) used by classical computer science as its fundamental building block. Essentially a bit can be represented by any two-state system. However, in the quantum world, a qubit is represented by any quantum two-state system. The difference is that quantum systems may find themselves in something called a superposition.

When a quantum system is in a superposition, its quantum state is composed of a linear combination of basis states. In quantum computer science, one uses the bra  $\langle |$ -ket $| \rangle$  notation of quantum mechanics to denote quantum states. A ket is a  $2^n$  sized vector in a Hilbert space  $\mathcal{H}$ , where  $n$  is the number of qubits (it is also known as the state vector notation). The bra is the complex conjugate transpose of the ket. Of course, a vector means nothing without a basis in which to interpret it. For one qubit systems the rule is generally that if no explicit basis is given with a bra-ket, then the  $|0\rangle, |1\rangle$  basis is assumed (also known as the Z-basis). That is  $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$  and  $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

We will give a physical example to illustrate the concept.

Assume there is a free electron with a spin (angular momentum of elementary particles, see [38] for more information.) in either the spin-up state or the spin-down state. We will define: spin-up state =

$|0\rangle$  and spin-down state  $= |1\rangle$ . The wave function (mathematical description of a quantum state, see [38] for more information) of the electron is then represented by:

$$|\psi\rangle = c_1 |0\rangle + c_2 |1\rangle \quad (6)$$

with:  $c_1, c_2 \in \mathbb{C}$ ,  $p_{up} = |c_1|^2$ ,  $p_{down} = |c_2|^2$ ,  $|c_1|^2 + |c_2|^2 = 1$ , where  $p_{up}$  and  $p_{down}$  represent the probability to measure the spin-up or spin-down state respectively.

So if the wave function of the electron is  $|\psi\rangle = \frac{3}{5}i |0\rangle + \frac{4}{5} |1\rangle$ , then the probability to measure a spin-up state is  $|\frac{3}{5}i|^2 = \frac{9}{25}$ , and the probability to measure a spin-down state is  $|\frac{4}{5}|^2 = \frac{16}{25}$ .

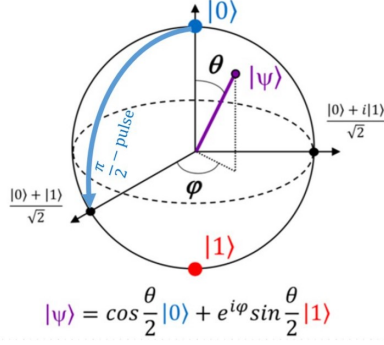


Figure 12: A Bloch sphere, the general form of a quantum state is given, and the blue line shows the effect of a  $\frac{\pi}{2}$  pulse.

**The Bloch sphere** is an intuitive way to visualise a quantum state. The way it is constructed is as follows: As seen before any pure state  $|\psi\rangle$  of a two-state quantum system can be constructed as a superposition out of the basis states  $|0\rangle$  and  $|1\rangle$  with complex coefficients  $c_1$  and  $c_2$ . This means that it takes four numbers to describe a single quantum state. However, the global phase of the quantum system is not physically measurable [37]. This means that only the relative phase between  $c_1$  and  $c_2$  has physical meaning. By setting  $c_1$  to a real non-negative number one can now describe the qubit with three numbers. This is also where the three dimensions (of which one is complex) come from.

It is also known from the laws of probabilities that  $|c_1|^2 + |c_2|^2 = 1$  [38] (one always has 100% chance to measure "something"), given this constraint and using the Pythagorean and Euler identities one can rewrite:

$$|\psi\rangle = \cos(\theta/2) |0\rangle + e^{i\phi} \sin(\theta/2) |1\rangle = \cos(\theta/2) |0\rangle + (\cos(\phi) + i \sin(\phi)) \sin(\theta/2) |1\rangle \quad (7)$$

Where  $0 \leq \theta \leq \pi$  and  $0 \leq \phi < 2\pi$ . Note that in this context  $\phi$  and  $\theta$  may be interpreted as spherical coordinates as can be seen in figure 12.

**The Density matrix** is another important concept we need to introduce to properly understand GST. It is a matrix that describes a quantum state, and it is vastly more powerful than the state vector notation in the sense that it can describe a "non-pure" quantum state known as a mixed state. Mixed states generally arise in three different situations. Firstly when the preparation of the system is not fully known. Secondly, when one wants to describe an entangled system, which will not be the case for this project as one needs more than a single qubit for entanglement. Finally, when noise or decoherence transforms a pure state into a mixed state.

The density operator is mathematically defined as:

$$\rho = \sum_j p_j |\psi_j\rangle \langle \psi_j| \quad (8)$$

Where  $p_j$  is the probability of quantum state  $|\psi_j\rangle$ . The density matrix must conform to a few requirements, it must be Hermitian, have a trace of 1 and must be positive and semi-definite.

For example if we reuse the quantum state of our last example, our density matrix would be:

$$\rho = (\frac{3}{5}i |0\rangle + \frac{4}{5} |1\rangle)(\frac{-3}{5}i \langle 0| + \frac{4}{5} \langle 1|) = \begin{pmatrix} \frac{3}{5}i \\ \frac{4}{5} \end{pmatrix} \begin{pmatrix} \frac{3}{5}i & \frac{4}{5} \end{pmatrix} = \begin{pmatrix} \frac{9}{25} & \frac{12i}{25} \\ \frac{-12i}{25} & \frac{16}{25} \end{pmatrix} \quad (9)$$

Notice that the diagonal elements are the same as our probabilities. The off-diagonal elements of a density matrix signify our superposition. This means that our state is not a classical statistical mix of two different states, but an actual quantum physical superposition. The difference is that a quantum superposition may exhibit quantum interference, proving that a quantum state is not in a state we don't know, but truly in multiple states **at once**. The implications of this cannot be stressed enough but are sadly outside of the scope of this paper.

Another convenient way to notate the density matrix is by using the Pauli matrices. These are defined as:  $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ ,  $\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ ,  $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$  These are Hermitian, and together with the Identity matrix  $I$  they form a basis for the real vector space of 2X2 Hermitian matrices. Therefore any Hermitian 2X2 matrices can be described as a linear combination of these matrices, which our density operator just happens to be.



## A.2 The Quantum Gate

In this subsection, we will introduce the quantum gate. We will first explain a quantum gate in the context of pure states. We will then show how the quantum gate can be visualised on the Bloch sphere. Finally, we will introduce the superoperator, which acts as a quantum gate for mixed states.

**The quantum gate** is the basic building block of quantum circuits and by extension quantum algorithms. In classical computer science, it finds its analogy in the logic gate. Note that the only one-bit logic gate possible in classical computer science is the NOT gate. In quantum computer science things are not quite as simple. In the previous section we learned that a pure quantum state can be represented by  $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$ , in which  $\theta$  and  $\phi$  are real numbers.

A quantum gate can then be seen as an entity that acts upon a quantum state  $|\psi\rangle$  and returns a quantum state  $|\phi\rangle$ . Quantum gates for pure states are defined as unitary operations, this has a consequence that they are reversible [39]. Just as a quantum state is usually defined by a state vector a quantum gate is usually defined by a unitary matrix. That is  $|\phi\rangle = G|\psi\rangle$ , where  $G$  is an  $2 \times 2$  unitary Hermetian matrix.

**The Bloch sphere** also gives an intuitive way to visualise a quantum gate. Namely, if a quantum state can be seen as a position on the surface of the Bloch sphere, then a quantum gate is simply a rotation on that surface. This view allows us to define the most general one qubit quantum gates; the parameterised rotation gates [37]. We will define these as  $R_x(\theta)$ ,  $R_y(\theta)$  and  $R_z(\theta)$ , which rotate a quantum state with  $\theta$  radians around the x, y and z axis respectively.

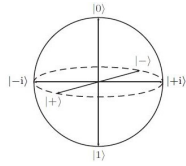


Figure 13:  
Bloch sphere:  
six basic states.

Commonly a quantum processor will have a specific gate set which it can execute. The gate set which we will encounter within this report are: X gate ( $R_x(\pi)$ ), Y gate ( $R_y(\pi)$ ),  $X^{\frac{1}{2}}$  gate ( $R_x(\frac{1}{2}\pi)$ ),  $Y^{\frac{1}{2}}$  gate ( $R_y(\frac{1}{2}\pi)$ ) and the Identity gate I ( $R_x(0) = R_y(0) = R_z(0)$ ).

Note that with this gate set it is only possible to visit the  $|0\rangle$ ,  $|1\rangle$ ,  $|+\rangle$ ,  $|-\rangle$ ,  $|+i\rangle$  and  $|-i\rangle$  states, which can be seen in figure 13. This is satisfactory for this project as the aim of GST is to characterise a specific gate set, and not to execute complicated quantum algorithms.

**Superoperators** are the matrices which act upon a superket. That is  $|\rho'\rangle = G|\rho\rangle$ . To be considered a physically valid superoperator it should follow two constraints.

Firstly, it should preserve the trace of the density matrix. As we saw in the previous section the trace of the density matrix encodes the measurement probabilities, and  $\text{tr}(\rho) \neq 1$  is considered nonphysical (like before, there is a 100% you measure "something"). The second constraint is that the density matrix should be completely positive, as negative probabilities make no sense physically. Together we call these constraints completely positive and trace-preserving (CPTP).

## A.3 The Quantum Circuit

**The quantum circuit** is the quantum counterpart of the classical circuit. The classical circuit uses classical bits and logic gates to define a sequence of operations. The quantum circuit does the same, albeit with qubits and quantum gates. Just like classical circuits, quantum circuits come in many different flavours, varying enormously in size and complexity.

To give a few examples see figure 14 in which some basic circuits are shown with the gate set available to us. For clarity's sake, there are intermediate renderings of the Bloch sphere to give an idea of what happens with a quantum state when a quantum gate is applied.

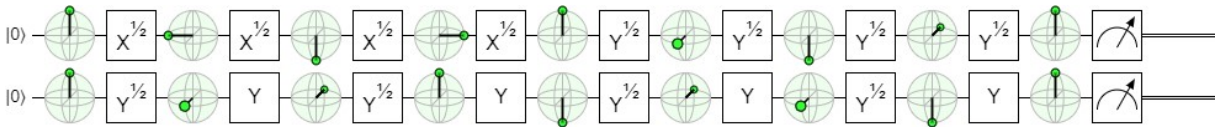


Figure 14: A simple quantum circuit made with algassert.com/quirk.

Note that a normal quantum circuit would not exhibit these intermediate Bloch sphere renderings, and would exhibit a bigger variety in gates used. The block at the end stands for the measuring operation, and it transforms the quantum data into classical data by destroying the superposition and forcing it into a classical 0 or 1 state.



Another point of interest is that this circuit only uses one qubit gates (as this project focuses on one qubit systems). However, the true magic of quantum computer science exhibits itself when two-qubit gates are used leading to phenomena such as entanglement [37]. We will also define a "layer" of a quantum circuit, which is the set of gates which are executed simultaneously. For example, in figure 14 the first layer is  $(X^{\frac{1}{2}}, Y^{\frac{1}{2}})$ , the second is  $(X^{\frac{1}{2}}, Y)$ , etc..

## B GSTBlochWidget Additional Data

### B.1 UML Diagram

GSTBlochWidget
labels : List[String] gates : List[Matrix] target_gates : List[Matrix] file_path : String rho : Matrix = zero_state target_rho : Matrix = zero_state gate_count : Int = 0 fidelity : Float = 0 bloch_coord : List[Float] = [0, 0, 1] history : List[String] = []
render_widget () update_bloch_coord () update_fidelity () evolve (rho, superOp): new_rho save_circuit ()

### B.2 Example Code

```
import GSTBlochWidget as bw
import pygsti

results = pygsti.io.read_results_from_dir(result_dir_path)
gi_res = results.estimated['GateSetTomography'].models['final iteration estimate'].
__getitem__(('Gi', 0))
gxpi_res = results.estimated['GateSetTomography'].models['final iteration estimate'].
__getitem__(('Gxpi', 0))
gyip_res = results.estimated['GateSetTomography'].models['final iteration estimate'].
__getitem__(('Gypi', 0))

widget = GSTBlochWidget(['I', 'X', 'Y'], [gi_res, gxpi_res, gyip_res], [bw.gi, bw.gxpi, bw.gypi],
"path_to_dir")
widget.render_widget()
```

## C Tables of Results

### C.1 Error Generators and Diamond Norm

Compared model: 3_gate_exp_init_state=p1				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	99.4%	0.4%	0.2%	0.005449
$X^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.018568
$Y^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.025465
Target model: 3_gate_sim_init_fid=0.95				
I	99.8%	0.1%	0.1%	0.00876
$X^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.018325
$Y^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.023264

(a)

Compared model: 3_gate_exp_init_state=m1				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	96.4%	2.9%	0.6%	0.006327
$X^{\frac{1}{2}}$	99.9 %	0.1%	0%	0.034369
$Y^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.050599
Target model: 3_gate_sim_init_fid=0.95				
I	98.5%	1.2%	0.3%	0.009337
$X^{\frac{1}{2}}$	99.9 %	0.1%	0%	0.033937
$Y^{\frac{1}{2}}$	99.9 %	0.1 %	0%	0.048303

(b)

Compared model: 3_gate_exp_init_state=0				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	27.9%	60.3%	11.8%	0.006768
$X^{\frac{1}{2}}$	100 %	0%	0%	0.029692
$Y^{\frac{1}{2}}$	0.3 %	99.7 %	0%	0.506936
Target model: 3_gate_sim_init_fid=0.95				
I	75.3%	20.5%	4.2%	0.009362
$X^{\frac{1}{2}}$	100 %	0%	0%	0.029191
$Y^{\frac{1}{2}}$	0.3 %	99.7 %	0%	0.506876

(c)

Compared model: 3_gate_exp_init_state=mixed				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	93.7%	4.4%	1.8%	0.002326
$X^{\frac{1}{2}}$	9.3 %	76.4%	14.3%	0.00205
$Y^{\frac{1}{2}}$	61.8 %	34.3 %	3.9%	0.00287
Target model: 3_gate_sim_init_fid=0				
I	99.2%	0.5%	0.3%	0.00551
$X^{\frac{1}{2}}$	81.7 %	15%	3.3%	0.003864
$Y^{\frac{1}{2}}$	60.3 %	35.1 %	4.6%	0.002951

(d)

Compared model: 5_gate_exp_init_state=0				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	99.3%	0.5%	0.2%	0.005164
$X^{\frac{1}{2}}$	99.9 %	0.1%	0%	0.023881
$Y^{\frac{1}{2}}$	99.8 %	0.1 %	0%	0.020028
X	99.9 %	0.1 %	0%	0.037754
Y	99.9 %	0.1%	0%	0.040837
Target model: 5_gate_sim_init_fid=0.95				
I	97.4%	0.5%	2.1%	0.005278
$X^{\frac{1}{2}}$	99.9 %	0%	0%	0.02841
$Y^{\frac{1}{2}}$	99.9 %	0.1 %	0.1%	0.022073
X	99.9 %	0%	0.1%	0.034841
Y	99.8 %	0 %	0.1%	0.035489

(e)

Compared model: 5_gate_exp_init_state=mixed				
Gate	$\mathbb{H}$	$\mathbb{S}$	$\mathbb{A} + \mathbb{C}$	D norm
Target model: ideal_model				
I	90.2%	9.5%	0.3%	0.17976
$X^{\frac{1}{2}}$	95.6 %	3.1%	1.3%	0.261779
$Y^{\frac{1}{2}}$	94.5 %	3.7 %	1.8%	0.273567
X	98 %	1.5%	0.5%	0.123189
Y	96.6 %	2.7 %	0.7%	0.100384
Target model: 5_gate_sim_init_fid=0.95				
I	96.7%	2.8%	0.5%	0.184688
$X^{\frac{1}{2}}$	95.2 %	1.1%	3.7%	0.315419
$Y^{\frac{1}{2}}$	81.2 %	1.9 %	16.9%	0.199483
X	99.2 %	0.2%	0.6%	0.213397
Y	97.5 %	2.1 %	0.4%	0.095017

(f)

Table 3: The first multi-column row specifies which experimental model is used, the second multi-column row tells us that we are now using the ideal\_model as our target, and the third multi-column row tells us that we are now using a simulated model as our target. Furthermore, the first column specifies what gate is considered, while the following columns specify how much of the error generator is captured by what subspace. The last column gives the diamond norm between the specified gates.

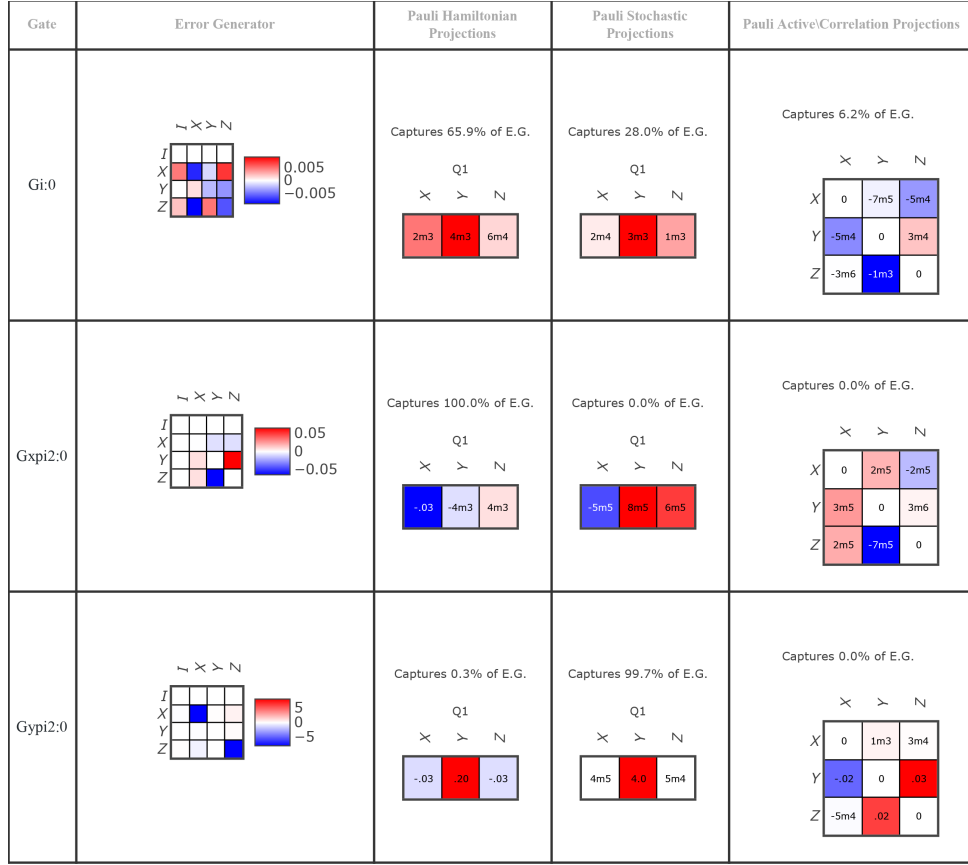


Figure 15: Error generator of 3 gate model with `init_state=0`. First column shows the error generator in the form of a heat-map. The following columns project the error generator into the  $\mathbb{H}$ ,  $\mathbb{S}$  and  $\mathbb{A} + \mathbb{C}$  subspaces. They also report what proportion of the error generator has been captured within these subspaces.

## C.2 $N_\sigma$ metric

$N_\sigma$ of 3 gate model with simulated data		
init_fid or con-straint	CPTP	TP
init_fid=0	-12.68	-12.74
init_fid=0.95	-17.09	-17.21
init_fid=1	-17.57	-17.82

(a)

$N_\sigma$ of 3 gate model with experimental data		
init_state or con-straint	CPTP	TP
init_state=0	2349.76	9.49
init_state=mixed	17.38	16.86
init_state=m1	14.39	13.63
init_state=p1	10.32	9.99

(b)

$N_\sigma$ of 5 gate model with simulated data		
init_fid or con-straint	CPTP	TP
init_fid=0	549.63	480.22
init_fid=0.95	-7.41	-8.15
init_fid=1	-19.06	-19.30

(c)

$N_\sigma$ of 5 gate model with experimental data		
init_state or con-straint	CPTP	TP
init_state=mixed	511.90	461.18
init_state=0	14.13	12.19

(d)

Table 4:  $N_\sigma$  metric of analysed models. The multicolumn row specifies the number of gates of the model in question, and if it is a simulated model or an experimental one. The first column specifies the differing factor between the models, which is initialisation fidelity for the simulated models, and initialisation state for the experimental models. The second column shows the  $N_\sigma$  of CPTP runs, while the third column shows the  $N_\sigma$  of TP runs.

## References

- [1] Paul Benioff. “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines”. In: *Journal of Statistical Physics* 22.5 (May 1980), pp. 563–591. DOI: 10.1007/bf01011339. URL: <https://doi.org/10.1007/bf01011339>.
- [2] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6–7 (June 1982), pp. 467–488. DOI: 10.1007/bf02650179. URL: <https://doi.org/10.1007/bf02650179>.
- [3] IU. I. Manin. *Vychislimoe i nevychislimoe / IU.I. Manin*. Russian. "Sov. radio," Moskva, 1980, 127 p.
- [4] Richard P. Feynman. “Quantum mechanical computers”. In: *Foundations of Physics* 16.6 (June 1986), pp. 507–531. DOI: 10.1007/bf01886518. URL: <https://doi.org/10.1007/bf01886518>.
- [5] P.W. Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc. Press, 1994. DOI: 10.1109/sfcs.1994.365700. URL: <https://doi.org/10.1109/sfcs.1994.365700>.
- [6] Isaac L. Chuang, Neil Gershenfeld, and Mark Kubinec. “Experimental Implementation of Fast Quantum Searching”. In: *Phys. Rev. Lett.* 80 (15 Apr. 1998), pp. 3408–3411. DOI: 10.1103/PhysRevLett.80.3408. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.80.3408>.
- [7] Juan Yin et al. “Satellite-based entanglement distribution over 1200 kilometers”. In: *Science* 356.6343 (June 2017), pp. 1140–1144. DOI: 10.1126/science.aan3211. URL: <https://doi.org/10.1126/science.aan3211>.
- [8] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5. URL: <https://doi.org/10.1038/s41586-019-1666-5>.
- [9] *IBM unveils breakthrough 127-qubit quantum processor*. URL: <https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor>.
- [10] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. “An introduction to quantum machine learning”. In: *Contemporary Physics* 56.2 (Oct. 2014), pp. 172–185. DOI: 10.1080/00107514.2014.964942. URL: <https://doi.org/10.1080/00107514.2014.964942>.
- [11] Emmanuel E Haven. “A discussion on embedding the Black–Scholes option pricing model in a quantum physics setting”. In: *Physica A: Statistical Mechanics and its Applications* 304.3–4 (Feb. 2002), pp. 507–524. DOI: 10.1016/s0378-4371(01)00568-4. URL: [https://doi.org/10.1016/s0378-4371\(01\)00568-4](https://doi.org/10.1016/s0378-4371(01)00568-4).
- [12] Carlos Outeiral et al. “The prospects of quantum computing in computational molecular biology”. In: *WIREs Computational Molecular Science* 11.1 (May 2020). DOI: 10.1002/wcms.1481. URL: <https://doi.org/10.1002/wcms.1481>.
- [13] Alexander J. McCaskey et al. “Quantum chemistry as a benchmark for near-term quantum computers”. In: *npj Quantum Information* 5.1 (Nov. 2019). DOI: 10.1038/s41534-019-0209-0. URL: <https://doi.org/10.1038/s41534-019-0209-0>.
- [14] Nicolas Gisin et al. “Quantum cryptography”. In: *Reviews of Modern Physics* 74.1 (Mar. 2002), pp. 145–195. DOI: 10.1103/revmodphys.74.145. URL: <https://doi.org/10.1103/revmodphys.74.145>.
- [15] Tommaso Gagliardoni. *Quantum Attack Resource Estimate: Using Shor’s algorithm to break RSA vs DH/DSA vs ECC*. Aug. 2021. URL: <https://research.kudelskisecurity.com/2021/08/24/quantum-attack-resource-estimate-using-shors-algorithm-to-break-rsa-vs-dh-dsa-vs-ecc>.
- [16] M. H. Aboeih et al. “Fault-tolerant operation of a logical qubit in a diamond quantum processor”. In: *Nature* (May 2022). DOI: 10.1038/s41586-022-04819-6. URL: <https://doi.org/10.1038/s41586-022-04819-6>.
- [17] Erik Nielsen et al. “Gate Set Tomography”. In: *Quantum* 5 (Oct. 2021), p. 557. DOI: 10.22331/q-2021-10-05-557. URL: <https://doi.org/10.22331/q-2021-10-05-557>.
- [18] Roman Schmied. “Quantum state tomography of a single qubit: comparison of methods”. In: *Journal of Modern Optics* 63.18 (Feb. 2016), pp. 1744–1758. DOI: 10.1080/09500340.2016.1142018. URL: <https://doi.org/10.1080/09500340.2016.1142018>.

- [19] M. Mohseni, A. T. Rezakhani, and D. A. Lidar. “Quantum-process tomography: Resource analysis of different strategies”. In: *Physical Review A* 77.3 (Mar. 2008). DOI: 10.1103/physreva.77.032322. URL: <https://doi.org/10.1103/2Fphysreva.77.032322>.
- [20] Adam C. Keith et al. “Joint quantum-state and measurement tomography with incomplete measurements”. In: *Physical Review A* 98.4 (Oct. 2018). DOI: 10.1103/physreva.98.042318. URL: <https://doi.org/10.1103/2Fphysreva.98.042318>.
- [21] S. S. Wilks. “The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses”. In: *The Annals of Mathematical Statistics* 9.1 (Mar. 1938), pp. 60–62. DOI: 10.1214/aoms/1177732360. URL: <https://doi.org/10.1214/aoms/1177732360>.
- [22] Robin Blume-Kohout et al. “A Taxonomy of Small Markovian Errors”. In: *PRX Quantum* 3.2 (May 2022). DOI: 10.1103/prxquantum.3.020335. URL: <https://doi.org/10.1103/2Fprxquantum.3.020335>.
- [23] Erik Nielsen et al. *Python GST Implementation (PyGSTi) v. 0.9*. [Computer Software] <https://doi.org/10.11578/dc.20190722.2>. July 2019. DOI: 10.11578/dc.20190722.2. URL: <https://doi.org/10.11578/dc.20190722.2>.
- [24] M.H.M.A Abobeih. “From atomic-scale imaging to quantum fault-tolerance with spins in diamond”. In: (Jan. 2021). DOI: 10.4233/uuid:cce8dbcb-cfc2-4fa2-b78b-99c803dee02d. URL: <https://doi.org/10.4233/uuid:cce8dbcb-cfc2-4fa2-b78b-99c803dee02d>.
- [25] Alexandre M. Souza, Gonzalo A. Álvarez, and Dieter Suter. “Experimental protection of quantum gates against decoherence and control errors”. In: *Physical Review A* 86.5 (Nov. 2012). DOI: 10.1103/physreva.86.050301. URL: <https://doi.org/10.1103/physreva.86.050301>.
- [26] E. D. Herbschleb et al. “Ultra-long coherence times amongst room-temperature solid-state spins”. In: *Nature Communications* 10.1 (2019). DOI: 10.1038/s41467-019-11776-8.
- [27] Jun-Feng Wang et al. “Coherent Control of Nitrogen-Vacancy Center Spins in Silicon Carbide at Room Temperature”. In: *Physical Review Letters* 124.22 (June 2020). DOI: 10.1103/physrevlett.124.223601. URL: <https://doi.org/10.1103/2Fphysrevlett.124.223601>.
- [28] G. Breit and I. I. Rabi. “Measurement of Nuclear Spin”. In: *Physical Review* 38.11 (Dec. 1931), pp. 2082–2083. DOI: 10.1103/physrev.38.2082.2. URL: <https://doi.org/10.1103/physrev.38.2082.2>.
- [29] L J Rogers et al. “Singlet levels of the NV centre in diamond”. In: *New Journal of Physics* 17.1 (Jan. 2015), p. 013048. DOI: 10.1088/1367-2630/17/1/013048. URL: <https://doi.org/10.1088/1367-2630/17/1/013048>.
- [30] Sandia National Laboratories. *Gate set tomography: How physicists are revealing the inner workings of quantum computers*. <https://phys.org/news/2022-01-gate-tomography-physicists-revealing-quantum.html>. Accessed: 2022-04-21. 2022.
- [31] Xiao Xue et al. “Quantum logic with spin qubits crossing the surface code threshold”. In: *Nature* 601.7893 (Jan. 2022), pp. 343–347. DOI: 10.1038/s41586-021-04273-w. URL: <https://doi.org/10.1038/5C%2Fs41586-021-04273-w>.
- [32] Mateusz T. Mądzik et al. “Precision tomography of a three-qubit donor quantum processor in silicon”. In: *Nature* 601.7893 (Jan. 2022), pp. 348–353. DOI: 10.1038/s41586-021-04292-7. URL: <https://doi.org/10.1038/5C%2Fs41586-021-04292-7>.
- [33] Mohan Sarovar et al. “Detecting crosstalk errors in quantum information processors”. In: *Quantum* 4 (Sept. 2020), p. 321. DOI: 10.22331/q-2020-09-11-321. URL: <https://doi.org/10.22331/5C%2Fq-2020-09-11-321>.
- [34] Robin Blume-Kohout et al. “Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography”. In: *Nature Communications* 8.1 (Feb. 2017). DOI: 10.1038/ncomms14485. URL: <https://doi.org/10.1038/ncomms14485>.
- [35] L. M. K. Vandersypen and I. L. Chuang. “NMR techniques for quantum control and computation”. In: *Reviews of Modern Physics* 76.4 (Jan. 2005), pp. 1037–1069. DOI: 10.1103/revmodphys.76.1037. URL: <https://doi.org/10.1103/2Frevmodphys.76.1037>.
- [36] K. R. Popper. *The Logic of Scientific Discovery*. London: Hutchinson, 1934.
- [37] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CB09780511976667.

- [38] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, Aug. 2018. DOI: 10.1017/9781316995433. URL: <https://doi.org/10.1017/9781316995433>.
- [39] Dipal Shah. *Design of Regular Reversible Quantum Circuits*. Tech. rep. Jan. 2000. DOI: 10.15760/etd.129. URL: <https://doi.org/10.15760/etd.129>.