Doctorate of Philosophy

Schrödinger's Catwalk

Brian Flynn

University of Bristol

February, 2021

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LISTINGS

ACRONYMS

C	carbon-13
^{14}N	nitrogen-14
AI	artificial intelligence
AIC	Akaike information criterion
AICC	Akaike information criterion corrected
BF	Bayes factor
BFEER	Bayes factor enhanced Elo ratings
BIC	Bayesian information criterion
CLE	classical likelihood estimation
CPU	central processing unit
DAG	directed acyclic graph
EDH	experiment design heuristic
ES	exploration strategy
ET	exploration tree
FH	Fermi-Hubbard
FN	false negatives
FP	false positives
GA	genetic algorithm
GES	genetic exploration strategy
GPU	graphics processing unit
HPD	high particle density
IQLE	interactive quantum likelihood estimation
JWT	Jordan Wigner transformation
LE	Loschmidt echo

LTL log total likelihood

ML machine learning

MVEE minimum volume enclosing ellipsoid
MW microwave

NN neural network
NV nitrogen-vacancy
NVC nitrogen-vacancy centre

OF objective function

PBS portable batch system
PGH particle guess heuristic
PL photoluminescence

QC quantum computer
QHL quantum Hamiltonian learning
QL quadratic loss
QLE quantum likelihood estimation
QM quantum mechanics
QML quantum machine learning
QMLA Quantum Model Learning Agent

SMC sequential monte carlo
SVM support vector machine

TLTL total log total likelihood
TN true negatives
TP true positives

VQE variational quantum eigensolver

GLOSSARY

Q	Quantum system which is the target of Quantum Model Learning Agent, i.e. the system to be characterised
champion model	The model deemed by QMLA as the most suitable for describing the target system
chromosome	A single candidate, in the space of valid solutions to the posed problem in a genetic algorithm
expectation value	Average outcome expected by measuring an observable of a quantum system many times, **??**
gene	Individual element within a chromosome
hyperparameter	Variable within an algorithm that determines how the algorithm itself proceeds
instance	A single implementation of the QMLA algorithm, resulting in a nominated champion model
likelihood	Value that represents how likely a hypothesis is. Usually used in the context of likelihood estiamation, **??**
model	The mathematical description of some quantum system, **??**
model space	Abstract space containing all descriptions (within defined constraints such as dimension) of the system as models
probe	Input probe state, $\|\psi\rangle$, which the target system is initialised to, before unitary evolution
results directory	Directory to which the data and analysis for a given run of QMLA are stored
run	Collection of QMLA instances, usually targeting the same system with the same initial conditions

spawn Process by which new models are generated, ususally by combining previously considered models

success rate Fraction of instances within a run where QMLA nominates the true model as champion

term Individual constituent of a model, e.g. a single operator within a sum of operators, which in total describe a Hamiltonian.

volume Volume of a parameter distribution's credible region, **??**

win rate For a given candidate model, the fraction of instances within a run which nominated it as champion

Part I

EXPERIMENTAL STUDIES

1

NITROGEN VACANCY CENTRE

It is of primary interest to apply the Quantum Model Learning Agent (QMLA) algorithm to real-life, experimental systems. In this chapter we devise an exploration strategy (ES) to operate in conjunction with experimental data in order to characterise an electron spin in an nitrogen-vacancy (NV) centre in diamond. In particular, we model, through Hamiltonian terms, interactions between the spin and the spin bath in which it resides, so that QMLA is finding an effective model for the open system dynamics.

Here we will first introduce a basic picture of nitrogen-vacancy centres (NVCs), using basic but nonstandard nomenclature for simplicity; for thorough descriptions of the underlying physics, readers are referred to [1]. We next discuss the target system with respect to its modelling, determining the suitable terms which *might* represent the NVC's interactions, to inform the starting point for the QMLA. Finally we describe the implementation of an ES for the examination of the NVC, and the results of the QMLA procedure.

1.1 NITROGEN-VACANCY CENTRE

NV centers are point defects in diamond, occuring intrinsically (naturally) [2] or extrinsically (synthetically) [3, 4]. A substitutional nitrogen-14 (^{14}N) isotope is embedded in a lattice of carbon atoms in diamond, adjacent to a lattice vacancy, such that it is surrounded by three carbon-13s (Cs) (either ^{12}C or ^{13}C) [5]. Of the ^{14}N atom's five valence electrons, three bond with nearby Cs; the remaining two unbonded electrons form a lone pair and can be thought of as a single spin-$\frac{1}{2}$ particle. The adjacent lattice vacancy has three unbonded electrons, two of which bond together leaving a single unpaired electron. The single electron in the lattice vacancy, together with the effective lone pair of the ^{14}N, form a system of two spin-$\frac{1}{2}$ particles: such systems have been thoroughly studied. Of particular interest are the resultant *triplet* states, i.e. the allowed permutations of the two particles with total quantum spin $S = 1$, with magnetic spin multiplicity allowing $m_s = -1, 0, 1$, giving rise to three distinct energy levels for the system.

A *manifold* is a set of states with marginal differences, such as a single differing quanutm number. For example, states near the absolute ground state might differ only in their magnetic spin quantum number: together they can be characterised as the *ground state manifold*. We consider two principle manifolds of the system: the ground state and excited manifolds, each consisting of three states, corresponding to the allowed values for magnetic spin m_s, see Fig. 1.1a. For brevity, we denote states with reference to their magnetic spin and manifold, e.g. the state in the ground state manifold with $m_s = 0$ is denoted $|m_s = 0\rangle_g$. In the absence of a magnetic field, the states corresponding to $|m_s = \pm 1\rangle$ are degenerate, but in the presence of a magnetic field, B, they have distinct energy levels, referred to as the Zeeman effect, Fig. 1.1b.

For the purposes of computation, we choose the ground state and one of the excited states as the two levels of a qubit. We designate the states $|m_s = 0\rangle_g$ and $|m_s = -1\rangle_g$ as the computational basis states $|0\rangle, |1\rangle$ respectively, such that we have defined a qubit and computational basis, Fig. 1.1d. We also require a reliable mechanism by which we can be confident that our qubit is in a definite state, to serve as the starting point of computation: usually qubits are initialised to $|0\rangle$, so here we aim to prepare the NVC in $|m_s = 0\rangle_g$. By shining a laser of 532 nm (green) on the NVC, irrespective of which state within the ground state manifold the spin starts, it is excited in to the excited manifold, from which it decays back to the ground state manifold. The process of this decay can be exploited for the preparation of the NVC in $|m_s = 0\rangle_g$ and therefore enable initialisation for computation. That is, when the NVC is excited to the $|m_s = 0\rangle_e$ level, the dominant decay process is spin-preserving, so after decay it ends in $|m_s = 0\rangle_g$. On the other hand, if the NVC had been excited instead to $|m_s = \pm 1\rangle_e$, the dominant decay process is through a meta-stable/shelving state, which does not preserve spin, so in this case it also ultimately decays to the $|m_s = 0\rangle_g$. Therefore, irrespective of the initial state, by shining the green laser on the NVC and exciting it into any of the states in the excited manifold, after decay it is most likely that it has been prepared in $|m_s = 0\rangle_g = |0\rangle$, providing us a starting point from which to perform computation.

The difference in energy between our defined computational basis states $|0\rangle$ and $|1\rangle$ is ~ 2.87GHz, i.e. it is addressible by microwave (MW) radiation. Via antenna, we can deliver a MW pulse upon the NVC, driving the NVC between the two levels providing an implementation of an X-gate. Likewise, having initialised the state to $|0\rangle$, we can perform a $\pi/2$ rotation about the logical z-axis, by running the MW laser for half the time, resulting in the state $|+\rangle$. We can similarly devise MW radiation to achieve quantum gates and operations on our NVC qubit. We depict these cycles in Fig. 1.1c.

We can further exploit the decay mechanism to compose a readout procedure, to infer the population of $\{|0\rangle, |1\rangle\}$ at a given instant, for example following the application of a series of gates (a circuit) to the system. We know that the excitation due to the green laser is spin-preserving, i.e. when the NVC has been excited to $|m_s = 0\rangle_e$, it had originated in $|m_s = 0\rangle_g$. We also know that the decay $|m_s = 0\rangle_e \to |m_s = 0\rangle_g$ is spin preserving, with the emission of a red photon: by simply counting the number of excess[1] photons emitted, we quantify the population of $|0\rangle$ at the time of query. On the contrary, when the $|m_s = -1\rangle_g$ is excited, spin is also preserved, so it goes to $|m_s = -1\rangle_e$, but $|m_s = -1\rangle_e$ decays through the shelving state as outlined earlier, *without* the emission of a red photon (the decay gives out infrared instead). We can hence infer the population of $|m_s = -1\rangle_g$ at the time of query by the fraction of incidents which don't emit a photon [6]. That is, say we first calibrate the system by retaining the green laser for some time: after a few μs, a steady state is achieved where the majority of the time, the triplet is in the computional state $|0\rangle = |m_s = 0\rangle_g$. Then, excitation from the same laser results in the excitation to $|m_s = 0\rangle_e$, which decays back to $|m_s = 0\rangle_g$ and emits a photon in the process; by counting the red photons emitted in a certain time window – equivalently, measuring the photoluminescence (PL) signal – we benchmark the population of $|0\rangle$ when nothing else has

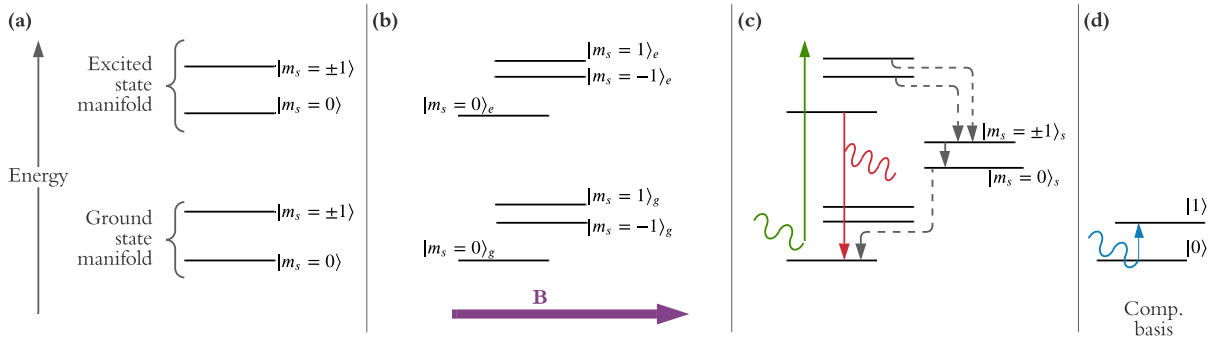


Figure 1.1: Simplified depiction of energy levels of the nitrogen-vacancy centre, corresponding to its triplet state. **a**, With no external magnetic field, the system has excited and ground-state manifolds, each of which consist of two energy levels depending on the magnetic spin, m_s. **b**, In the presence of a magnetic field (purple, B), the magnetic spins have distinct energy levels, i.e. Zeeman splitting giving distinct m_s. States are denoted by their magnetic spin, m_s and subscripted by their manifold (e for excited and g for ground-state). **c**, Application of a 532 nm laser (green arrow) excites the nitrogen-vacancy centre from any of the states in the ground state manifold into the excited manifold. The dominant decay mechanism for the excited states are shown: (i) $|m_s = 0\rangle_e \rightarrow |m_s = 0\rangle_g$ (photoluminescence, red) through the emission of a photon at 637 nm); (ii) $|m_s = \pm 1\rangle_e \rightarrow |m_s = 0\rangle_g$ (dotted grey lines) via the shelving manifold which allows for non-spin-preserving transition, emitting a photon in the infrared (not shown). **d**, Computational basis states $|0\rangle$ and $|1\rangle$ are assigned to the two lowest energy states. The difference in energy between these states is such that a microwave (MW, blue) can drive transition from $|0\rangle \leftrightarrow |1\rangle$. MW pulses can also be used to achieve other states apart from the basis states, allowing for the implementation of quantum logic gates.

happened as p_0. Now, when we apply gates (i.e. MW pulses) to the NVC, we can similarly read out the population of $|0\rangle$ as p_0', and infer that the likelihood that the NVC is found in the initial state $|0\rangle$ is p_o'/p_0. We can use this quantity as the likelihood within quantum likelihood estimation (QLE), allowing us to learn from the NVC, as we will discuss in the next sections.

In summary then, by assigning basis states $|0\rangle, |1\rangle$ to energy levels of the ground state manifold, we are able to ensure the preparation of the NVC in $|0\rangle$ by first shining a green laser on the NVC. We can then apply MW radiation to achieve quantum logical gates on the system, and read out the final state of the system, again by shining a green laser and observing the

1 A large number of photons are emitted by the NVC when it is excited by a 532 nm laser, which can be profiled by its emission spectrum. At the *zero phonon line* (637 nm), a relatively large number of photons are emitted, compared with nearby wavelengths. This is where decay from the excited to ground state occurs without interacting with proximal phonons, as is the case during the spin-preserving decay $|m_s = 0\rangle_e \rightarrow |m_s = 0\rangle_g$. The excess photons are taken as indication that the electron had been in the state $|m_s = 0\rangle_e$ immediately prior to emission.

photoluminescence (i.e. the emitted photons), and inferring the population level of each basis state. We represent these concepts in a simplified format in Fig. 1.1.

1.2 TARGET SYSTEM

We take the axis of the NVC, i.e. the axis connecting the ^{14}N with the lattice vacancy, as the z-axis. While the NVC is subject to myriad interactions which result in decoherence, we choose to focus on its dominant interactions with proximal environmental nuclei. These interactions are characterised by hyperfine terms [7]. The overall Hamiltonian for such systems, where the set of nuclear sites is $\{\chi\}$, is given by

$$\hat{H}_{\text{full}} = \Delta_{\text{gs}}\hat{S}_z^2 + \mu_B g\mathbf{B}\cdot\mathbf{S} + \mathbf{S}\cdot\sum_\chi\left(\mathbf{A}_\chi\cdot\hat{I}_\chi\right) + P\hat{I}_z^2 + \mu_n g\mathbf{B}\cdot\sum_\chi\hat{I}_\chi. \tag{1.1}$$

Our overarching intention is to design an approximate model $\hat{H}'$, i.e. a subset of the terms in $\hat{H}_{\text{full}}$ which can explain the observed dynamics and decoherence of the NVC. It is therefore prudent only to retain terms which *may* contribute to the spin's decoherence. First, we will describe each term in Eq. (1.1), as well as approximations which enable us to drastically reduce the space of terms to consider for inclusion in $\hat{H}'$.

ISOLATED-SPIN TERMS describe the spin independent of the environmental nuclei

0.1. $\Delta_{\text{gs}}\hat{S}_z^2$: the *zero-field* splitting, or ground state splitting between the computational basis states. $\Delta_{\text{gs}} \sim$ GHz is a constant offset which does not contribute to the decoherence, so it is excluded from our study.

0.2. $\mu_B g\mathbf{B}\cdot\mathbf{S}$: the spin's precession about the magnetic field, $\mathbf{B} = \begin{pmatrix}B_x & B_y & B_z\end{pmatrix}$, via the total spin operator[2] $\mathbf{S} = \begin{pmatrix}\hat{S}_x & \hat{S}_y & \hat{S}_z\end{pmatrix}$, where μ_B is the Bohr magneton and g is the electron g-factor (≈ 2, simplified from the g-factor tensor).

HYPERFINE TERMS

0.3. $\hat{S}\cdot\sum_\chi\left(\mathbf{A}_\chi\cdot\hat{I}_\chi\right)$: The NVC total spin operator $\mathbf{S}$ couples the spin with each site, χ. At each site there is a nucleus which has total spin operator $\mathbf{I}_\chi = \begin{pmatrix}\hat{I}_x & \hat{I}_y & \hat{I}_z\end{pmatrix}_\chi$. $\mathbf{A}$ is the hyperfine tensor, containing the hyperfine parameters of interest. The coupling between the NVC and these nuceli is one of the primary decoherence mechanisms, so is essential to any model aiming to capture those dynamics.

BATH-ONLY TERMS describe the other nuceli independent of the spin

0.4. $P\hat{I}_z^2$: the quadrupole splitting, which provides another constant shift, and is therefore not of interest when modelling the spin's decoherence, and can be neglected.

0.5. $\mu_n g\mathbf{B}\cdot\sum_\chi\hat{I}_\chi$: μ_n is the nuclear magneton and g is the nuclear g-factor (again from the g-factor tensor).

Given that we are interested in the spin and its interactions with the environment only, we can immediately drop the bath-only terms, by assuming the bath is static apart from its interactions with the NVC. This is a usual assumption in the treatment of open system dynamics, to allow for focus on the dominant interactions for the system of interest [8]. Additionally, since the zero field splitting contributes a constant shift in energy, we can safely omit it by moving to the rotating frame. We are then left only with the second and third terms of Eq. (1.1), from which to define the space of terms in which QMLA will search:

$$\mu_B g \mathbf{B} \cdot \mathbf{S}; \tag{1.2a}$$

$$\mathbf{S} \cdot \sum_{\chi} \left(\mathbf{A}_\chi \cdot \hat{I}_\chi \right). \tag{1.2b}$$

1.2.1 *Mapping to model terms*

Next we will focus on mapping the remaining terms to operators to compose the set of terms $\mathcal{T}$ to use in our ES. In our modelling, the NVC spin is represented by the first logical qubit, with a further $|\{\chi\}|$ qubits, each representing a unique nuclear site, as discussed later in this section. As standard, we take the axis[3] of the NVC as parallel to the qubit's z-axis.

The first terms included, Eq. (1.2a), come from the spin's precession about the magnetic field. It is usually assumed that the external, applied magnetic field is well-aligned with the spin qubit's z-axis: if the field is misaligned, it leads to decoherence effects. Determining the alignment is treated as a core role of QMLA, i.e. we will endeavour to establish whether the x-, y-axis components of the magnetic field are important for describing the spin's decoherence. Then, we have

$$\mu_B g \mathbf{B} \cdot \mathbf{S} = \mu_B g (B_x \quad B_y \quad B_z) \cdot (\hat{S}_x \quad \hat{S}_y \quad \hat{S}_z) \rightarrow \alpha_x \hat{S}_x + \alpha_y \hat{S}_y + \alpha_z \hat{S}_z, \tag{1.3}$$

with $\alpha_i = \mu_B g B_i$. The spin's rotation terms to be included in QMLA's deliberations are therefore

$$\mathcal{T}_s = \{\hat{S}_x, \quad \hat{S}_y, \quad \hat{S}_z\}. \tag{1.4}$$

Next, we consider the hyperfine coupling term. In general we sum over the nuclear sites $\{\chi\}$, since the NVC spin will interact with every nucleus within a certain range. We show in [9] that a realistic system requires modelling a finite-size bath of $|\{\chi\}| \sim 15$ nuclei to capture the dynamics of interest, which is infeasible for complete characterisation via classical simulation,

2 We invoke an inexact representation of high dimensional tensors here for ease of interpretation. For instance, the total nuclear spin operator exists in arbitrary dimension (depending on the number of sites modelled), but we present it simply as $\mathbf{I} = (\hat{I}_x \ \hat{I}_y \ \hat{I}_z)$ at each site to convey that we can separate the terms in the construction of models.

3 The quantisation axis, i.e. the axis along the ^{14}N and lattice vacancy.

where we are limited to ~ 11 qubit calculations[4]. Instead, by focusing only on the *short-time* dynamics of the NVC, we can isolate the effects of dominant interactions, most notably with a single nearby C. Indeed, by assigning a first qubit as representing the NVC spin, we can map the entire environment onto a generic second *environmental qubit*, representing the amalgamation of said interactions, though we can think of the two-qubit system as the NVC coupled with a single ^{14}N [7].

$$\mathbf{S} \cdot \sum_{\chi} (\mathbf{A}_{\chi} \cdot \mathbf{I}_{\chi}) \rightarrow \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{I} \tag{1.5}$$

This reduces the dimension of our approximation: the number of qubits required, n_q reduces from $n_q = 1 + |\{\chi\}|$ to $n_q = 2$, since now we only retain qubits for the NVC and the ^{14}N (which also represents the entire bath). The hyperfine tensor $\mathbf{A}$ consists of the hyperfine parameters, i.e. the strength of correspdoning interactions.

$$\mathbf{A} = \begin{pmatrix} A_{\perp} & 0 & 0 \\ 0 & A_{\perp} & 0 \\ 0 & 0 & A_{\parallel} \end{pmatrix}, \tag{1.6}$$

where $A_{\perp}$ is the non-axial hyperfine coupling term and $A_{\parallel}$ is the axial coupling term, since the axis of the NVC is used to define the z-axis for our qubits.

The total spin operators are then those of the NVC operating on the first logical qubit, e.g. $\hat{S}_x^{(1)}$, and those of the environmental qubit on the second, e.g. $\hat{I}_x^{(2)}$. They can be summarised as

$$\begin{aligned} \mathbf{S} &= (\hat{S}_x^{(1)} \quad \hat{S}_y^{(1)} \quad \hat{S}_z^{(1)}) \\ \mathbf{I} &= (\hat{I}_x^{(2)} \quad \hat{I}_y^{(2)} \quad \hat{I}_z^{(2)}) \end{aligned} \tag{1.7}$$

So we can write,

$$\begin{aligned} \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{I} =& A_{\perp} \hat{S}_x \hat{I}_x + A_{\perp} \hat{S}_y \hat{I}_y + A_{\parallel} \hat{S}_y \hat{I}_y \\ &+ A_{xy} \left(\hat{S}_x \hat{I}_y + \hat{S}_y \hat{I}_x \right) \\ &+ A_{xz} \left(\hat{S}_x \hat{I}_z + \hat{S}_z \hat{I}_x \right) \\ &+ A_{yz} \left(\hat{S}_y \hat{I}_z + \hat{S}_z \hat{I}_y \right) \end{aligned} \tag{1.8}$$

Similarly to α_i in Eq. (1.3), we replace the expected (and theoretically computable) scalar parameters, e.g. $A_{\perp}$, with generic parameters α, to be learned. Off-diagonal terms, referred to hereafter as *transverse* terms ($\hat{S}_i \hat{I}_j$ where $i \neq j$), are usually neglected [10]. Here we will employ QMLA to determine whether the transverse contributions are worthy of inclusion in the decoherence model, although we consider only $\{\hat{S}_x \hat{I}_y, \hat{S}_x \hat{I}_z, \hat{S}_y \hat{I}_z\}$ for brevity. The hyperfine terms to be entertained by QMLA are then

$$\mathcal{T}_{HF} = \begin{Bmatrix} \hat{S}_x \hat{I}_x, & \hat{S}_y \hat{I}_y, & \hat{S}_z \hat{I}_z, \\ \hat{S}_x \hat{I}_y, & \hat{S}_x \hat{I}_z, & \hat{S}_y \hat{I}_z \end{Bmatrix}. \tag{1.9}$$

Finally, combining Eq. (1.4) and Eq. (1.9), we have the full set of terms to incorporate into the ES for the QMLA model search:

$$\mathcal{T}_{NV} = \begin{Bmatrix} \hat{S}_x, & \hat{S}_y, & \hat{S}_z, \\ \hat{S}_x\hat{I}_x, & \hat{S}_y\hat{I}_y, & \hat{S}_z\hat{I}_z, \\ \hat{S}_x\hat{I}_y, & \hat{S}_x\hat{I}_z, & \hat{S}_y\hat{I}_z \end{Bmatrix}. \tag{1.10}$$

We introduce a shorthand notation to ease model representation for the remainder of this chapter. Recall that we have defined a two-qubit Hilbert space for model construction. Terms which affect only the spin act only on the first qubit, $\hat{S}_i = \hat{S}_i^{(1)} = \hat{\sigma}_i \otimes \hat{\mathbb{1}}$, where $\hat{\sigma}_i$ is the Pauli operator giving rotation about the i-axis, and $\hat{\mathbb{1}}$ is the one-qubit identity matrix. Retaining the hyperfine notation, for the expectedly-dominant diagonal terms, we denote $\hat{A}_i = \hat{S}_i^{(1)}\hat{I}_i^{(2)} = \hat{\sigma}_i \otimes \hat{\sigma}_i$. We refer to the less-dominant off-diagonal terms as *transverse* terms, $\hat{T}_{kl} = \hat{S}_k^{(1)}\hat{I}_l^2 = \hat{\sigma}_k \otimes \hat{\sigma}_l$. We can hence rewrite Eq. (1.10) as

$$\mathcal{T}_{NV} = \begin{Bmatrix} \hat{S}_x, & \hat{S}_y, & \hat{S}_z, \\ \hat{A}_x, & \hat{A}_y, & \hat{A}_z, \\ \hat{T}_{xy}, & \hat{T}_{xz}, & \hat{T}_{yz} \end{Bmatrix}. \tag{1.11}$$

We also use a succinct representation for brevity, e.g. $\hat{S}_{xy}\hat{A}_z = \hat{S}_x + \hat{S}_y + \hat{A}_z$, where parameters $\alpha_x, \alpha_y, \alpha_z$ are implicitly assumed.

1.2.2 *Prior knowledge*

QMLA will construct models using the pool of terms defined in Eq. (1.11). Recall from **??** that each model considered must be trained independently, where the purpose of model training is to optimise the parameter vector $\vec{\alpha}$ which characterises the model. For example, the model $\hat{H}_i = \hat{S}_{x,y}\hat{A}_z = \alpha_1\hat{S}_x + \alpha_2\hat{S}_y + \alpha_3\hat{S}_3$, is trained to retrieve the optimal $\vec{\alpha}' = (\alpha_1' \quad \alpha_2' \quad \alpha_3')$. Models are trained through quantum Hamiltonian learning (QHL), described in **??**, which iteratively updates a probability distribution for the associated parameters, $\Pr(\vec{\alpha})$. As such, a *prior* distribution must be drawn, from which QHL begins its training. While QHL can redraw the probability distribution iteratively, and even find parameters entirely outside of the initial range, it is necessary at least to identify the order of magnitude where the true parameter should be found. The algorithm therefore demands that the user specifies the *range* of each parameter in which to search, which can be based on domain knowledge and theoretical predictions. For example, recall from Section 1.2 that the zero field splitting, Δ_{gs} in Eq. (1.1) (and excluded in our modelling), is expected to be $\sim$ GHz: in order to provide a reasonable chance at learning

4 This limitation arises from the requirement to compute the total evolution of the global state, involving calculation of $e^{-i\hat{H}t}$, i.e. the characterisation of an n_q-qubit model depends on classical exponentiation of the $2^{n_q} \times 2^{n_q}$ Hamiltonian for each particle and experiment in classical likelihood estimation (CLE), which is a prohibitive expense.

the true parameter, here we would propose a prior distribution of 5 ± 2GHz. We must similarly identify the rough range in which we reasonably expect to find parameters associated with each term in Eq. (1.11).

The spin-only terms, $\hat{S}_i$, are consequences of the magnetic field, expected in the range $\sim 2 - 3$MHz. Likewise, the hyperfine terms, $\hat{A}_i$ are expected in the range of $\sim$ MHz [11], while in the *secular approximation* only the z-component is expected to contribute substantially [12]. The non axial hyperfine terms, i.e. the transverse terms $\hat{T}_{kl}$ are not usually included in effective models, but can be found of order $\mathcal{O}(10\text{kHz})$ [13]. We utilise this prior understanding of the system to inform the parameter range used for training candidate models: for each of the terms in Eq. (1.11), we will adopt a normal prior distribution of 4 ± 1.5MHz. This range is sufficiently specific to ensure the training subroutine operates in a physically meaningful – and likely appropriate – space, whlie also broad enough to allow for significant differences between expectation and reality. Moverover this distribution supports hypotheses where each parameter is zero: if these prove favourable, negligible contributions can be identified and excluded from the model.

1.2.3 *Experimental procedure*

We have decided to characterise the hyperfine interactions of the NVC which dominate its decoherence process: these interactions are evident most strongly at very short timescales [14]. To isolate the effects of the hyperfine interactions, we run Hahn echo experiments, which are known to emphasise weak interactions. Hahn echo sequences attempt to decouple the spin's dynamics from the nuclear bath [10, 14, 15, 16, 17] providing a helpful platform for studying residual contributions of terms in Eq. (1.11). The NVC qubit undergoes a series of evolutions – either according to application of quantum logic gates or the natural evolution of the system interacting with its environment. We depict the stages of the experiment in Fig. 1.2, starting from the initialised computational state, $|\psi_0\rangle = |0\rangle$, through to its final state which is read out through PL, both of which as described in Section 1.1.

In particular, the final state, $|\psi\rangle_5$, is effectively read out by projection onto $|0\rangle$; we can interpret the normalised PL after evolution time t as the likelihood that the NVC is found in $|0\rangle$ after evolution of its *true* Hamiltonian, $\hat{H}_0$ for t. That is, we assign this projection as the quantity $\Pr(0|\hat{H}_0, t)$ (the likelihood), and it can be used within likelihood estimation in order to refine a candidate model $\hat{H}_j$, effectively[5] by changing the structure of $\hat{H}_j$ until $\Pr(0|\hat{H}_0, t) \approx \Pr(0|\hat{H}_j, t)\ \forall t$.

By varying the evolution time of the Hahn echo sequence, we can map the likelihood against time, which we can view as capturing the dynamics of the NVC spin Fig. 1.3. We vary time up to $t \sim 4\mu s$ in the short-time range in intervals of $\Delta t = 50ns$, so we have 425 data points. Note the data for the studied NVC is taken once and analysed offline, i.e. QMLA does not have complete authority to design experiments to run on the NVC, although it can aim to choose the most

5 Of course this is a gross simplification of QHL which is described fully in **??**

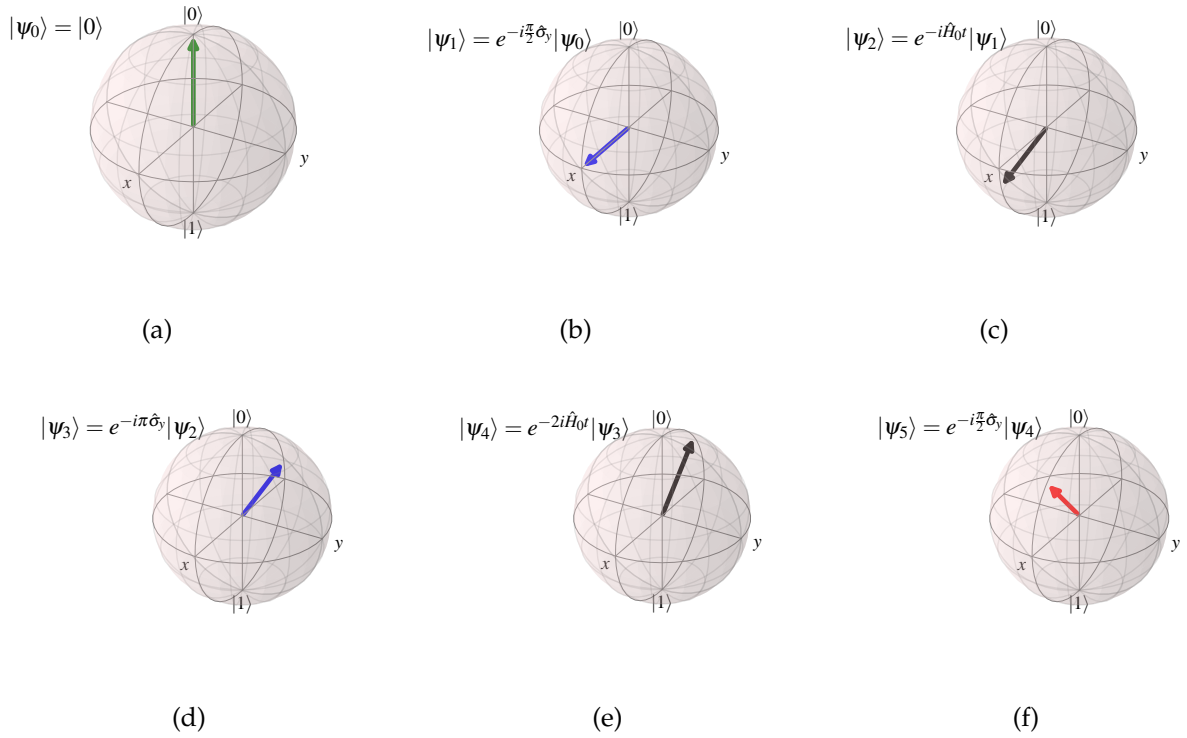


Figure 1.2: States of spin qubit at each stage of Hahn echo sequence.
(a) The state of the NVC spin is initialised by a green laser into state $|\psi_0\rangle = |0\rangle$. **(b)** We apply a rotation about the y-axis (i.e. a microwave (MW) pulse), yielding the state $|\psi_1\rangle = |+\rangle$. **(c)** The system is allowed to evolve according to its own $\hat{H}_0$ for t, $|\psi_2\rangle = e^{-i\hat{H}_0 t}|+\rangle$. **(d)** We apply a second MW pulse, this time for a π-rotation about the y-axis, $|\psi_3\rangle = e^{-i\pi\hat{\sigma}_y}e^{-i\hat{H}_0 t}|+\rangle$. **(e)** Again the system evolves according to interactions with the environment, this time for $t' = 2t$. **(f)** We apply a final MW pulse to rotate about the y-axis again, projecting it upon $|0\rangle$. Here $|\psi_5\rangle$ is roughly half way between $|0\rangle$ and $|+\rangle$, i.e. along the z-axis. The spin is read out from $|\psi_5\rangle$ via the NVC's photoluminescence. Here $\hat{H}_0 = 0.25\,\hat{\sigma}_y$ was evolved for $t = 0.5$ (arbitrary units), and the final state overlap with the initial state, i.e. the likelihood of measuring the spin in $|0\rangle$ is $\Pr(0|\hat{H}_0, t) = 0.865$.

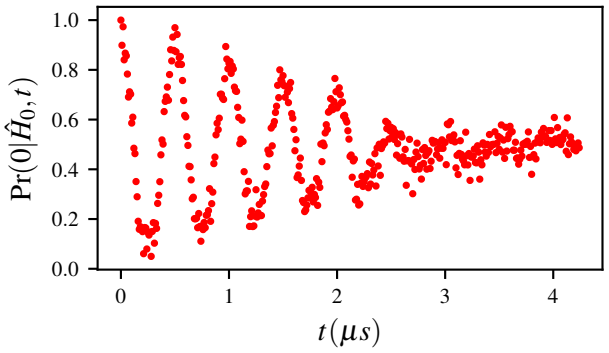


Figure 1.3: Raw data for the nitrogen-vacancy centre's dynamics. The y-axis shows the normalised photoluminescence of the NVC, equivalently the likelihood $\Pr(0|\hat{H}_0, t)$.

informative t available in the predefined set; we will discuss the consequences of this restriction later in this chapter.

1.3 Exploration strategy

We may now turn to the specific implementation details by which QMLA is applied to the study this NVC system. Recalling the terminology of QMLA from **??**, we design an exploration strategy (ES) specifically for the system under study. The ES will account for the details listed in this chapter so far, in summary:

- we aim to assign a model, $\hat{H}'$, to the NVC to describe its decoherence processes
  - we especially focus on its hyperfine interactions
- we use a 2-qubit approximation
  - the first qubit represent the spin itself
  - the second qubit represents the environment in which the NVC resides
- we query the NVC by performing Hahn echo experiments (Fig. 1.2)
- the outcome of those experiments are thought of as the system's likelihoods (Fig. 1.3)
- candidate models are composed of the terms defined in Eq. (1.11)
  - likelihoods are used for the training of individual candidate models through QHL
  - we assign approximate ranges to the scalar parameters corresponding to each term based on theoretical arguments
    - those parameters are to be learned precisely by QHL.

As outlined in **??**, the central role of any ES is to specify the model generation procedure, which QMLA relies upon for deciding the next set of candidate models to test. In this case, we

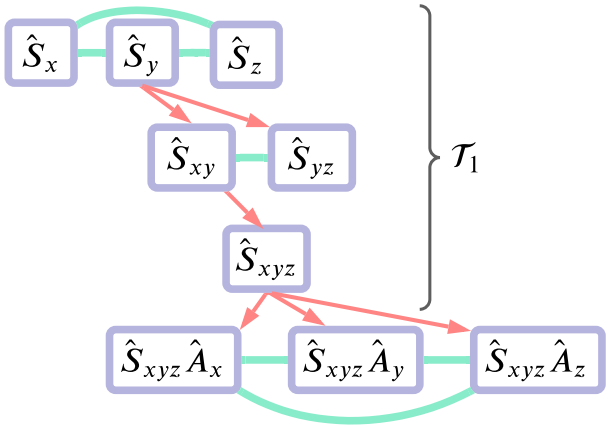


Figure 1.4: Greedy model search. Models (purple) are placed on branches, trained and consolidated (green) as in **??**, with the branch champion spawning (red) candidates to place on the subsequent branch. Branches are grouped in tiers, correspdoning to levels of approximation: the first tier of the model generation strategy is shown, where $\mathcal{T}_1 = \{\hat{S}_x, \hat{S}_y, \hat{S}_z\}$ is explored. The final champion from the first tier seeds the second tier.

exploit some intuition and prior knowledge of how such systems work, to design a bespoke model generation subroutine: we can think of this as a midway point between the completely specified ESs used for identifying the underlying lattices from a prescribed set in **??**, and the entirely general genetic algorithm which does not restrict model generation, of **??**. We use the standard structure of exploration trees (ETs) introduced in **??**, where models are placed on consecutive branches, μ, and branches are consolidated by pairwise comparisons between all models on μ, where comparisons are computed through Bayes factors (BFs). The outcome of consolidation on μ is the determination of a single *branch champion*, $\hat{H}_{C(\mu)}$.

We use a *greedy search rule*: terms are added one-by-one to gradually increase the complexity of candidate models until terms are exhausted [18]. We break the ET into three distinct tiers, each corresponding to an intuitive degree of complexity: the first tier involves the spin-only terms, $\mathcal{T}_1 = \{\hat{S}_i\}$; the second considers the hyperfine terms, $\mathcal{T}_2 = \{\hat{A}_i\}$; the final tier the transverse terms, $\mathcal{T}_3 = \{\hat{T}_i\}$. Within each tier, terms are added greedily to the previous branch's champion, $\hat{H}_{C(\mu)}$. So, the first branch is given by $\mu = \{\hat{S}_x, \hat{S}_y, \hat{S}_z\}$; say $\hat{H}_{C(\mu)} = \hat{S}_y$, then $\mu + 1$ determines that $\hat{S}_x, \hat{S}_z$ are not yet considerd, so it constructs the models $\hat{S}_{x,y}, \hat{S}_{y,z}$, e.g. Fig. 1.4. After exhausting all tiers, we consolidate the set of branch champions, $\mathbb{H}_C = \{\hat{H}_{C(\mu)}\}$, to determine the best model considered globally, $\hat{H}'$.

Clearly, this growth rule is partially deterministic, insofar as some models are gauranteed to be considered, while others are not reachable. Indeed, the space of available models are heavily constrained, in particular models in later tiers will always involve all of the tier 1 models,

e.g. $\hat{S}_x \hat{A}_y$ can not occur organically. In general, restrictions of this kind undermine the ES and are considered a weakness. To account for this, we add a final test of reducibility on the champion model, tiggered if any of the parameters of $\vec{\alpha}'$ are potentially negligible, i.e. the posterior distribution of any parameter assigns credibility to the hypothesis that the parameter is 0. This champion reducibility test simply removes the negligible-parmeter terms from $\hat{H}'$, yielding a reduced global champion, $\hat{H}'_r$. We then compute the BF between $\hat{H}', \hat{H}'_r$: if the BF indicates strong evidence in favour of the reduced version, we replace the champion model, $\hat{H}' \leftarrow \hat{H}'_r$. In effect, we thus verify the statistical significance of each term included in $\hat{H}'$.

The total model in Eq. (1.1) supports $N_t = 1 + 3 + 6|\chi| + 3 + 3|\chi| = 7 + 9|\chi|$ terms, which we reduced to a space of $N_t = 3 + 6|\chi| = 9$ through several approximations[6]. Even so, the remainng 2^9 permitted models were reduced further by building the logic of this ES from our intuition aroud existing knowledge of typical NVC systems. As such, the described ES will only ever consider < 20 models per instance. The described ES seems overly perscriptive, but should be viewed as a first attempt at a generalisable approach: essentially we can view the tiers of the greedy search as characterising the system at various approximations, e.g. the first tier examines one-qubit terms, while subsequent tiers inspect 2-qubit terms. We can envision future work where the greedy search is gradually extended to less rigid approximations, enabling study of more complex quantum systems. This leads to some important remarks:

1. Realistic, near-term applications of QMLA can not be thought of as a solution to black-box characterisation problems: it must be used in conjunction with domain expertise for the system under study.
2. While this test-case yields promising results, the outcome of QMLA here may not be especially insightful, since the available model terms were so deliberately constrained – we demonstrate a use-case in Chapter 2 where a broader scope is enabled in simulation.

A common charge against QMLA supposes to first write down the most complex model, train it fully, and then infer which terms are negligible, in a similar process to the champion reduction test outlined here. While this may be feasible in the case described here, with $N_t = 9$ and a closed term set, it is unscalable: adding just a second nuclear site increases the model search to a space of $N_t = 15$. Models of higher cardinality ($|\vec{\alpha}|$) demand higher N_e, N_p to train well, so immediately trainng the most involved model would require infeasible resources[7], and risks significantly overfitting the data. It seems more appropriate to work "from the ground up", testing terms and only keeping those justified, rather than training all terms and attempting to decouple their effects post-hoc.

7 Note: in the case studies presented in this thesis, it was found that the same resources were sufficient for the simplest and most complex models, due to the relatively small number of terms therein. We expect for larger models, e.g. $|\vec{\alpha}| > 10$, that the resources allocated ought to be proportional to the cardinality, which is an in-built option in the QMLA software.

1.3.1 *Test in simulation*

Before considering the real experimental data (Fig. 1.3), we first test the ES in simulation under ideal conditions. That is, we assume the ability to prepare arbitrary probe states, and use a random probe set (see **??**), and use the full expectation value as the likelihood, $|\langle\psi|e^{-i\hat{H}_j t}|\psi\rangle|^2$. Of course, this is infeasible since we presume access to the full state at the time of measurement, but this can be seen as a best-case scenario for this application, because the realistic case loses information by tracing out the environmental qubit at measurement. We vary the target $\hat{H}_0$, among a series of ten models, which are all valid models achievable by the ES.

1.4 EXPERIMENT DESIGN CONSTRAINTS

Moving to analyse the experimental setup, there are a number of constraints which we must account for in training models. Firstly, the $\pi/2$-pulse applied to the prepared qubit ($|\psi_0\rangle \rightarrow |\psi_1\rangle$ in Fig. 1.2) means that the state before evolution is always $|+\rangle$ in the computational basis; this is a severe limitation on model training, as we saw in **??**. Moreover, this places a bias on the interactions QMLA is likely to identify: we show in Fig. 1.2 how QHL performs in training the same model using (i) the probe set available experimentally; (ii) a more general (random) probe set. This bias adds a caveat to the outcome of this study: the suppression of terms means we are more likely to find some genuine interactions than others, so the champion model is capturing the decoherence with respect only to one basis.

The experiment was run with increasing t for the duration of the first decay of the NVC, i.e. until it had dephased, so the data available for examination terminate at $t_{\text{max}} \sim 4\mu s$, see Fig. 1.3. As discussed in **??**, usually it is helpful to allow an experiment design heuristic (EDH) to choose the experimental controls, including the evoltuion time, t, against which the model is trained at each experiment; the default particle guess heuristic (PGH) attempts to select t at the upper boundary of times where the model is expected to be predictive, such as to maximise the information gained by the experiment (see **??**). Here, however, we can not allow the EDH to select arbitrary t, since we do not have data beyond t_{max}.

We require a custom EDH to account for the constraints outlined, with the following considerations:

1. We may only assume access to the probe $|+\rangle$ on the spin qubit
   (a) we further assume the environmental spin is polarised by the same microwave pulse, such that the global probe available is $|\psi\rangle = |+\rangle\,|+'\rangle$, with $|+'\rangle = \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}}$ and ϕ is random [19].
2. We can not allow the choice of any t:
   (a) Any $t > t_{\text{max}}$, arising from a thin parameter distribution, must be mapped to some $0 < t \leq t_{\text{max}}$.

(b) All nominated t must be mapped to the nearest available t in the dataset so that the likelihoods are as close as possible to simulating the true system.

3. Much of the physics of interest occurs at relatively high times, i.e. because the rotation (MHz) terms dominate, the decay of the peaks can be seen as evidence of the bath, notably through hyperfine terms in the model.

(a) We therefore wish to enforce that all models are trained on those data ($t \geq 2\mu s$), even if their parameter distribution is insufficiently narrow to yield those times naturally.

Accounting for these, we construct an EDH which mixes the robust, adaptive nature of particle guess heuristic (PGH), useful for refining an initially broad $\Pr(\vec{\alpha})$, with a primitive, linear time-selection, useful to ensure the trained parameters at least attempt to account for the physics we are actually interested in. That is, with each model trained for N_e experiments, we train according to the standard PGH for the first $N_e/2$, but force the training to mediate over the available data for the latter $N_e/2$.

1.5 RESULTS

We apply the ES described in Section 1.3 to the raw data of Fig. 1.3: the results are summarised in Fig. 1.5. We first focus on the overall outcomes: the most blunt figure of merit of interest is simply whether QMLA overfits or underits the true parameterisation. In preliminary analysis we run 500 instances with varying $\hat{H}_0$, varying the cardinality of $\hat{H}_0$, so we can broadly gauge the tendency towards over- and under-fitting: we see that in $\sim 50\%$ of instances the correct cardinality is found, rising to $\sim 86\%$ by allowing ± 1 term, Fig. 1.5**(b)**. In general, the champion models from each instance are highly predictive: the median coefficient of determination between the systems' and corresponding champion models' data is $R^2 = 0.84$.

Then, considering the performance of the algorithm on whole, we perform runs of 100 instances on the exerpimental data as well as simulated data, where the simulation assumes[8] $\hat{H}_0 = \hat{S}_{xyz}\hat{A}_z$, The set of models selected most frequently are shown in Fig. 1.5**(c)**, and each model is trained with $N_e = 1000, N_p = 3000$, with the volumes of those models (in the experimental case) shown in Fig. 1.5**(d)**. In particular, the most prominent models, $\{\hat{S}_{x,y,z}\hat{A}_z, \hat{S}_{x,y,z}\hat{A}_{y,z}, \hat{S}_{x,y,z}\hat{A}_{x,z}, \hat{S}_{y,z}\hat{A}_z\}$ are found collectively in 74% (87%) of instances on the experimental (simulated) data; the win rate and R^2 of all models (which won at least one instance) are reported in Table 1.1. It is noteworthy that even in the simulated case, the same models mislead QMLA: this suggests that the resultant physics from these models is substantially similar to that of the true model[9]. These models are defensible with respect to the descriptions of Section 1.2, since in each case they detect the interaction between the spin qubit and the

8 Here we work backwards by setting the target model as that which QMLA deemed most appropriate for the available data. We posit that this choice is arbitrary and doesn't fundamentally change the discussion of this chapter, merely aiding in analysing the performance of the algorithm with respect to a concrete example.

9 Alternatively, that the same systematic error misdirects the search in both cases.

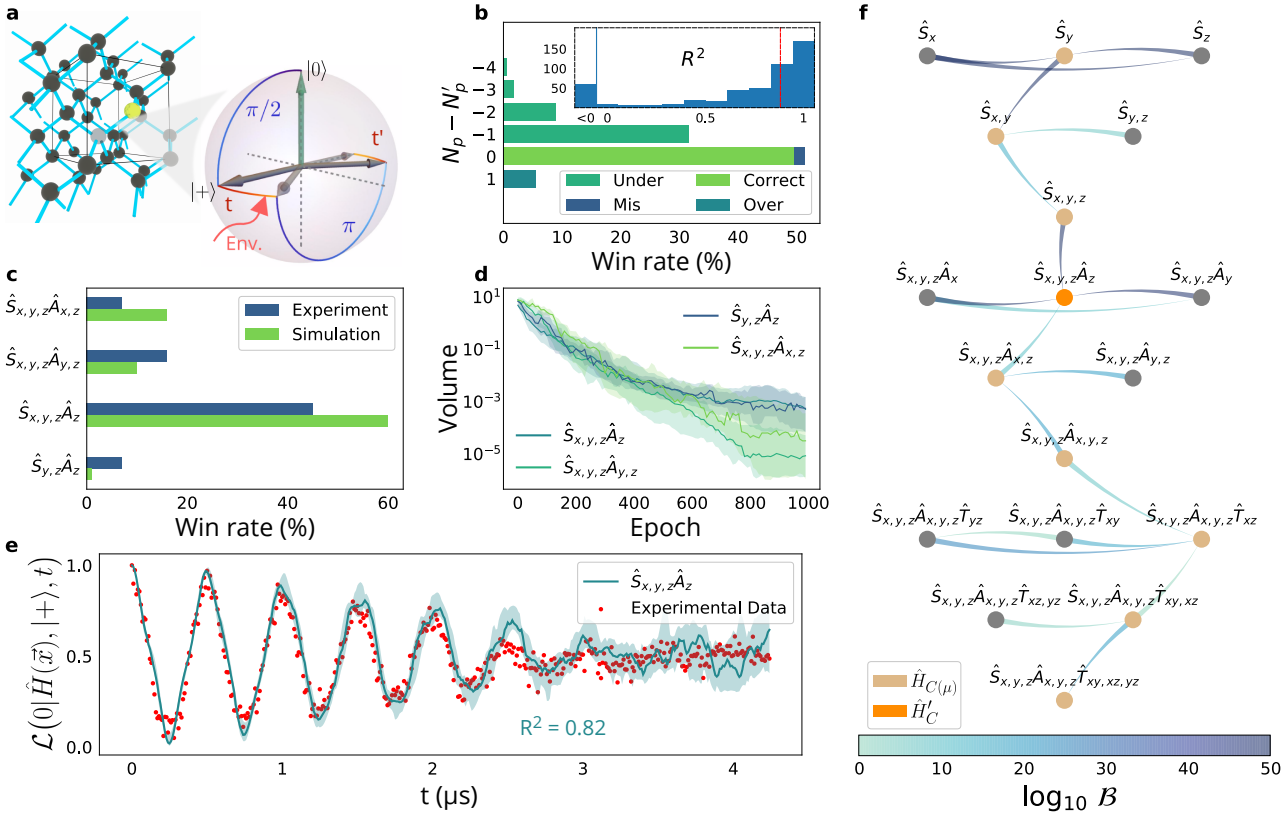


Figure 1.5: QMLA results on simulated and experimental data, describing a nitrogen-vacancy centre (NVC) system. **a**, The carbon lattice providing the outer environment for the NV centre, along with the time evolution of the electron spin state (represented on a Bloch sphere) during the pulses for the Hahn echo sequences. These steps are expanded in Fig. 1.2, although here the final $\pi/2$ pulse is omitted. **b**, Simulation of 500 independent QMLA instances, where $\hat{H}_0$ is chosen randomly. The win rate is reported against the difference $(N_p - N'_p)$ between the number of parameters in $\hat{H}'$ and $\hat{H}_0$, respectively. The *under-parameterised (over-parameterised)* class refers to models with less (more) parameters than $\hat{H}_0$. *Correct* indicates that exactly $\hat{H}_0$ was found. The *mis-parameterised* class groups models with the same parametrisation cardinality as $\hat{H}_0$, but different Hamiltonian terms. **Inset**, Histogram of occurrences of R^2 values for each retrieved $\hat{H}'$ against a sampling of datapoints from $\hat{H}_0$. The blue vertical line groups together all those instance champions with $R^2 < 0$, and the median $R^2 = 0.84$ is shown as a red dotted line. **c**, Win rates of the top four models for 100 QMLA instances, against both simulated and experimental data. On experimental data, $\hat{H}_0$ is unknown, while simulations use $\hat{H}_0 = \hat{S}_{x,y,z}\hat{A}_z$. **d**, Total volume spanned by the parameters' probability distribution across progressive epochs, for the models in **(c)**. The shaded area show the 67% confidence region of volumes, taken from instances where those models were deemed $\hat{H}'$. **e**, Simulated likelihoods reproduced by the model with the highest win rate ($\hat{S}_{x,y,z}\hat{A}_z$, turquoise), compared with corresponding NV-centre system experimental data (red dots, extracted from the observed photoluminescence of the first Hahn echo decay). Error bars are smaller than the dots. The shaded area indicates the 67% confidence region of likelihoods predicted from the instances where $\hat{H}' = \hat{S}_{x,y,z}\hat{A}_z$ **f**, A single QMLA instance against experimental data in **(e)**, depicted as a directed acyclic graph. The thin end of each edge points to the favoured model; the colour of the edges depict the strength of evidence, $\log_{10}\mathcal{B}$, where $\mathcal{B}$ is the BF between those two models. Champions of each layer, $\hat{H}_{C(\mu)}$, are in light brown, whereas the global champion $\hat{H}'$ is in orange and all other candidate models are grey.

Model	Experiment		Simulation	
	Wins	R^2	Wins	R^2
$\hat{S}_{y,z}\hat{A}_z$	9	0.8	1	0.26
$\hat{S}_y\hat{A}_{x,z}$	2	0.63		
$\hat{S}_{x,y,z}\hat{A}_z$	45	0.86	61	0.97
$\hat{S}_{x,y,z}\hat{A}_y$			1	-0.54
$\hat{S}_{x,y,z}\hat{A}_{x,y}$	3	0.81		
$\hat{S}_{x,y,z}\hat{A}_{y,z}$	14	0.83	10	0.96
$\hat{S}_{x,y,z}\hat{A}_{x,z}$	6	0.64	15	0.99
$\hat{S}_{x,y,z}\hat{A}_{x,y,z}$	2	0.72	5	0.97
$\hat{S}_{x,y,z}\hat{A}_{x,z}\hat{T}_{xz}$			1	0.68
$\hat{S}_{x,y,z}\hat{A}_{x,y,z}\hat{T}_{xz}$			5	0.77
$\hat{S}_y\hat{A}_{x,y,z}\hat{T}_{xy,xz,yz}$	2	0.31		
$\hat{S}_{x,y,z}\hat{A}_{x,y,z}\hat{T}_{xy,xz}$	4	0.67	1	0.32

Table 1.1: QMLA win rates and R^2 for models based on experimental data and simulations. We state the number of QMLA instances won by each model and the average R^2 for those instances as an indication of the predictive power of winning models.

environmental qubit, i.e. the hyperfine terms $\hat{A}_i$, especially $\hat{A}_z$ which occurs in 97% (99%) of champion models. We discuss some physical insights from these results in Section 1.5.1.

The most frequently identified model, $\hat{H}' = \hat{S}_{x,y,z}\hat{A}_z$, is found in 45% (61%) of instances: we show its attempt to reproduce the dynamics of Fig. 1.3 in Fig. 1.5**(e)**, showing excellent agreement with the raw data, with $R^2 = 0.82$. This serves as an essential sanity check: we can intuitively see that QMLA has distilled a model which captures at least *some* of the most important physical interactions the target NVC system is subject to; otherwise we would not see such clear overlap between the prediced and true dynamics.

Finally we display the model search as a directed acyclic graph (DAG) in Fig. 1.5**(f)**, where models are represented on nodes on the graph's layers (equivalent to ET branches), and their parents are resident on the branch immediately above their own. Comparisons between models, $\hat{H}_i, \hat{H}_j$, are shown as edges between nodes on the graph, coloured by the strength of evidence of the outcome, i.e. the Bayes factor, B_{ij}. Each layer, μ, nominates their branch champion, $\hat{H}_{C(\mu)}$; the set of branch champions are consolidated to determine the global champion, $\hat{H}'_C$.

1.5.1 *Analysis*

Considering the runs summarised in Fig. 1.5, here we will present some further perspectives. Fig. 1.6 first details all models considered in the 200 instances comprising the experimental and simulated QMLA runs, as well as the win rate of each model. This ES is designed to study

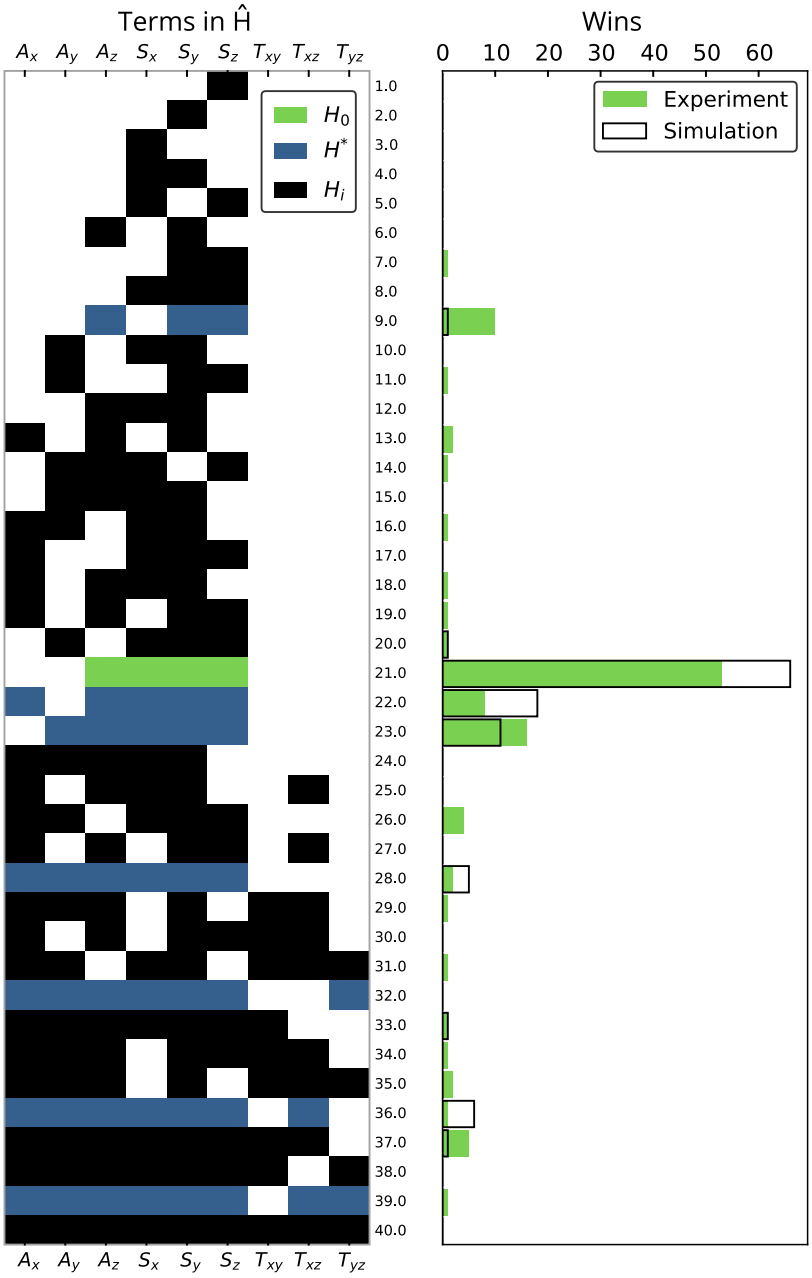
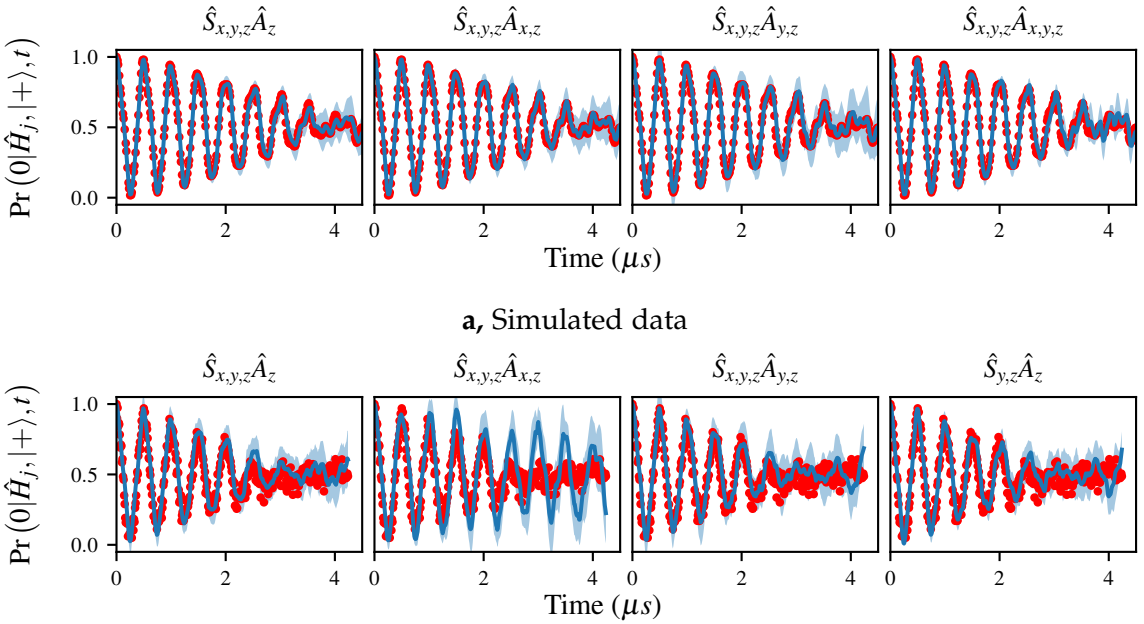


Figure 1.6: **Left**: map of the various Hamiltonian terms included in each of the possible 40 candidate models explored by QMLA during any of the 100 instances on either simulated or experimental data. IDs of candidate models are on the vertical axis, and labels for the terms on the horizontal axis. The true model $\hat{H}_0$ for the simulated case is highlighted in green and a subset of credible models in blue, i.e. models which may reasonbaly be expected to describe the targeted nitrogen-vacancy centre (NVC) from theoretical arguments. **Right**: number of wins for each of the candidate models out of 100 independent QMLA runs. Cases adopting simulated data are shown by empty bars, with those using the experimental dataset shown by green bars. Implementation details are listed in Table A.1



a, Simulated data

b, Experimental data

Figure 1.7: Dynamics reproduced by various QMLA champion models for **(a)** simulated and **(b)** experimental data. Likelihoods, $\Pr(0)$ are shown on y–axis with time on x–axis. Red dots give the true dynamics of $\hat{H}_0$, while the blue lines show the median reconstruction by $\hat{H}'$ from all the instances where that model was deemed $\hat{H}'$, with light blue showing the 67% confidence region. $\hat{H}'$ is listed on top of each plot; the number of instances won by each model can be read from Table 1.1. Implementation details are listed in Table A.1

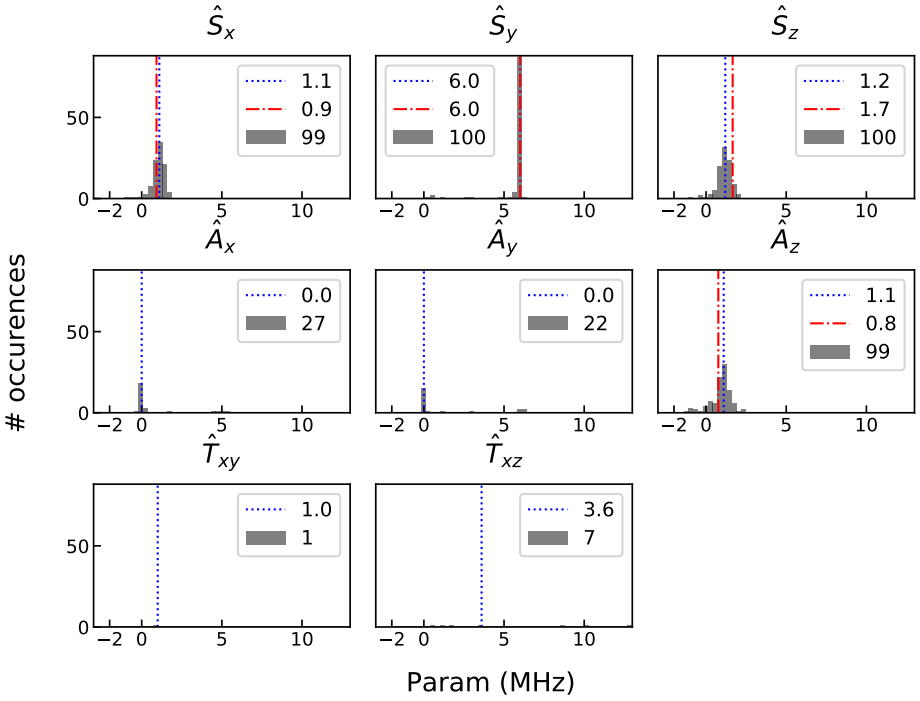
a small subspace of the overall available space: only 40 unique models are constructed. We highlight a number of *credible* models which we deem especially valid approximations of the target system, i.e. which contain the most viable approximations.

Fig. 1.7 shows the reproduction of dynamics of the top four models from both simulated and experimental runs. We see that each model faithfully captures the essential dynamics arising from the respective target systems; this alone is insufficient to conclude that the true model has been identified, but serves as a valuable *sanity-check*, convincing us that the output of QMLA is at least a sensible approximation of $\hat{H}_0$, if not the absolute true model.

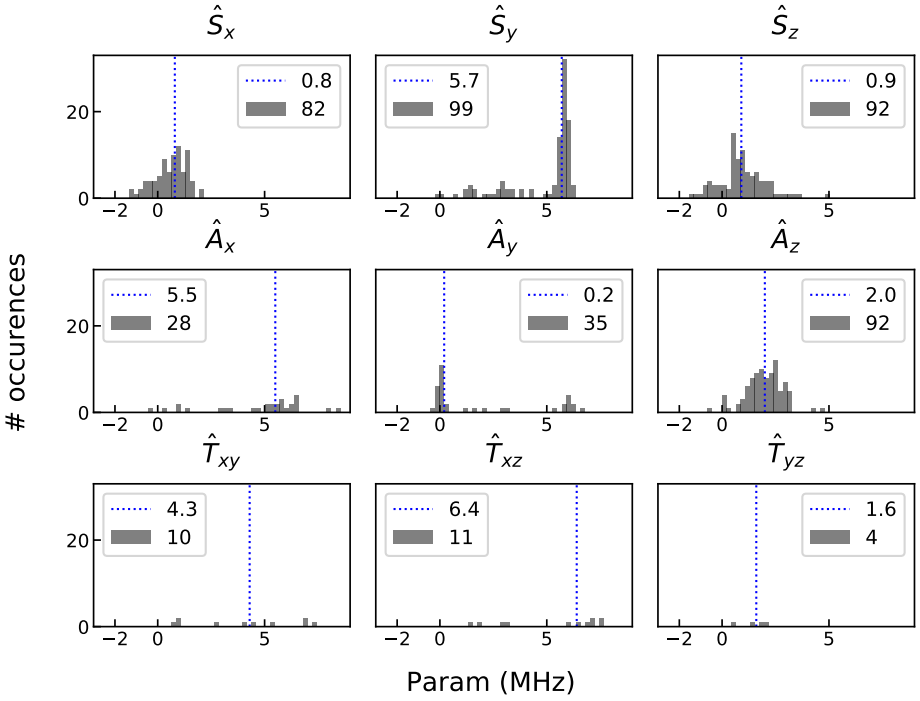
The key insight promised by QMLA is to identfiy the interactions present in the studied system, which in this case was the decoherence processes of an NVC. In Fig. 1.8 we show the number of times each of the terms permitted, i.e. those in Eq. (1.11), are included in the champion model, as well as the distribution of parameter estimates for those terms. From the simulated case, we see that those terms which are in $\hat{H}_0$, i.e. $\mathcal{T}_0$, are found in almost all instances. Furthermore, while some terms are erroneoulsy found in $> 40\%$ of instances, they are found with less than a quarter of the frequency of the true terms, so these may be reasonably ruled

out in post-processing the QMLA results. The inaccurate terms found most often are seen to have (almost) negligible parameters: in conjunction with domain expertise, users can determine whether the inclusion of these terms are meaningful or simply artefacts of slight overfitting. In the experimental run, on the other hand, we see a similar gulf in frequency between some terms. Namely, $\{\hat{S}_x, \hat{S}_y, \hat{S}_z, \hat{A}_z\}$ are found in over 50 instances *more* than all other terms: we therefore conclude that those terms contribute most strongly to the NVC's decoherence process.

We have thus characterised the interactions which dominate the decoherence process for a given NVC. In doing so, we identified not only the terms present, but also the strength (i.e. parameteres) of those terms. Automated characterisation of quantum systems will be essential in the development of quantum technologies, whether for calibration of controlled devices, or alignment of experimental systems for optimal results. While this demonstration must be understood in the context of its limitations, e.g. the restricted basis studied, and the constrained model space searched, it represents a crucial proof of principle that QMLA is applicable to the task of automated quantum system characterisation. In the next chapter, we extend QMLA in simulation, to overcome the limitations mentioned here.



a, Simluated QMLA instances. Red dotted lines show the true parameters.



b, Experimental QMLA instances.

Figure 1.8: Histograms for parameters learned by QMLA champion models on (**a**) simulated and (**b**) experimental data. Blue dotted lines indicate the median for that parameter, while red dotted lines give the true parameter in the simulated case. Grey blocks show the number of champion models which found the parameter to have that value, and the number listed in the legend reports the total number of champion models which contained that term. Implementation details are listed in Table A.1

2

LARGER SYSTEMS

Chapter 1 concerned a two-qubit approximation of the short-time dynamics of an NVC. It is valid criticism of this analysis that the model space searched was reduced substantially through prior knowledge, and it therefore remains to test QMLA in a large model space, on physically meaningful data. In this chapter, we extend QMLA, to consider appoximations of NVC sytems using more qubits, representing several nuclear sites, which aim to capture the interactions between the target NVC and the environment. We will simulate the target system here, allowing us to make definite statements on the performance of QMLA, unlike the experimental data where we can not be sure of the dynamics' genreator.

2.1 TARGET SYSTEM

A more realistic model may be expected from consdiering the environment as a finite-size bath, consisting of n_s nuclear spins in addition to the NVC spin, i.e. the total number of qubits of such a model is $n_q = 1 + n_s$. Such effects can be highlighted by Hahn echo measurements, as in Fig. 1.2, except reversing the evoluting by $t' = t$ instead of $t' = 2t$ [14, 8], so our simulations will use this measurement scheme.

Since we are simulating the target system, we may choose the approximation we wish to invoke. We use the secular approximation, i.e. we assume the magnetic field is perfectly aligned along the z-axis [15]: recalling Eq. (1.1), the NVC spin qubit rotates only about the z-axis, and coupling between the electron and nuclear qubits are only via $\hat{S}_z \cdot \hat{A}_z^\chi$. We include the effect of the nuclear spins' rotations, which are much weaker. In total then, the set of nuclear spins, $\{\chi\}$, are mapped to n_s qubits:

$$\hat{H}_0 = \hat{S}_z + \sum_{j=2}^{n_q} \hat{S}_z \cdot \hat{A}_z^j + \sum_{w \in \{x,y,z\}} \sum_{j=2}^{n_q} \hat{I}_w^j. \tag{2.1}$$

For simplicity, we restate this in terms only of the Pauli matrices, where the first qubit refers to the NVC and the remaining qubits give the interactions and nuclear terms.

$$\hat{H}_0 = \hat{\sigma}_z^1 + \sum_{j=2}^{n_q} \hat{\sigma}_z^1 \hat{\sigma}_z^j + \sum_{w \in \{x,y,z\}} \sum_{j=2}^{n_q} \hat{\sigma}_w^j, \tag{2.2}$$

so in total, $\mathcal{T}_0$ has 1 term for the NVC qubit, n_s terms for hyperfine couplings and $3n_s$ terms for the nuclei: $|\mathcal{T}_0| = 1 + 4n_s$.

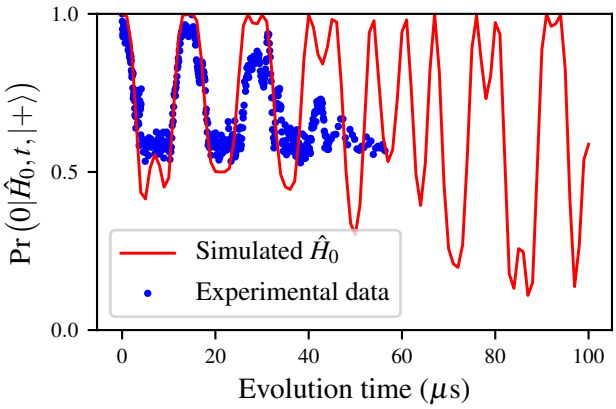


Figure 2.1: Long-time dynamics for nitrogen-vacancy centre, red, showing revivals, generated by $\hat{H}_0$ from Eq. (2.2), via Hahn echo measurement with $t' = t$. For comparison, experimentally generated dynamics are shown in blue.

We set the goal of QMLA as finding the approximation of Eq. (2.2), by allowing it to consider a wider set of terms. The permissible terms are then all spin rotation terms, as well as all nuclei rotation terms, and the coupling terms:

$$\mathcal{T} = \left\{ \begin{array}{c} \hat{S}_w = \hat{\sigma}^1_w, \\ \hat{I}^j_w = \hat{\sigma}^j_w, \\ \hat{S}_w \cdot \hat{A}_w = \hat{\sigma}^1_w \hat{\sigma}^j_w \end{array} \right\} \tag{2.3}$$

for $w = \{x, y, z\}$ and $j \in \{2, ..., n'_q\}$. Importantly, in general $n'_s + 1 = n'_q \neq n_q$, i.e. QMLA can search a space of models with more or less spins, n'_s, than are truly in $\hat{H}_0$. In total then, $|\mathcal{T}| = 3 + 3n'_s + 3n'_s = 3 + 6n'_s$.

We set the system parameters based on theoretical predictions, listed in Table 2.1.

Here our aim is to test QMLA, so the choice of n_s and n'_s are arbitrary; for the target system we use $n_s = 4$ proximal spins, so that, from Eq. (2.2), $|\mathcal{T}_0| = 13$, and we allow candidates to consider $n'_s = 5$, so $|\mathcal{T}| = 33$.

In summary, irrespective of the underlying physics we are simulating, here QMLA is aiming to identify the 13 terms truly present in Q, while searching the space of 33 permissible terms. Without imposing any restrictions on which combinations of terms are allowed, each term is simply either in $\hat{H}'$ or not, so can be thought of as binary variables: the total model space is therefore of size $2^{33} \approx 10^{10}$.

Term	$\hat{t}$	Meaning	Parameter (Hz)	$\in \hat{H}_0$
$\hat{S}_x$	$\hat{\sigma}_x^1$	NVC rotation about x-axis	2×10^9	No
$\hat{S}_y$	$\hat{\sigma}_y^1$	NVC rotation about y-axis	2×10^9	No
$\hat{S}_z$	$\hat{\sigma}_z^1$	NVC rotation about z-axis	2×10^9	Yes
$\hat{S}_x \cdot \hat{A}_x^j$	$\hat{\sigma}_x^1\hat{\sigma}_x^j$	Coupling b/w spin and j^{th} nuclear qubit about x-axis	0.2×10^6	No
$\hat{S}_y \cdot \hat{A}_y^j$	$\hat{\sigma}_y^1\hat{\sigma}_y^j$	Coupling b/w spin and j^{th} nuclear qubit about y-axis	0.2×10^6	No
$\hat{S}_z \cdot \hat{A}_z^j$	$\hat{\sigma}_z^1\hat{\sigma}_z^j$	Coupling b/w spin and j^{th} nuclear qubit about z-axis	0.2×10^6	Yes
$\hat{I}_x^j$	$\hat{\sigma}_x^j$	j^{th} nuclear spin rotation about x-axis	66×10^3	Yes
$\hat{I}_y^j$	$\hat{\sigma}_y^j$	j^{th} nuclear spin rotation about y-axis	66×10^3	Yes
$\hat{I}_z^j$	$\hat{\sigma}_z^j$	j^{th} nuclear spin rotation about z-axis	15×10^3	Yes

Table 2.1: Extended model NVC terms

2.2 GENETIC ALGORITHM

genetic algorithms (GAs) provide a robust and thoroughly tested paradigm for searching large candidate spaces; this is a natural framework through which we can explore such an unresetricted model space. We have already extensively discussed the formalism of GAs in **??**, and specifically in the context of QMLA in **??**. Here we will use the same ES as described in **??**, i.e. where model generation is driven by a GA, and models are cast to chromosomes. In particular, candidate model's fitness will be computed from the residuals between their and the sytem's dynamics, described fully in **??**. This objective function (OF) relies on the definition of a validation dataset, $\mathcal{E}_v$, which we compose of tomographic probes and times generated uniformly up to $t_{max} = 100\mu s$, Fig. 2.2.

2.2.1 *Parameter learning*

Here, our primary goal is to validate QMLA's performance in a very large model space, with over 10^{10} valid candidates. Our focus on model generation, then, we do *not* train models individually: instead we assume access to a *perfect* parameter learning subroutine. That is, for each candidate considered, we simply assume knowledge of its parameters, $\vec{\alpha}$. This is a major caveat to the results of this chapter: no such perfect training scheme is known, so it remains to examine the detrimental effects of imprecisely finding $\vec{\alpha}' \approx \vec{\alpha}$. Moreover, while it is possible to extract information on the nuclear qubits from measuring only the NVC qubit, as in the Hahn echo measurements, it is uncertain whether any technique can simultaneously detect parameters of significantly varying orders of magnitude; For instance, some terms in Table 2.1 are $\mathcal{O}(\text{GHz})$,

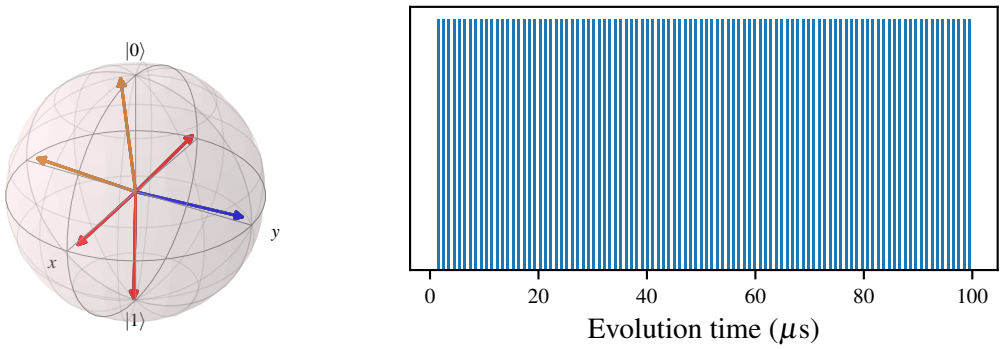


Figure 2.2: Evaluation dataset, $\mathcal{E}_v$, for nitrogen-vacancy centre genetic algorithm. **Left**, Probe state the NVC qubit is prepared in, on the Bloch sphere, i.e. Ψ_v is close to the tomographic basis. **Right**, Time comb evaluated against, i.e. uniformly distributed times up to $t_{max} = 100\mu$s are used for experiments in $\mathcal{E}_v$.

while others are $\mathcal{O}$(kHz); it is likely to prove difficult to discern the kHz parameters well, given that their contribution is equivalent to errors of order $\mathcal{O}(10^{-6})$ in the dominant GHz terms. Therefore we must caution that the results presented here, while demonstrating that QMLA *can* operate in large model spaces, are not immediately applicable to experimental systems, since there are outstanding challenges in the assessment of individual candidates, which must be overcome before the technique outlined can realisitcally succeed.

2.2.2 *Results*

At the instance level, we can see that the gene pool tends towards models of higher quality, captured by their F_1-score, Fig. 2.3**a**. The improvement in modelling is reflected in the branch champions' predictive power at reproducing data generated by the system, Fig. 2.3**b**.

Considering the overall run, we see that QMLA is searching in a vast model space where randomly sampled models have poor F_1-score on average, Fig. 2.4**a**. QMLA efficiently explores the space by quickly moving into a subspace of high F_1-score, nominating $\hat{H}' = \hat{H}_0$ precisely in 85% of instances, Fig. 2.4**b,c**. The number of times each of the terms considered, Eq. (2.3), are present in $\hat{H}_0$ offers the most important insight from QMLA, namely the evidence in favour of each term's presence, which can be used to infer the most likely underlying physics. Here, $\hat{t} \in \mathcal{T}_0$ are found in $\geq 94\%$ of instances, while $\hat{t} \notin \mathcal{T}_0$ are found in $\leq 11\%$, shown in Fig. 2.5 and listed in Table 2.2. Such a discrepency, as well as the win rates for the models, allows for the clear declaration of the model $\hat{H}_0$ as the favoured representation for the quantum system.

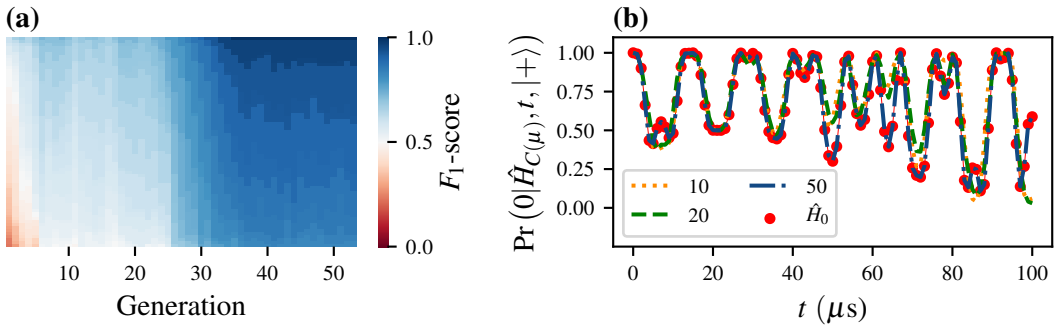


Figure 2.3: Instance of the genetic algorithm for simulated nitrogen-vacancy centre system with four qubits. **a**, Branch champions' dynamics. Each generation, μ, nominates a branch champion, $\hat{H}_{C(\mu)}$. Here, progressive generations' champions dynamics are shown against those of the target system, $\hat{H}_0$ (red). **b**, Gene pool progression for the GA. Each tile represents a candidate model by its F_1-score. Each generation considers $N_m = 72$ models; the GA runs for $N_g = 53$ generations.

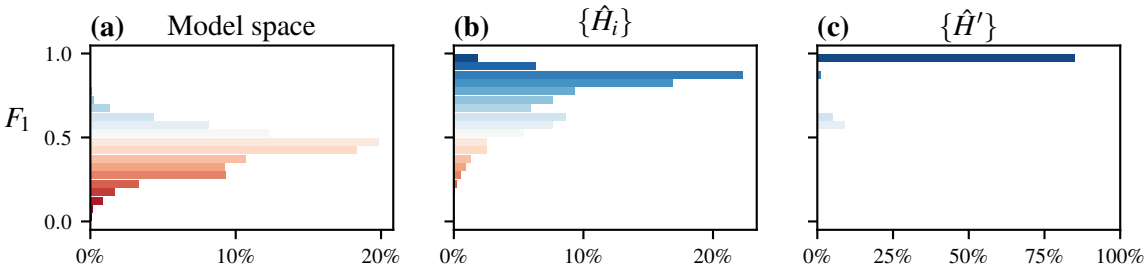


Figure 2.4: Nitrogen-vacancy centre genetic algorithm run. **(a)**, F_1-score of 10^6 samples from the model space of $2^{33} \approx 10^{10}$ candidate models, normally distributed around $f = 0.44 \pm 0.12$. **(b)**, The models explored during the model search of all instances combined, $\{\hat{H}_i\}$, show that QMLA tends towards stronger models overall, with $f = 0.79 \pm 0.16$ from $\sim 140,000$ chromosomes across the 100 instances, i.e. each instance tests ~ 1400 distinct models. **(c)**, Champion models from each instance, showing QMLA finds strong models in general, and in particular finds the true model ($\hat{H}_0$, with $f = 1$) in 85% of cases.

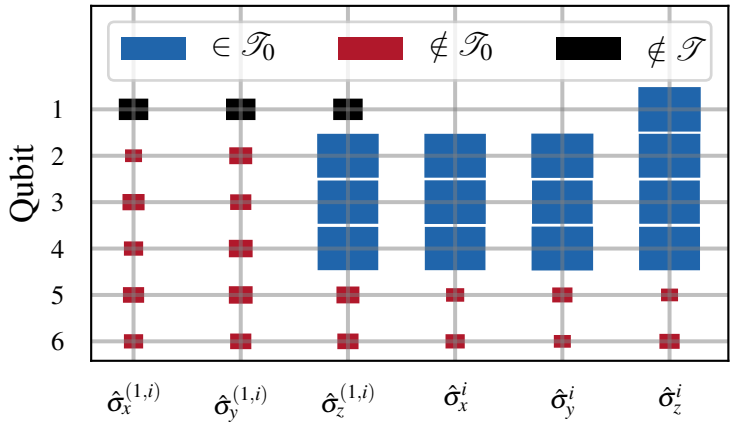


Figure 2.5: Hinton diagram of terms found for 4-qubit nitrogen-vacancy centre model. Terms are either in the target model ($\in \mathcal{T}_0$, blue) or not ($\notin \mathcal{T}_0$, red), or else not considered ($\notin \mathcal{T}$, black). Terms acting solely on the first qubit are the NVC spin's rotation terms, $\hat{\sigma}_w^1$, while each nuclear site also has rotation terms $\hat{\sigma}_w^j$. Hyperfine terms, $\hat{\sigma}_w^{(1,j)}$, couple the NVC qubit with the j^{th} nuclear spin. The precise rate at which each term is detected can be read from Table 2.2.

Qubit	$\hat{\sigma}_x^{(1,i)}$	$\hat{\sigma}_y^{(1,i)}$	$\hat{\sigma}_z^{(1,i)}$	$\hat{\sigma}_x^i$	$\hat{\sigma}_y^i$	$\hat{\sigma}_z^i$
1	-	-	-	0	0	100
2	5	11	97	97	99	97
3	10	9	94	96	94	94
4	7	12	94	94	97	95
5	9	12	11	6	8	5
6	7	9	9	7	5	8

Table 2.2: Percentage of instances for which each term is found by QMLA GA studying NVC system.

APPENDIX

A

FIGURE REPRODUCTION

Most of the figures presented in the main text are generated directly by the QMLA framework. Here we list the implementation details of each figure so they may be reproduced by ensuring the configuration in Table A.1 are set in the launch script. The default behaviour of QMLA is to generate a results folder uniquely identified by the date and time the run was launched, e.g. results can be found at the *results directory* qmla/Launch/Jan_01/12_34. Given the large number of plots available, ranging from high-level run perspective down to the training of individual models, we introduce a plot_level $\in \{1, ..., 6\}$ for each run of QMLA: higher plot_level informs QMLA to generate more plots.

Within the results directory, the outcome of the run's instances are stored, with analysis plots broadly grouped as

1. evaluation: plots of probes and times used as the evaluation dataset.
2. single_instance_plots: outcomes of an individual QMLA instance, grouped by the instance ID. Includes results of training of individual models (in model_training), as well as sub-directories for anlaysis at the branch level (in branches) and comparisons.
3. combined_datasets: pandas dataframes containing most of the data used during analysis of the run. Note that data on the individual model/instance level may be discarded so some minor analyses can not be performed offline.
4. exploration_strategy_plots plots specifically required by the ES at the run level.
5. champion_models: analysis of the models deemed champions by at least one instance in the run, e.g. average parameter estimation for a model which wins multiple instances.
6. performance: evaluation of the QMLA run, e.g. the win rate of each model and the number of times each term is found in champion models.
7. meta analysis of the algorithm' implementation, e.g. timing of jobs on each process in a cluster; generally users need not be concerned with these.

In order to produce the results presented in this thesis, the configurations listed in Table A.1 were input to the launch script. The launch scripts in the QMLA codebase consist of many configuration settings for running QMLA; only the lines in snippet in Listing A.1 need to be set according to altered to retrieve the corresponding figures. Note that the runtime of QMLA grows quite quickly with N_e, N_p (except for the AnalyticalLikelihood ES), especially for the entire QMLA algorithm; running QHL is feasible on a personal computer in < 30 minutes for $N_e = 1000; N_p = 3000$.

```
#!/bin/bash
```

```
###############
# QMLA run configuration
###############
num_instances=1
run_qhl=1 # perform QHL on known (true) model
run_qhl_mulit_model=0 # perform QHL for defined list of models.
exp=200 # number of experiments
prt=1000 # number of particles

###############
# QMLA settings
###############
plot_level=6
debug_mode=0

###############
# Choose an exploration strategy
###############

exploration_strategy='AnalyticalLikelihood'
```

Listing A.1: QMLA Launch scipt

Figure	Exploration Strategy	N_E	N_P	Data
??	DemoHeuristicPGH	1000	3000	Nov_27/19_39
	DemoHeuristicNineEighths	1000	3000	Nov_27/19_40
	DemoHeuristicTimeList	1000	3000	Nov_27/19_42
	DemoHeuristicRandom	1000	3000	Nov_27/19_47
??	DemoProbesPlus	1000	3000	Nov_27/14_43
	DemoProbesZero	1000	3000	Nov_27/14_45
	DemoProbesTomographic	1000	3000	Nov_27/14_46
	DemoProbes	1000	3000	Nov_27/14_47
??	DemoProbesPlus	1000	3000	Nov_27/14_43
	DemoProbesZero	1000	3000	Nov_27/14_45
	DemoProbesTomographic	1000	3000	Nov_27/14_46
	DemoProbes	1000	3000	Nov_27/14_47
??	AnalyticalLikelihood	500	2000	Nov_16/14_28
??	DemoIsing	500	5000	Nov_18/13_56
??	DemoIsing	1000	5000	Nov_18/13_56
??	DemoIsing	1000	5000	Nov_18/13_56
??	IsingLatticeSet	1000	4000	Nov_19/12_04
	IsingLatticeSet	1000	4000	Nov_19/12_04
	IsingLatticeSet	1000	4000	Nov_19/12_04
??	IsingLatticeSet	1000	4000	Sep_30/22_40
	HeisenbergLatticeSet	1000	4000	Oct_22/20_45
	FermiHubbardLatticeSet	1000	4000	Oct_02/00_09

Table A.1: Implementation details for figures used in the main text. Continued in Table A.2.

Figure	Exploration Strategy	N_E	N_P	Data
??	DemoBayesFactorsByFscore	500	2500	Dec_09/12_29
	DemoFractionalResourcesBayesFactorsByFscore	500	2500	Dec_09/12_31
	DemoBayesFactorsByFscore	1000	5000	Dec_09/12_33
	DemoBayesFactorsByFscoreEloGraphs	500	2500	Dec_09/12_32
??	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
??	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
??	HeisenbergGeneticXYZ	500	2500	Dec_10/16_12
	HeisenbergGeneticXYZ	500	2500	Dec_10/16_12
Fig. 1.6	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
Fig. 1.7	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
Fig. 1.6	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
Fig. 1.8	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
Fig. 2.2	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
Fig. 2.3	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
Fig. 2.4	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_08/23_58
Fig. 2.5	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_08/23_58

Table A.2: [Continued from Table A.1] Implementation details for figures used in the main text.

B

FUNDAMENTALS

There are a number of concepts which are fundamental to any discussion of quantum mechanics (QM), but are likely to be known to most readers, and are therefore cumbersome to include in the main body of the thesis. We include them here for completeness[1].

B.1 LINEAR ALGEBRA

Here we review the language of linear algebra and summarise the basic mathematical techniques used throughout this thesis. We will briefly recall some definitions for reference.

- Notation

Definition of	Representation
Vector (or *ket*)	$\|\psi\rangle$
Dual Vector (or *bra*)	$\langle\psi\|$
Tensor Product	$\|\psi\rangle \otimes \|\phi\rangle$
Complex conjugate	$\|\psi^*\rangle$
Transpose	$\|\psi\rangle^T$
Adjoint	$\|\psi\rangle^\dagger = (\|\psi\rangle^*)^T$

Table B.1: Linear algebra definitions.

The dual vector of a vector (ket) $|\psi\rangle$ is given by $\langle\psi| = |\psi\rangle^\dagger$.

The *adjoint* of a matrix replaces each matrix element with its own complex conjugate, and then switches its columns with rows.

$$M^\dagger = \begin{pmatrix} M_{0,0} & M_{0,1} \\ M_{1,0} & M_{1,1} \end{pmatrix}^\dagger = \begin{pmatrix} M^*_{0,0} & M^*_{0,1} \\ M^*_{1,0} & M^*_{1,1} \end{pmatrix}^T = \begin{pmatrix} M^*_{0,0} & M^*_{1,0} \\ M^*_{0,1} & M^*_{1,1} \end{pmatrix} \tag{B.1}$$

1 Much of this description is reproduced from my undergraduate thesis [20].

The *inner product* of two vectors, $|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix}$ and $|\phi\rangle = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{pmatrix}$ is given by

$$\langle\phi|\psi\rangle = (|\phi\rangle^\dagger)\,|\psi\rangle = (\phi_1^* \ \phi_2^* \ \dots \ \phi_n^*) \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} = \phi_1^*\psi_1 + \phi_2^*\psi_2 + \cdots + \phi_n^*\psi_n \tag{B.2}$$

$|\psi\rangle_i, |\phi\rangle_i$ are complex numbers, and therefore the above is simply a sum of products of complex numbers. The inner product is often called the *scalar product*, which is in general complex.

B.2 POSTULATES OF QUANTUM MECHANICS

There are numerous statements of the postulates of quantum mechanics. Each version of the statements aims to achieve the same foundation, so we endeavour to explain them in the simplest terms.

1 Every moving particle in a conservative force field has an associated wave-function, $|\psi\rangle$. From this wave-function, it is possible to determine all physical information about the system.

2 All particles have physical properties called observables (denoted Q). In order to determine a value, Q, for a particular observable, there is an associated *operator* $\hat{Q}$, which, when acting on the particles wavefunction, yields the value times the wavefunction. The observable Q is then the eigenvalue of the operator $\hat{Q}$.

$$\hat{Q}\,|\psi\rangle = q|\psi\rangle \tag{B.3}$$

3 Any such operator $\hat{Q}$ is Hermitian

$$\hat{Q}^\dagger = \hat{Q} \tag{B.4}$$

4 The set of eigenfunctions for any operator $\hat{Q}$ forms a complete set of linearly independent functions.

5 For a system with wavefunction $|\psi\rangle$, the expectation value of an observable Q with respect to an operator $\hat{Q}$ is denoted by $\langle q\rangle$ and is given by

$$\langle q\rangle = \langle\psi|\hat{Q}|\psi\rangle \tag{B.5}$$

6 The time evolution of $|\psi\rangle$ is given by the time dependent *Schrodinger Equation*

$$i\hbar\frac{\partial\psi}{\partial t} = \hat{H}\psi, \tag{B.6}$$

where $\hat{H}$ is the system's Hamiltonian.

Using these building blocks, we can begin to construct a language to describe quantum systems.

B.3 STATES

An orhthonormal basis consists of vectors of unit length which do not overlap, e.g. $|x_1\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |x_2\rangle = \begin{pmatrix}0\\1\end{pmatrix} \Rightarrow \langle x_1|x_2\rangle = 0$. In general, if $\{|x\rangle\}$ are the eigenstates of a system, then the system can be written as some state vector, $|\psi\rangle$, in general a superposition over the basis-vectors:

$$|\psi\rangle = \sum_x a_x|x\rangle \tag{B.7a}$$

$$\text{subject to} \quad \sum_x |a_x|^2 = 1, \quad a_x \in \mathbb{C} \tag{B.7b}$$

The *state space* of a physical system (classical or quantum) is then the set of all possible states the system can exist in, i.e the set of all possible values for $|\psi\rangle$ such that Eq. (B.7b) are satisfied.

For example, photons can be polarised horizontally ($\leftrightarrow$) or vertically ($\updownarrow$); take those two conditions as observable states to define the eigenstates of a two-level system, so we can designate the photon as a qubit. Then we can map the two states to a 2-dimensional, x-y plane: a general vector on such a plane can be represented by a vector with coordinates $\begin{pmatrix}x\\y\end{pmatrix}$. These polarisations can then be thought of as standard basis vectors in linear algebra. Denote $\leftrightarrow$ as the eigenstate $|0\rangle$ and $\updownarrow$ as $|1\rangle$

$$|\leftrightarrow\rangle = |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad \text{A unit vector along x-axis} \tag{B.8a}$$

$$|\updownarrow\rangle = |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \qquad \text{A unit vector along y-axis} \tag{B.8b}$$

Now, in relation to the concept of superposition, we can consider, for example, a photon in an even superposition of the vertical and horizontal polarisations, evenly splitting the two basis vectors. As such, we would require that, upon measurement, it is equally likely that the

photon will *collapse* into the polarised state along x as it is to collapse along y. That is, we want $\Pr(\updownarrow) = \Pr(\leftrightarrow)$ so assign equal modulus amplitudes to the two possibilities:

$$|\psi\rangle = a\,|\leftrightarrow\rangle + b\,|\updownarrow\rangle\,, \quad \text{with} \quad \Pr(\updownarrow) = \Pr(\leftrightarrow) \Rightarrow |a|^2 = |b|^2 \tag{B.9}$$

We consider here a particular case, due to the significance of the resultant basis, where $\leftrightarrow$-polarisation and $\updownarrow$-polarisation have real amplitudes $a, b \in \mathbb{R}$.

$$\begin{aligned} &\Rightarrow a = \pm b \quad \text{but also} \quad |a|^2 + |b|^2 = 1 \\ \Rightarrow \quad &a = \frac{1}{\sqrt{2}} \quad ; \quad b = \pm\frac{1}{\sqrt{2}} \\ &\Rightarrow |\psi\rangle = \frac{1}{\sqrt{2}}\,|\leftrightarrow\rangle \pm \frac{1}{\sqrt{2}}\,|\updownarrow\rangle \\ &\Rightarrow |\psi\rangle = \frac{1}{\sqrt{2}}\,|0\rangle \pm \frac{1}{\sqrt{2}}\,|1\rangle \end{aligned} \tag{B.10}$$

These particular superpositions are of significance:

$$|+\rangle = \frac{1}{\sqrt{2}}\,(|0\rangle + |1\rangle) \tag{B.11a}$$

$$|-\rangle = \frac{1}{\sqrt{2}}\,(|0\rangle - |1\rangle) \tag{B.11b}$$

This is called the Hadamard basis: it is an equally valid vector space as the standard basis which is spanned by $\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}$, as it is simply a rotation of the standard basis.

B.3.1 *Mulitpartite systems*

In reality, we often deal with systems of multiple particles, represented by multiple qubits. Mathematically, we consider the state vector of a system containing n qubits as being the tensor product of the n qubits' individual state vectors[2]. For instance, suppose a 2-qubit system, $|\psi\rangle$ consisting of two independent qubits $|\psi_A\rangle$ and $|\psi_B\rangle$:

$$|\psi\rangle = |\psi_A\rangle\,|\psi_B\rangle = |\psi_A\psi_B\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \tag{B.12}$$

Consider first a simple system of 2 qubits. Measuring in the standard basis, these qubits will have to collapse in to one of the basis states $|0,0\rangle, |0,1\rangle, |1,0\rangle, |1,1\rangle$. Thus, for such a 2-qubit system, we have the general superposition

$$|\psi\rangle = \alpha_{0,0}|0,0\rangle + \alpha_{0,1}|0,1\rangle + \alpha_{1,0}|1,0\rangle + \alpha_{1,1}|1,1\rangle$$

2 We will later discuss entangled states, which can not be described thus.

where $\alpha_{i,j}$ is the amplitude for measuring the system as the state $|i,j\rangle$. This is perfectly analogous to a classical 2-bit system necessarily occupying one of the four possibilities $\{(0,0),(0,1),(1,0),(1,1)\}$.

Hence, for example, if we wanted to concoct a two-qubit system composed of one qubit in the state $|+\rangle$ and one in $|-\rangle$

$$\begin{aligned}
|\psi\rangle &= |+\rangle \otimes |-\rangle \\
|\psi\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \\
&= \frac{1}{2}\left[|00\rangle - |01\rangle + |10\rangle - |11\rangle\right] \\
&= \frac{1}{2}\left[\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}\right] \\
&= \frac{1}{2}\left[\begin{pmatrix}1\\0\\0\\0\end{pmatrix} - \begin{pmatrix}0\\1\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} - \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\right]. \\
\Rightarrow |\psi\rangle &= \frac{1}{2}\begin{pmatrix}1\\-1\\-1\\1\end{pmatrix}
\end{aligned} \tag{B.13}$$

That is, the two qubit system – and indeed any two qubit system – is given by a linear combination of the four basis vectors

$$\{|00\rangle, |0,1\rangle, |10\rangle, |11\rangle\} = \left\{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\right\}. \tag{B.14}$$

We can notice that a single qubit system can be described by a linear combination of two basis vectors, and that a two qubit system requires four basis vectors to describe it. In general we can say that an n-qubit system is represented by a linear combination of 2^n basis vectors.

B.3.2 *Registers*

A *register* is generally the name given to an array of controllable quantum systems; here we invoke it to mean a system of multiple qubits, specifically a subset of the total number of

available qubits. For example, a register of ten qubits can be denoted $|x[10]\rangle$, and we can think of the system as a register of six qubits together with a register of three and another register of one qubit.

$$|x[10]\rangle = |x_1[6]\rangle \otimes |x_2[3]\rangle \otimes |x_3[1]\rangle$$

B.4 ENTANGLEMENT

Another unique property of quantum systems is that of *entanglement*: when two or more particles interact in such a way that their individual quantum states can not be described independent of the other particles. A quantum state then exists for the system as a whole instead. Mathematically, we consider such entangled states as those whose state can not be expressed as a tensor product of the states of the individual qubits it's composed of: they are dependent upon the other.

To understand what we mean by this dependence, consider a counter-example. Consider the Bell state,

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right), \tag{B.15}$$

if we measure this state, we expect that it will be observed in either eigenstate $|00\rangle$ or $|11\rangle$, with equal probability due to their amplitudes' equal magnitudes. The bases for this state are simply the standard bases, $|0\rangle$ and $|1\rangle$. Thus, according to our previous definition of systems of multiple qubits, we would say this state can be given as a combination of two states, like Eq. (B.12),

$$\begin{aligned} |\Phi^+\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\ &= (a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \\ &= a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle \end{aligned} \tag{B.16}$$

However we require $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, which would imply $a_1b_2 = 0$ and $b_1a_2 = 0$. These imply that either $a_1 = 0$ or $b_2 = 0$, and also that $b_1 = 0$ or a_2=0, which are obviously invalid since we require that $a_1a_2 = b_1b_2 = \frac{1}{\sqrt{2}}$. Thus, we cannot express $|\Phi^+\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$; this inability to separate the first and second qubits is what we term *entanglement*.

B.5 UNITARY TRANSFORMATIONS

A fundamental concept in quantum mechanics is that of performing transformations on states. *Quantum transformations*, or *quantum operators*, map a quantum state into a new state within the same Hilbert space. There are certain restrictions on a physically possible quantum transformation: in order that U is a valid transformation acting on some superposition $|\psi\rangle = a_1|\psi_1\rangle + a_2|\psi_2\rangle + \ldots a_k|\psi_k\rangle$, U must be linear

$$U(a_1|\psi_1\rangle + a_2|\psi_2\rangle + \ldots a_k|\psi_k\rangle) = a_1(U|\psi_1\rangle) + a_2(U|\psi_2\rangle) + \cdots + a_k(U|\psi_k\rangle). \tag{B.17}$$

To fulfil these properties, we require that U *preserve the inner product*:

$$\langle \psi_0 | U^\dagger U | \psi \rangle = \langle \psi_0 | \psi \rangle$$

That is, we require that any such transformation be *unitary*:

$$UU^\dagger = I \Rightarrow U^\dagger = U^{-1} \tag{B.18}$$

Unitarity is a sufficient condition to describe any valid quantum operation: any quantum transformation can be described by a unitary transformation, and any unitary transformation corresponds to a physically implementable quantum transformation.

Then, if U_1 is a unitary transformation that acts on the space $\mathcal{H}_1$ and U_2 acts on $\mathcal{H}_2$, the product of the two unitary transformations is also unitary. The tensor product $U_1 \otimes U_2$ acts on the space $\mathcal{H}_1 \otimes \mathcal{H}_2$. So, then, supposing a system of two separable qubits, $|\psi_1\rangle$ and $|\psi_2\rangle$ where we wish to act on $|\psi_1\rangle$ with operator U_1 and on $|\psi_2\rangle$ with U_2, we perform it as

$$(U_1 \otimes U_2)(|\psi_1\rangle \otimes |\psi_2\rangle) = (U_1 |\psi_1\rangle) \otimes (U_2 |\psi_2\rangle) \tag{B.19}$$

B.6 DIRAC NOTATION

In keeping with standard practice, we employ *Dirac notation* throughout this thesis. Vectors are denoted by *kets* of the form $|a\rangle$. For example, the standard basis is represented by,

$$\begin{aligned} |x\rangle = |0\rangle &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ |y\rangle = |1\rangle &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \tag{B.20}$$

We saw in Table B.1 that for every such ket, $|\psi\rangle$, there exists a *dual vector*: its complex conjugate transpose, called the *bra* of such a vector, denoted $\langle \psi |$. That is,

$$\begin{aligned} \langle \psi |^\dagger &= |\psi\rangle \\ |\psi\rangle^\dagger &= \langle \psi | \end{aligned} \tag{B.21}$$

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} \Rightarrow \langle \psi | = \begin{pmatrix} \psi_1^* & \psi_2^* & \dots & \psi_n^* \end{pmatrix} \tag{B.22}$$

Then if we have two vectors $|\psi\rangle$ and $|\phi\rangle$, their *inner product* is given as $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle$.

$$\begin{aligned} |\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_n \end{pmatrix} \quad ; \quad |\phi\rangle = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_n \end{pmatrix} \\ \Rightarrow \langle\phi| = \begin{pmatrix} \phi_1^* & \phi_2^* & \phi_3^* & \dots & \phi_n^* \end{pmatrix} \\ \Rightarrow \langle\phi|\,|\psi\rangle = \begin{pmatrix} \phi_1^* & \phi_2^* & \phi_3^* & \dots & \phi_n^* \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_n \end{pmatrix} \\ \Rightarrow \langle\phi|\,|\psi\rangle = \phi_1^*\psi_1 + \phi_2^*\psi_2 + \phi_3^*\psi_3 + \cdots + \phi_n^*\psi_n \end{aligned} \tag{B.23}$$

Example B.6.1.

$$\begin{aligned} |\psi\rangle = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad ; \quad |\phi\rangle = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \\ \Rightarrow \langle\phi|\,|\psi\rangle = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \\ = (4)(1) + (5)(2) + (6)(3) = 32 \end{aligned} \tag{B.24}$$

Similarly, their *outer product* is given as $|\phi\rangle\langle\psi|$. Multiplying a column vector by a row vector thus gives a matrix. Matrices generated by a outer products then define operators:

Example B.6.2.

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 3 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 6 & 8 \end{pmatrix} \tag{B.25}$$

Then we can say, for $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$$|0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{B.26a}$$

$$|0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{B.26b}$$

$$|1\rangle\langle 0| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \tag{B.26c}$$

$$|1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{B.26d}$$

And so any 2-dimensional linear transformation in the standard basis $|0\rangle, |1\rangle$ can be given as a sum

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = a|0\rangle\langle 0| + b|0\rangle\langle 1| + c|1\rangle\langle 0| + d|1\rangle\langle 1| \tag{B.27}$$

This is a common method of representing operators as outer products of vectors. A transformation that *exchanges* a particle between two states, say $|0\rangle \leftrightarrow |1\rangle$ is given by the operation

$$\hat{Q} : \begin{cases} |0\rangle \to |1\rangle \\ |1\rangle \to |0\rangle \end{cases}$$

Which is equivalent to the outer product representation

$$\hat{Q} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

For clarity, here we will prove this operation

Example B.6.3.

$$\begin{aligned} \hat{Q} &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{aligned}$$

So then, acting on $|0\rangle$ *and* $|1\rangle$ *gives*

$$\hat{Q}|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

$$\hat{Q}\left|1\right\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \left|0\right\rangle$$

To demonstrate how Dirac notation simplifies this:

$$\begin{aligned} \hat{Q}\left|0\right\rangle &= \left(\left|0\right\rangle\left\langle 1\right| + \left|1\right\rangle\left\langle 0\right|\right)\left|0\right\rangle \\ &= \left|0\right\rangle\left\langle 1\right|\left|0\right\rangle + \left|1\right\rangle\left\langle 0|0\right\rangle \\ &= \left|0\right\rangle\left\langle 1|0\right\rangle + \left|1\right\rangle\left\langle 0|0\right\rangle \end{aligned}$$

Then, since $\left|0\right\rangle$ and $\left|1\right\rangle$ are orthogonal basis, their inner product is 0 and the inner product of a vector with itself is 1, ($\left\langle 1|1\right\rangle = \left\langle 0|0\right\rangle = 1, \left\langle 0|1\right\rangle = \left\langle 1|0\right\rangle = 0$). So,

$$\begin{aligned} \hat{Q}\left|0\right\rangle &= \left|0\right\rangle(0) + \left|1\right\rangle(1) \\ \Rightarrow \hat{Q}\left|0\right\rangle &= \left|1\right\rangle \end{aligned} \tag{B.28}$$

And similarly for $\hat{Q}\left|1\right\rangle$. This simple example then shows why Dirac notation can significantly simplify calculations across quantum mechanics, compared to standard matrix and vector notation. To see this more clearly, we will examine a simple 2-qubit state under such operations. The method generalises to operating on two or more qubits generically: we can define any operator which acts on two qubits as a sum of outer products of the basis vectors $\left|00\right\rangle, \left|01\right\rangle, \left|10\right\rangle$ and $\left|11\right\rangle$. We can similarly define any operator which acts on an n qubit state as a linear combination of the 2^n basis states generated by the n qubits.

Example B.6.4. *To define a transformation that will exchange basis vectors $\left|00\right\rangle$ and $\left|11\right\rangle$, while leaving $\left|01\right\rangle$ and $\left|10\right\rangle$ unchanged (ie exchanging $\left|01\right\rangle \leftrightarrow \left|01\right\rangle, \left|10\right\rangle \leftrightarrow \left|10\right\rangle$) we define an operator*

$$\hat{Q} = \left|00\right\rangle\left\langle 11\right| + \left|11\right\rangle\left\langle 00\right| + \left|10\right\rangle\left\langle 10\right| + \left|01\right\rangle\left\langle 01\right| \tag{B.29}$$

Then, using matrix calculations this would require separately calculating the four outer products in the above sum and adding them to find a 4×4 matrix to represent $\hat{Q}$, which then acts on a state $\left|\psi\right\rangle$. Instead, consider first that $\left|\psi\right\rangle = \left|00\right\rangle$, ie one of the basis vectors our transformation is to change:

$$\hat{Q}\left|00\right\rangle = \left(\left|00\right\rangle\left\langle 11\right| + \left|11\right\rangle\left\langle 00\right| + \left|10\right\rangle\left\langle 10\right| + \left|01\right\rangle\left\langle 01\right|\right)\ \left|00\right\rangle \tag{B.30}$$

And as before, only the inner products of a vector with itself remains:

$$\begin{aligned} &= \left|00\right\rangle\left\langle 11\left|00\right.\right\rangle + \left|11\right\rangle\left\langle 00\left|00\right.\right\rangle + \left|10\right\rangle\left\langle 10\left|00\right.\right\rangle + \left|01\right\rangle\left\langle 01\left|00\right.\right\rangle \\ &= \left|00\right\rangle(0) + \left|11\right\rangle(1) + \left|10\right\rangle(0) + \left|01\right\rangle(0) \\ \Rightarrow &\hat{Q}\left|00\right\rangle = \left|11\right\rangle \end{aligned} \tag{B.31}$$

i.e the transformation has performed $\hat{Q} : |00\rangle \rightarrow |11\rangle$ *as expected. Then, if we apply the same transformation to a state which does not depend on one of the target states, eg,*

$$\begin{aligned} |\psi\rangle =& a\,|10\rangle + b\,|01\rangle \\ \hat{Q}\,|\psi\rangle =& \Big(|00\rangle\langle 11| + |11\rangle\langle 00| + |10\rangle\langle 10| + |01\rangle\langle 01| \Big)\Big(a\,|10\rangle + b\,|01\rangle \Big) \\ =& a\Big(|00\rangle\langle 11|10\rangle + |11\rangle\langle 00|10\rangle + |10\rangle\langle 10|10\rangle + |01\rangle\langle 01|10\rangle \Big) \\ &+ b\Big(|00\rangle\langle 11|01\rangle + |11\rangle\langle 00|01\rangle + |10\rangle\langle 10|01\rangle + |01\rangle\langle 01|01\rangle \Big) \end{aligned} \tag{B.32}$$

And since the inner product is a scalar, we can factor terms such as $\langle 11|10\rangle$ *to the beginning of expressions, eg* $|00\rangle\langle 11|\,|10\rangle = \langle 11\,|10\rangle\,|00\rangle$*, and we also know*

$$\begin{aligned} \langle 11|10\rangle = \langle 00|10\rangle = \langle 01|10\rangle = \langle 11|01\rangle = \langle 00|01\rangle = \langle 10|01\rangle = 0 \\ \langle 10|10\rangle = \langle 01|01\rangle = 1 \end{aligned} \tag{B.33}$$

We can express the above as

$$\begin{aligned} \hat{Q}\,|\psi\rangle =& a\Big((0)\,|00\rangle + (0)\,|11\rangle + (1)\,|10\rangle + (0)\,|01\rangle \Big) \\ &+ b\Big((0)\,|00\rangle + (0)\,|11\rangle + (0)\,|10\rangle + (1)|01\rangle \Big) \\ =& a\,|10\rangle\,| + b|01\rangle \\ =& |\psi\rangle \end{aligned} \tag{B.34}$$

Then it is clear that, when $|\psi\rangle$ *is a superposition of states unaffected by transformation* $\hat{Q}$*, then* $\hat{Q}\,|\psi\rangle = |\psi\rangle$*.*

This method generalises to systems with greater numbers of particles (qubits). If we briefly consider a 3 qubit system - and initialise all qubits in the standard basis state $|0\rangle$ - then the system is represented by $|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. This quantity is an 8-row vector. To calculate the outer product $\langle 000|000\rangle$, we would be multiplying an 8-column bra $\langle 000|$ by an 8-row ket $|000\rangle$. Clearly then we will be working with 8×8 matrices, which will become quite difficult to maintain effectively and efficiently quite fast. As we move to systems of larger size, standard matrix multiplication becomes impractical for hand-written analysis, although of course remains tractable computationally up to $n \sim 10$ qubits. It is obvious that Dirac's bra/ket notation is a helpful, pathematically precise tool for QM.

C

EXAMPLE EXPLORATION STRATEGY RUN

Here we provide a complete example of how to run the Quantum Model Learning Agent (QMLA) framework, including how to implement a custom exploration strategy (ES), and generate/interpret analysis. Note: these examples are included in the QMLA documentation in a format that may be easier to follow – where possible, we recommend readers follow the Tutorial section of [21].

First, *fork* the QMLA codebase from [22] to a Github user account (referred to as username in Listing C.1). Now, we must download the code base and ensure it runs properly; these instructions are implemented via the command line[1].

The steps of preparing the codebase are

1. install redis;
2. create a virtual Python environment for installing QMLA dependencies without damaging other parts of the user's environment;
3. download the QMLA codebase from the forked Github repository;
4. install packages upon which QMLA depends.

```
# Install redis (database broker)
sudo apt update
sudo apt install redis-server

# make directory for QMLA
cd
mkdir qmla_test
cd qmla_test

# make Python virtual environment for QMLA
# note: change Python3.6 to desired version
sudo apt-get install python3.6-venv
python3.6 -m venv qmla-env
source qmla-env/bin/activate
```

1 Note: these instructions are tested for Linux and presumed to work on Mac, but untested on Windows. It is likely some of the underlying software (redis servers) can not be installed on Windows, so running on *Windows Subsystem for Linux* is advised.

```
# Download QMLA
git clone --depth 1 https://github.com/username/QMLA.git #
    REPLACE username

# Install dependencies
cd QMLA
pip install -r requirements.txt
```

Listing C.1: QMLA codebase setup language

Note there may be a problem with some packages in the requirements.txt arising from the attempt to install them all through a single call to pip install. Ensure these are all installed before proceeding.

When all of the requirements are installed, test that the framework runs. QMLA uses redis databases to store intermittent data: we must manually initialise the database. Run the following (note: here we list redis-4.0.8, but this must be corrected to reflect the version installed on the user's machine in the above setup section):

```
~/redis-4.0.8/src/redis-server
```

Listing C.2: Launch redis database

which should give something like Fig. C.1.

Figure C.1: Terminal running redis-server.

In a text editor, open qmla_test/QMLA/launch/local_launch.sh; here we will ensure that we are running the QHL algorithm, with 5 experiments and 20 particles, on the ES named TestInstall. Ensure the first few lines of local_launch.sh read:

```
#!/bin/bash

##### ---------------------------------------------- #####
# QMLA run configuration
##### ---------------------------------------------- #####
num_instances=2 # number of instances in run
run_qhl=0 # perform QHL on known (true) model
run_qhl_multi_model=0 # perform QHL for defined list of models
experiments=2 # number of experiments
particles=10 # number of particles
plot_level=5


##### ---------------------------------------------- #####
# Choose an exploration strategy
# This will determine how QMLA proceeds.
##### ---------------------------------------------- #####
exploration_strategy="TestInstall"
```

Listing C.3: local_launch script

Ensure the terminal running redis is kept active, and open a separate terminal window. We must activate the Python virtual environment configured for QMLA, which we set up in Listing C.1. Then, we navigate to the QMLA directory, and launch:

```
# activate the QMLA Python virtual environment
source qmla_test/qmla-env/bin/activate

# move to the QMLA directory
cd qmla_test/QMLA
# Run QMLA
cd launch
./local_launch.sh
```

Listing C.4: Launch QMLA

There may be numerous warnings, but they should not affect whether QMLA has succeeded; QMLA will raise any significant error. Assuming the run has completed successfully, QMLA stores the run's results in a subdirectory named by the date and time it was started. For example, if the run was initialised on January 1st at 01:23, navigate to the corresponding directory by

```
cd results/Jan_01/01_23
```

Listing C.5: QMLA results directory

For now it is sufficient to notice that the code has run successfully: it should have generated (in results/Jan_01/01_23) files like storage_001.p and results_001.p.

C.1 CUSTOM EXPLORATION STRATEGY

Next, we design a basic ES, for the purpose of demonstrating how to run the algorithm. ESs are placed in the directory qmla/exploration_strategies. To make a new one, navigate to the exploration strategies directory, make a new subdirectory, and copy the template file.

```
cd ~/qmla_test/QMLA/exploration_strategies/
mkdir custom_es

# Copy template file into example
cp template.py custom_es/example.py
cd custom_es
```

Listing C.6: QMLA codebase setup

Ensure QMLA will know where to find the ES by importing everything from the custom ES directory into to the main exploration_strategy module. Then, in the custom_es directory, make a file called __init__.py which imports the new ES from the example.py file. To add any further ESs inside the directory custom_es, include them in the custom __init__.py, and they will automatically be available to QMLA.

```
# inside qmla/exploration_strategies/custom_es
#   __init__.py
from qmla.exploration_strategies.custom_es.example import *

# inside qmla/exploration_strategies , add to the existing
# __init__.py
from qmla.exploration_strategies.custom_es import *
```

Listing C.7: Providing custom exploration strategy to QMLA

Now, change the structure (and name) of the ES inside custom_es/example.py. Say we wish to target the true model

$$\begin{aligned} \vec{\alpha} &= (\alpha_{1,2} \quad \alpha_{2,3} \quad \alpha_{3,4}) \\ \vec{T} &= \begin{pmatrix} \hat{\sigma}_z^1 \otimes \hat{\sigma}_z^2 \\ \hat{\sigma}_z^2 \otimes \hat{\sigma}_z^3 \\ \hat{\sigma}_z^3 \otimes \hat{\sigma}_z^4 \end{pmatrix} \\ \implies \hat{H}_0 &= \hat{\sigma}_z^{(1,2)} \hat{\sigma}_z^{(2,3)} \hat{\sigma}_z^{(3,4)} \end{aligned} \tag{C.1}$$

QMLA interprets models as strings, where terms are separated by $+$, and parameters are implicit. So the target model in Eq. (C.1) will be given by

pauliSet_1J2_zJz_d4+pauliSet_2J3_zJz_d4+pauliSet_3J4_zJz_d4.

Adapting the template ES slightly, we can define a model generation strategy with a small number of hard coded candidate models introduced at the first branch of the exploration tree. We will also set the parameters of the terms which are present in $\hat{H}_0$, as well as the range in which to search parameters. Keeping the imports at the top of the example.py, rewrite the ES as:

```
class ExampleBasic(
    exploration_strategy.ExplorationStrategy
):

    def __init__(
        self,
        exploration_rules,
        true_model=None,
        **kwargs
    ):
        self.true_model = 'pauliSet_1J2_zJz_d4+
            pauliSet_2J3_zJz_d4+pauliSet_3J4_zJz_d4'
        super().__init__(
            exploration_rules=exploration_rules,
            true_model=self.true_model,
            **kwargs
        )

        self.initial_models = None
        self.true_model_terms_params = {
            'pauliSet_1J2_zJz_d4' : 2.5,
```

```
            'pauliSet_2J3_zJz_d4' : 7.5,
            'pauliSet_3J4_zJz_d4' : 3.5,
        }
        self.tree_completed_initially = True
        self.min_param = 0
        self.max_param = 10

    def generate_models(self, **kwargs):

        self.log_print(["Generating models; spawn step {}".format
            (self.spawn_step)])
        if self.spawn_step == 0:
            # chains up to 4 sites
            new_models = [
                'pauliSet_1J2_zJz_d4',
                'pauliSet_1J2_zJz_d4+pauliSet_2J3_zJz_d4',
                'pauliSet_1J2_zJz_d4+pauliSet_2J3_zJz_d4+
                    pauliSet_3J4_zJz_d4',
            ]
            self.spawn_stage.append('Complete')

        return new_models
```

Listing C.8: ExampleBasic exploration strategy.

To run[2] the example ES for a meaningful test, return to the local_launch of Listing C.3, but change some of the settings:

```
particles=2000
experiments=500
run_qhl=1
exploration_strategy=ExampleBasic
```

Listing C.9: local_launch configuration for QHL.

Run locally again as in Listing C.4; then move to the results directory as in Listing C.5.

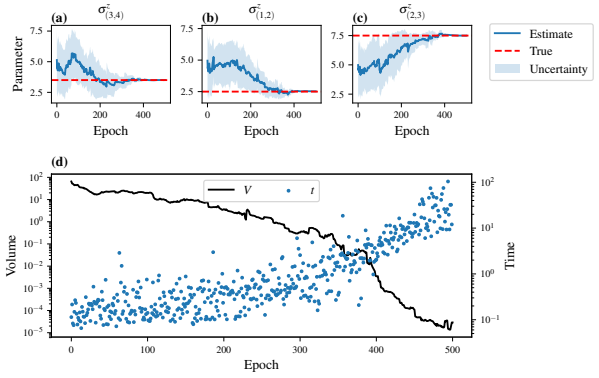
2 Note this will take up to 15 minutes to run. This can be reduced by lowering the values of particles, experiments, which is sufficient for testing but note that the outcomes will be less effective than those presented in the figures of this section.

C.2 ANALYSIS

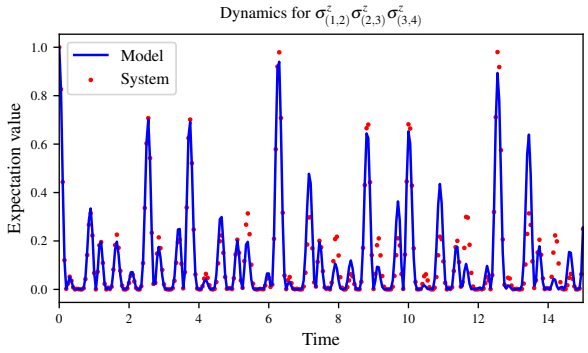
QMLA stores results and generates plots over the entire range of the algorithm[3], i.e. the run, instance and models. The depth of analysis performed automatically is set by the user control plot_level in local_launch.sh; for plot_level=1, only the most crucial figures are generated, while plot_level=6 generates plots for every individual model considered. For model searches across large model spaces and/or considering many candidates, excessive plotting can cause considerable slow-down, so users should be careful to generate plots only to the degree they will be useful. Next we show some examples of the available plots.

C.2.1 *Model analysis*

We have just run quantum Hamiltonian learning (QHL) for the model in Eq. (C.1) for a single instance, using a reasonable number of particles and experiments, so we expect to have trained the model well. Instance-level results are stored (e.g. for the instance with qmla_id=1) in Jan_01/01_23/instances/qmla_1. Individual models' insights can be found in model_training, e.g. the model's learning_summary Fig. C.2a, and dynamics in Fig. C.2b.



(a)



(b)

Figure C.2: Model analysis plots, stored in (for example) Jan_01/01_23/instances/qmla_1/model_training. **a** learning_summary_1. Displays the outcome of QHL for the given model: Subfigures (a)-(c) show the estimates of the parameters; (d) shows the total parameterisation volume against experiments trained upon, along with the evolution times used for those experiments. **(b)** dynamics_1 The model's attempt at reproducing dynamics from $\hat{H}_0$.

3 Recall that a single implementation of QMLA is called an instance, while a series of instances – which share the same target model – is called the run.

C.2.2 *Instance analysis*

Now we can run the full QMLA algorithm, i.e. train several models and determine the most suitable. QMLA will call the generate_models method of the ExampleBasic ES, set in Listing C.8, which tells QMLA to construct three models on the first branch, then terminate the search. Here we need to train and compare all models so it takes considerably longer to run: for the purpose of testing, we reduce the resources so the entire algorithm runs in about 15 minutes. Some applications will require significantly more resources to learn effectively. In realistic cases, these processes are run in parallel, as we will cover in Appendix C.3.

Reconfigure a subset of the settings in the local_launch.sh script (Listing C.3) and run it again:

```
experiments=250
particles=1000
run_qhl=0
exploration_strategy=ExampleBasic
```

Listing C.10: local_launch configuration for QMLA.

In the corresponding results directory, navigate to instances/qmla_1, where instance level analysis are available.

```
cd results/Jan_01/01_23/instances/qmla_1
```

Listing C.11: Navigating to instance results.

Figures of interest here show the composition of the models (Fig. C.3a), as well as the Bayes factors between candidates (Fig. C.3b). Individual model comparisons – i.e. Bayes factor (BF) – are shown in Fig. C.3c, with the dynamics of all candidates shown in Fig. C.4c. The probes used during the training of all candidates are also plotted (Fig. C.3e).

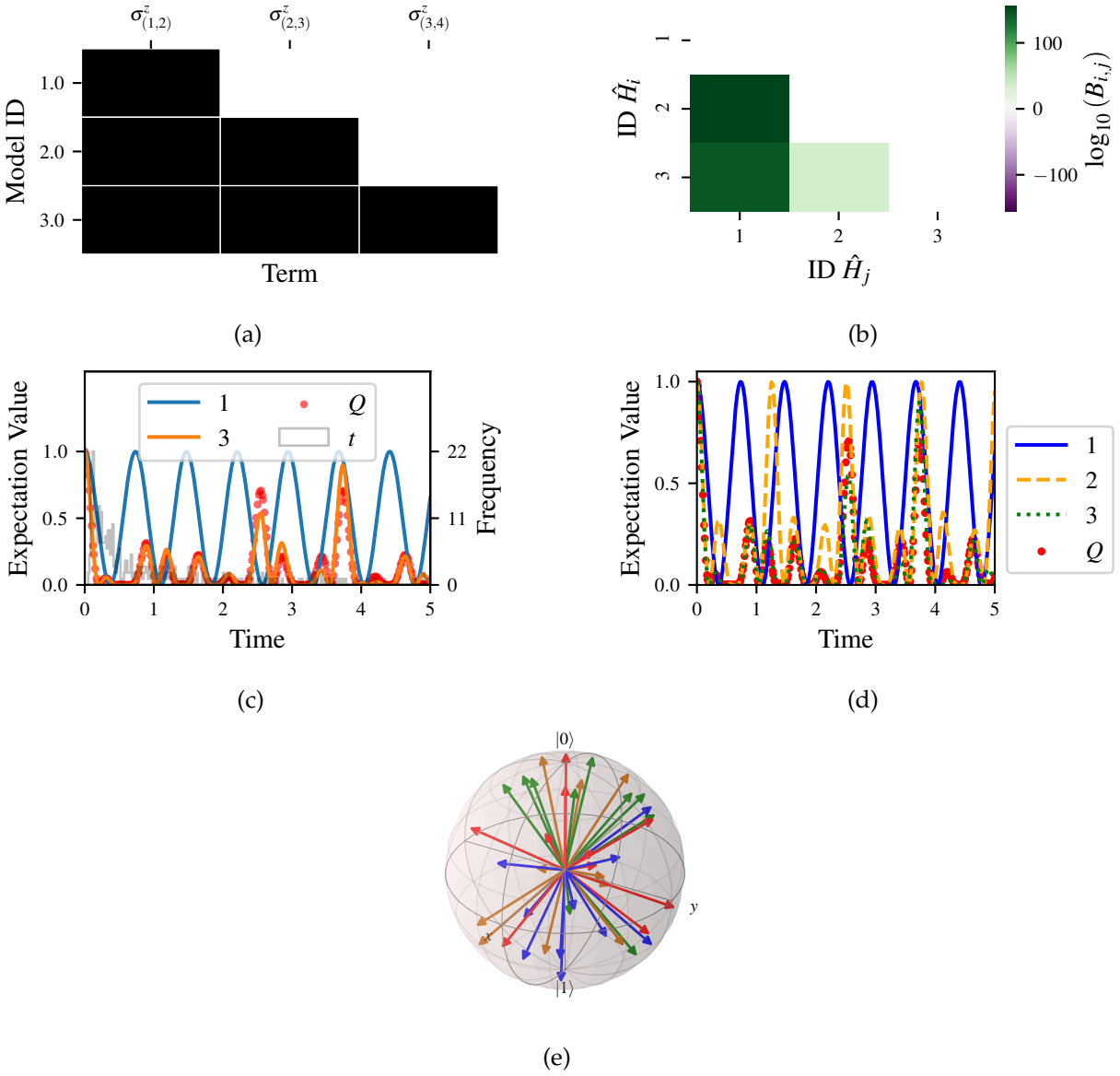


Figure C.3: QMLA plots; found within instance directory e.g. Jan_01/01_23/instances/qmla_1, and its subdirectories. **(a)** composition_of_models: constituent terms of all considered models, indexed by their model IDs. Here model 3 is $\hat{H}_0$. **(b)** bayes_factors: Bayes factor (BF) comparisons between all models. BFs are read as $B_{i,j}$ where i is the model with lower ID, e.g. $B_{1,2}$ rather than $B_{2,1}$. Thus $B_{ij} > 0$ (< 0) indicates $\hat{H}_i$ ($\hat{H}_j$), i.e. the model on the y-axis (x-axis) is the stronger model. **(c)** comparisons/BF_1_3: direct comparison between models with IDs 1 and 3, showing their reproduction of the system dynamics (red dots, Q), as well as the times (experiments) against which the BF was calculated. **(d)** branches/dynamics_branch_1: dynamics of all models considered on the branch compared with system dynamics (red dots, Q). **(e)** probes_bloch_sphere: probes used for training models in this instance (only showing 1-qubit versions).

C.2.3 *Run analysis*

Considering a number of instances together is a *run*. In general, this is the level of analysis of most interest: an individual instance is liable to errors due to the probabilistic nature of the model training and generation subroutines. On average, however, we expect those elements to perform well, so across a significant number of instances, we expect the average outcomes to be meaningful.

Each results directory has an analyse.sh script to generate plots at the run level.

```
cd results/Jan_01/01_23
./analyse.sh
```

Listing C.12: Analysing QMLA run.

Run level analysis are held in the main results directory and several sub-directories created by the analyse script. Here, we recommend running a number of instances with very few resources so that the test finishes quickly[4]. The results will therefore be meaningless, but allow fo elucidation of the resultant plots. First, reconfigure some settings of Listing C.3 and launch again.

```
num_instances=10
experiments=20
particles=100
run_qhl=0
exploration_strategy=ExampleBasic
```

Listing C.13: local_launch configuration for QMLA run.

Some of the generated analysis are shown in Figs. C.4 to C.5. The number of instances for which each model was deemed champion, i.e. their *win rates* are given in Fig. C.4a. The *top models*, i.e. those with highest win rates, analysed further: the average parameter estimation progression for $\hat{H}_0$ – including only the instances where $\hat{H}_0$ was deemed champion – are shown in Fig. C.4b. Irrespecitve of the champion models, the rate with which each term is found in the champion model ($\hat{t} \in \hat{H}'$) indicates the likelihood that the term is really present; these rates – along with the parameter values learned – are shown in Fig. C.4c. The champion model from each instance can attempt to reproduce system dynamics: we group together these reproductions for each model in Fig. C.5.

4 This run will take about ten minutes

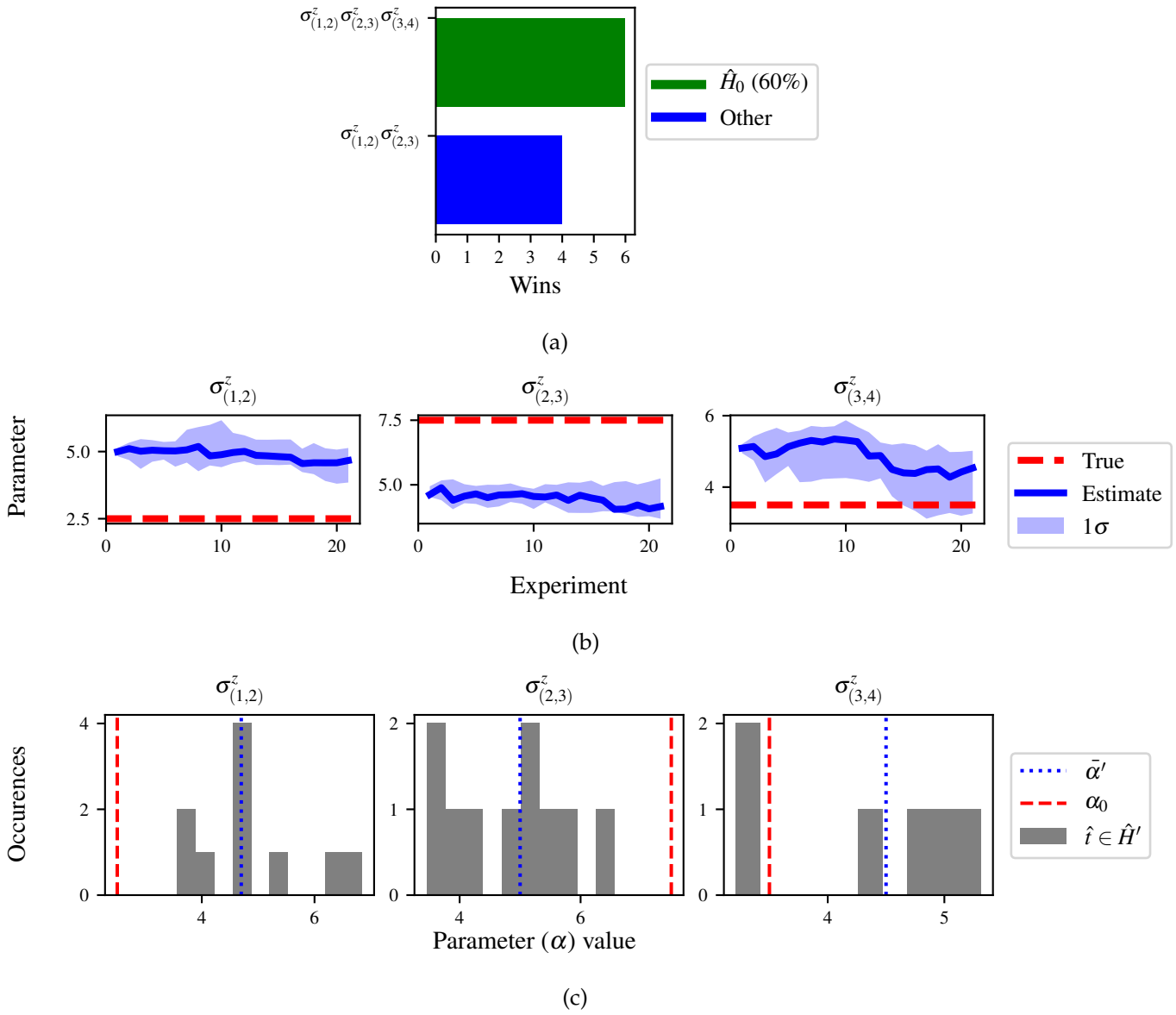


Figure C.4: QMLA run plots; found within run directory e.g. Jan_01/01_23/. **(a)** performance/model_wins: number of instance wins achieved by each model. **(b)** champion_models/params_params_pauliSet_1J2_zJz_d4+pauliSet_2J3_zJz_d4+pauliSet_3J4_zJz_d4: parameter estimation progression for the true model , only for the instances where it was deemed champion. **(c)** champion_models/terms_and_params: histogram of parameter values found for each term which appears in any champion model, with the true parameter (α_0) in red and the median learned parameter ($\bar{\alpha}'$) in blue.

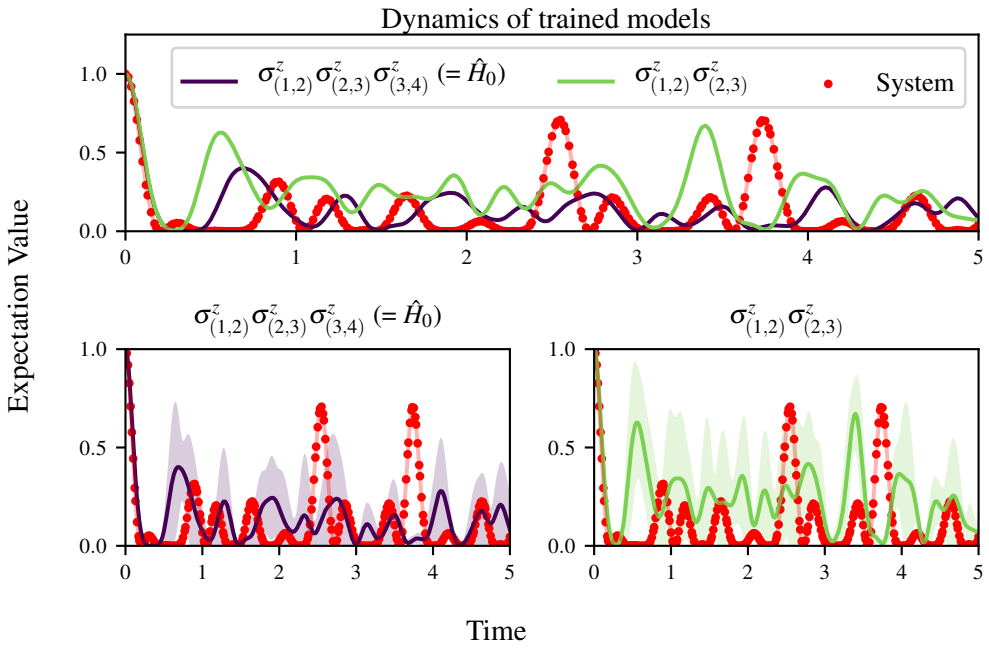


Figure C.5: Run plot performace/dynamics: median dynamics of the champion models. The models which won most instances are shown together in the top panel, and individually in the lower panels. The median dynamics from the models' learnings in its winning instances are shown, with the shaded region indicating the 66% confidence region.

C.3 PARALLEL IMPLEMENTATION

We provide utility to run QMLA on parallel processes. Individual models' training can run in parallel, as well as the calculation of BF between models. The provided script is designed for portable batch system (PBS) job scheduler running on a compute cluster. It will require a few adjustments to match the system being used. Overall, though, it has mostly a similar structure as the local_launch.sh script used above.

QMLA must be downloaded on the compute cluster as in Listing C.1; this can be a new fork of the repository, though it is sensible to test installation locally as described in this chapter so far, then *push* that version, including the new ES, to Github, and cloning the latest version. It is again advisable to create a Python virtual environment in order to isolate QMLA and its dependencies[5]. Open the parallel launch script, QMLA/launch/parallel_launch.sh, and prepare the first few lines as

```
#!/bin/bash
```

5 Indeed it is sensible to do this for any Python development project.

```
##### ------------------------------------------------ #####
# QMLA run configuration
##### ------------------------------------------------ #####
num_instances=10 # number of \glspl{instance} in run
run_qhl=0 # perform QHL on known (true) model
run_qhl_multi_model=0 # perform QHL for defined list of models
experiments=250
particles=1000
plot_level=5


##### ------------------------------------------------ #####
# Choose an exploration strategy
# This will determine how QMLA proceeds.
##### ------------------------------------------------ #####
exploration_strategy="ExampleBasic"
```

Listing C.14: parallel_launch script

When submitting jobs to schedulers like PBS, we must specify the time required, so that it can determine a fair distribution of resources among users. We must therefore *estimate* the time it will take for an instance to complete: clearly this is strongly dependent on the numbers of experiments (N_e) and particles (N_p), and the number of models which must be trained. QMLA attempts to determine a reasonable time to request based on the max_num_models_by_shape attribute of the ES, by calling QMLA/scripts/time_required_calculation.py. In practice, this can be difficult to set perfectly, so the timing_insurance_factor attribute of the ES can be used to correct for heavily over- or under-estimated time requests. Instances are run in parallel, and each instance trains/compares models in parallel. The number of processes to request, N_c for each instance is set as num_processes_to_parallelise_over in the ES. Then, if there are N_r instances in the run, we will be requesting the job scheduler to admit N_r distinct jobs, each requiring N_c processes, for the time specified.

The parallel_launch script works together with launch/run_single_qmla_instance.sh, though note a number of steps in the latter are configured to the cluster and may need to be adapted. In particular, the first command is used to load the redis utility, and later lines are used to initialise a redis server. These commands will probably not work with most machines, so must be configured to achieve those steps.

```
module load tools/redis-4.0.8

...
```

```
SERVER_HOST=$(head -1 "$PBS_NODEFILE")
let REDIS_PORT="6300 + $QMLA_ID"

cd $LIBRARY_DIR
redis-server RedisDatabaseConfig.conf --protected-mode no --port
    $REDIS_PORT &
redis-cli -p $REDIS_PORT flushall
```

Listing C.15: run_single_qmla_instance script

When the modifications are finished, QMLA can be launched in parallel similarly to the local version:

```
source qmla_test/qmla-env/bin/activate

cd qmla_test/QMLA/launch
./parallel_launch.sh
```

Listing C.16: run_single_qmla_instance script

Jobs are likely to queue for some time, depending on the demands on the job scheduler. When all jobs have finished, results are stored as in the local case, in QMLA/launch/results/-Jan_01/01_23, where analyse.sh can be used to generate a series of automatic analyses.

C.4 CUSTOMISING EXPLORATION STRATEGYS

User interaction with the QMLA codebase should be achieveable primarily through the exploration strategy (ES) framework. Throughout the algorithm(s) available, QMLA calls upon the ES before determining how to proceed. The usual mechanism through which the actions of QMLA are directed, is to set attributes of the ES class: the complete set of influential attributes are available at [21].

QMLA directly uses several methods of the ES class, all of which can be overwritten in the course of customising an ES. Most such methods need not be replaced, however, with the exception of generate_models, which is the most important aspect of any ES: it determines which models are built and tested by QMLA. This method allows the user to impose any logic desired in constructing models; it is called after the completion of every branch of the exploration tree on the ES.

C.4.1 *Greedy search*

A first non-trivial ES is to build models greedily from a set of *primitive* terms, $\mathcal{T} = \{\hat{t}\}$. New models are constructed by combining the previous branch champion with each of the remaining, unused terms. The process is repeated until no terms remain.

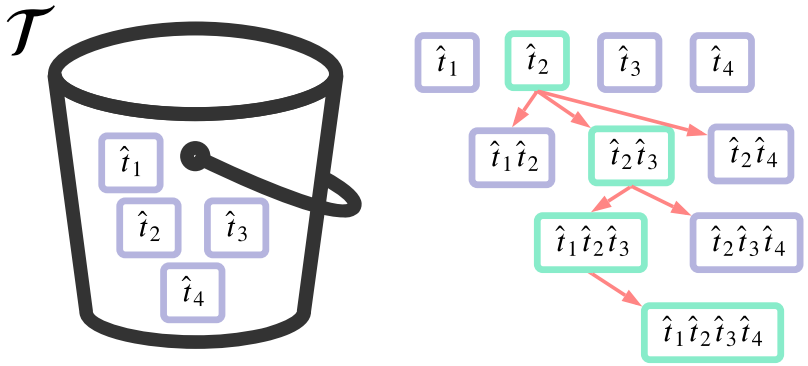


Figure C.6: Greedy search mechanism. **Left**, a set of primitive terms, $\mathcal{T}$, are defined in advance. **Right**, models are constructed from $\mathcal{T}$. On the first branch, the primitve terms alone constitute models. Thereafter, the strongest model (marked in green) from the previous branch is combined with all the unused terms.

We can compose an ES using these rules, say for

$$\mathcal{T} = \left\{\hat{\sigma}_x^1,\ \hat{\sigma}_y^1,\ \hat{\sigma}_x^1 \otimes \hat{\sigma}_x^2,\ \hat{\sigma}_y^1 \otimes \hat{\sigma}_y^2\right\}$$

as follows. Note the termination criteria must work in conjunction with the model generation routine. Users can overwrite the method check_tree_completed for custom logic, although a straightforward mechanism is to use the spawn_stage attribute of the ES class: when the final element of this list is Complete, QMLA will terminate the search by default. Also note that the default termination test checks whether the number of branches (spawn_step) exceeds the limit max_spawn_depth, which must be set artifically high to avoid ceasing the search too early, if relying solely on spawn_stage. Here we demonstrate how to impose custom logic to terminate the seach also.

```
class ExampleGreedySearch(
    exploration_strategy.ExplorationStrategy
):
    r"""
    From a fixed set of terms, construct models iteratively,
```

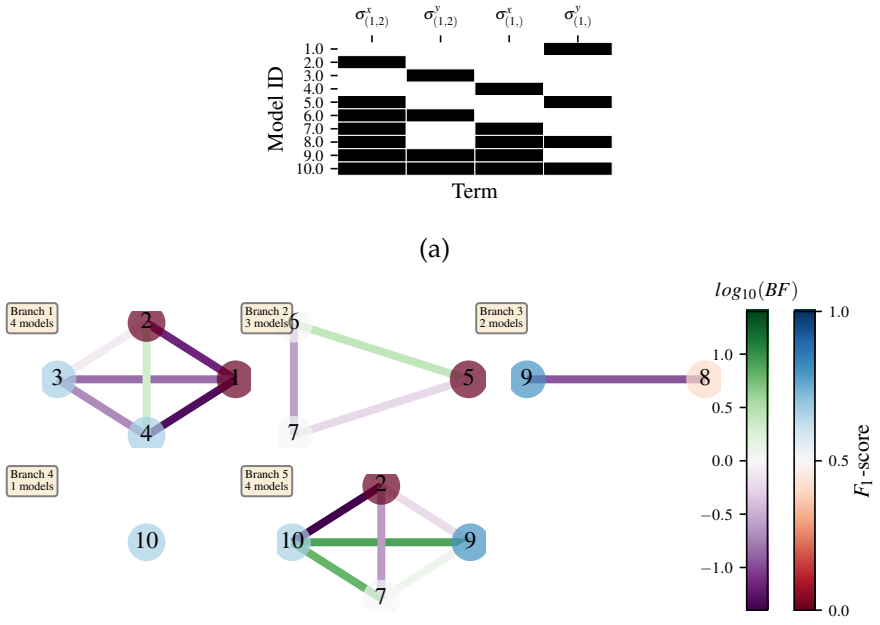
```
    greedily adding all unused terms to separate models at each
       call to the generate_models.

    """

    def __init__(
        self,
        exploration_rules,
        **kwargs
    ):

        super().__init__(
            exploration_rules=exploration_rules,
            **kwargs
        )
        self.true_model = 'pauliSet_1_x_d3+pauliSet_1J2_yJy_d3+
           pauliSet_1J2J3_zJzJz_d3'
        self.initial_models = None
        self.available_terms = [
            'pauliSet_1_x_d3', 'pauliSet_1_y_d3',
            'pauliSet_1J2_xJx_d3', 'pauliSet_1J2_yJy_d3'
        ]
        self.branch_champions = []
        self.prune_completed_initially = True
        self.check_champion_reducibility = False

    def generate_models(
        self,
        model_list,
        **kwargs
    ):
        self.log_print([
            "Generating models in tiered greedy search at spawn
               step {}.".format(
                 self.spawn_step,
            )
        ])
        try:
            previous_branch_champ = model_list[0]
            self.branch_champions.append(previous_branch_champ)
```

```
        except:
            previous_branch_champ = ""

        if self.spawn_step == 0 :
            new_models = self.available_terms
        else:
            new_models = greedy_add(
                current_model = previous_branch_champ,
                terms = self.available_terms
            )

        if len(new_models) == 0:
            # Greedy search has exhausted the available models;
            # send back the list of branch champions and
               terminate search.
            new_models = self.branch_champions
            self.spawn_stage.append('Complete')

        return new_models

def greedy_add(
    current_model,
    terms,
):
    r"""
    Combines given model with all terms from a set.

    Determines which terms are not yet present in the model,
    and adds them each separately to the current model.

    :param str current_model: base model
    :param list terms: list of strings of terms which are to be
       added greedily.
    """


    try:
        present_terms = current_model.split('+')
    except:
        present_terms = []
    nonpresent_terms = list(set(terms) - set(present_terms))
```

```
    term_sets = [
        present_terms+[t] for t in nonpresent_terms
    ]

    new_models = ["+".join(term_set) for term_set in term_sets]

    return new_models
```

Listing C.17: ExampleGreedySearch exploration stategy

This run can be implemented locally or in parallel as described above[6], and analysed as in Listing C.12, generating figures in accordance with the plot_level set by the user in the launch script. Outputs can again be found in the instances subdirectory, including a map of the models generated, as well as the branches they reside on, and the BFs between candidates, Fig. C.7.



(a)

(b)

Figure C.7: Greedy exploration strategy. **(a)**, composition_of_models. **b**, graphs_of_branches_ExampleGreedySearch: shows which models reside on each branches of the exploration tree. Models are coloured by their F_1-score, and edges represent the BF between models. The first four branches are equivalent to those in Fig. C.6 , while the final branch considers the set of branch champions, in order to determine the overall champion.

6 We advise reducing plot_level to 3 to avoid excessive/slow figure generation.

C.4.2 *Tiered greedy search*

We provide one final example of a non-trivial ES: tiered greedy search. Similar to the idea of Appendix C.4.1, except terms are introduced hierarchically: sets of terms $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n$ are each examined greedily, where the overall strongest model of one tier forms the seed model for the subsequent tier. This is depicted in the main text in Fig. 1.4. A corresponding ES is given as follows.

```
class ExampleGreedySearchTiered(
    exploration_strategy.ExplorationStrategy
):
    r"""
    Greedy search in tiers.

    Terms are batched together in tiers;
    tiers are searched greedily;
    a single tier champion is elevated to the subsequent tier.

    """

    def __init__(
        self,
        exploration_rules,
        **kwargs
    ):
        super().__init__(
            exploration_rules=exploration_rules,
            **kwargs
        )
        self.true_model = 'pauliSet_1_x_d3+pauliSet_1J2_yJy_d3+
           pauliSet_1J2J3_zJzJz_d3'
        self.initial_models = None
        self.term_tiers = {
            1 : ['pauliSet_1_x_d3', 'pauliSet_1_y_d3', '
               pauliSet_1_z_d3' ],
            2 : ['pauliSet_1J2_xJx_d3', 'pauliSet_1J2_yJy_d3', '
               pauliSet_1J2_zJz_d3'],
```

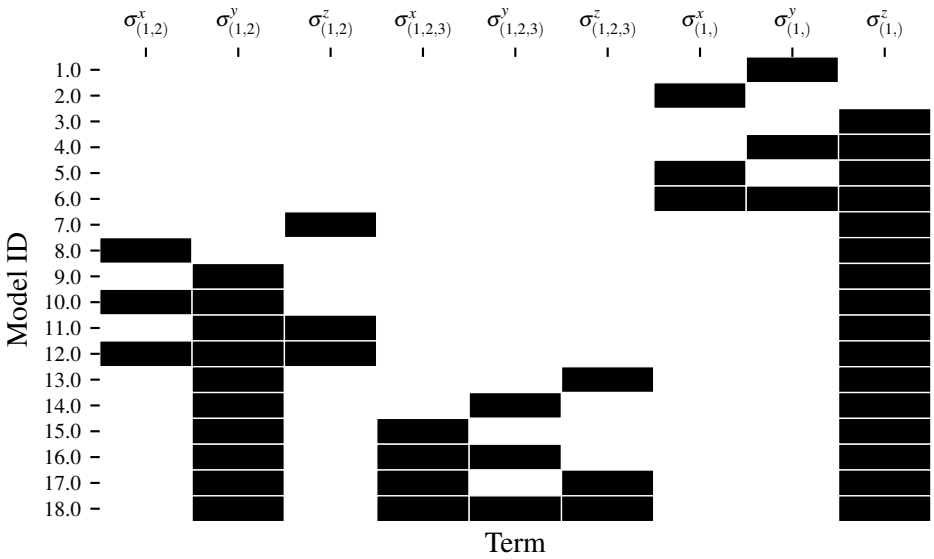
```
            3 : ['pauliSet_1J2J3_xJxJx_d3', '
                pauliSet_1J2J3_yJyJy_d3', 'pauliSet_1J2J3_zJzJz_d3
                '],
        }
        self.tier = 1
        self.max_tier = max(self.term_tiers)
        self.tier_branch_champs = {k : [] for k in self.
            term_tiers}
        self.tier_champs = {}
        self.prune_completed_initially = True
        self.check_champion_reducibility = True

    def generate_models(
        self,
        model_list,
        **kwargs
    ):
        self.log_print([
            "Generating models in tiered greedy search at spawn
                step {}.".format(
                self.spawn_step,
            )
        ])

        if self.spawn_stage[-1] is None:
            try:
                previous_branch_champ = model_list[0]
                self.tier_branch_champs[self.tier].append(
                    previous_branch_champ)
            except:
                previous_branch_champ = None

        elif "getting_tier_champ" in self.spawn_stage[-1]:
            previous_branch_champ = model_list[0]
            self.log_print([
                "Tier champ for {} is {}".format(self.tier,
                    model_list[0])
            ])
            self.tier_champs[self.tier] = model_list[0]
            self.tier += 1
```

```
            self.log_print(["Tier now = ", self.tier])
            self.spawn_stage.append(None) # normal processing

            if self.tier > self.max_tier:
                self.log_print(["Completed tree for ES"])
                self.spawn_stage.append('Complete')
                return list(self.tier_champs.values())
        else:
            self.log_print([
                "Spawn stage:", self.spawn_stage
            ])

        new_models = greedy_add(
            current_model = previous_branch_champ,
            terms = self.term_tiers[self.tier]
        )
        self.log_print([
            "tiered search new_models=", new_models
        ])

        if len(new_models) == 0:
            # no models left to find - get champions of branches
               from this tier
            new_models = self.tier_branch_champs[self.tier]
            self.log_print([
                "tier champions: {}".format(new_models)
            ])
            self.spawn_stage.append("getting_tier_champ_{}".
               format(self.tier))
        return new_models

    def check_tree_completed(
        self,
        spawn_step,
        **kwargs
    ):
        r"""
        QMLA asks the exploration tree whether it has finished
           growing;
```

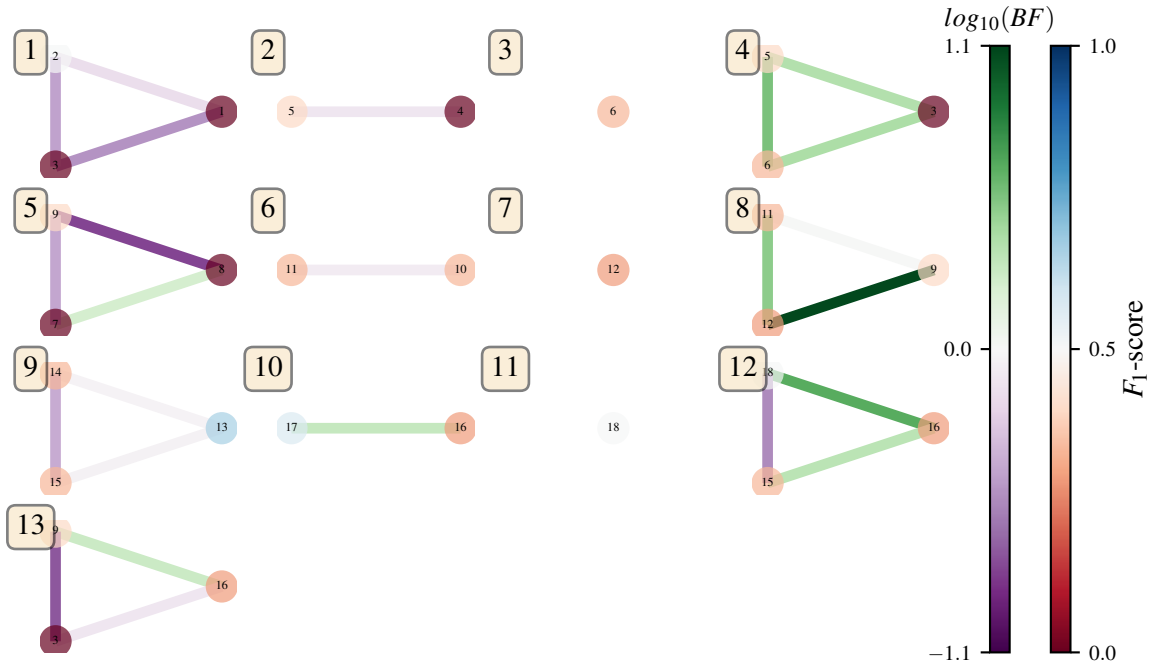
```
        the exploration tree queries the exploration strategy
           through this method
        """
        if self.tree_completed_initially:
            return True
        elif self.spawn_stage[-1] == "Complete":
            return True
        else:
            return False


def greedy_add(
    current_model,
    terms,
):
    r"""
    Combines given model with all terms from a set.

    Determines which terms are not yet present in the model,
    and adds them each separately to the current model.

    :param str current_model: base model
    :param list terms: list of strings of terms which are to be
       added greedily.
    """

    try:
        present_terms = current_model.split('+')
    except:
        present_terms = []
    nonpresent_terms = list(set(terms) - set(present_terms))

    term_sets = [
        present_terms+[t] for t in nonpresent_terms
    ]

    new_models = ["+".join(term_set) for term_set in term_sets]

    return new_models
```

Listing C.18: ExampleGreedySearchTiered exploration stategy

with corresponding results in Fig. C.8.



(a)



(b)

Figure C.8: Tiered greedy exploration strategy. **(a)**, composition_of_models. **b**, graphs_of_branches_ExampleGreedySearchTiered: shows which models reside on each branches of the exploration tree. Models are coloured by their F_1-score, and edges represent the BF between models. In each tier, three branches greedily add terms, and a fourth branch considers the champions of the first three branches in order to nominate a tier champion. The final branch consists only of the tier champions, to nominate the global champion, $\hat{H}'$.

BIBLIOGRAPHY

[1] Marcus W Doherty, Neil B Manson, Paul Delaney, Fedor Jelezko, Jörg Wrachtrup, and Lloyd CL Hollenberg. The nitrogen-vacancy colour centre in diamond. *Physics Reports*, 528(1):1–45, 2013.

[2] Gordon Davies and MF Hamer. Optical studies of the 1.945 ev vibronic band in diamond. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 348(1653):285–298, 1976.

[3] J Meijer, B Burchard, M Domhan, C Wittmann, Torsten Gaebel, I Popa, F Jelezko, and J Wrachtrup. Generation of single color centers by focused nitrogen implantation. *Applied Physics Letters*, 87(26):261909, 2005.

[4] AM Edmonds, UFS D'Haenens-Johansson, RJ Cruddace, ME Newton, K-MC Fu, C Santori, RG Beausoleil, DJ Twitchen, and ML Markham. Production of oriented nitrogen-vacancy color centers in synthetic diamond. *Physical Review B*, 86(3):035201, 2012.

[5] A Lenef and SC Rand. Electronic structure of the n-v center in diamond: Theory. *Physical Review B*, 53(20):13441, 1996.

[6] Sebastian Knauer. *Photonic Structure Coupling and Strain Sensing with Single Photon Emitters*. PhD thesis, University of Bristol, 2016.

[7] Benjamin Smeltzer, Lilian Childress, and Adam Gali. 13c hyperfine interactions in the nitrogen-vacancy centre in diamond. *New Journal of Physics*, 13(2):025021, 2011.

[8] Heinz-Peter Breuer, Francesco Petruccione, et al. *The theory of open quantum systems*. Oxford University Press on Demand, 2002.

[9] Antonio A. Gentile, Brian Flynn, Sebastian Knauer, Nathan Wiebe, Stefano Paesani, Christopher E. Granade, John G. Rarity, Raffaele Santagati, and Anthony Laing. Learning models of quantum systems from experiments, 2020.

[10] MS Blok, Cristian Bonato, ML Markham, DJ Twitchen, VV Dobrovitski, and R Hanson. Manipulating a qubit through the backaction of sequential partial measurements and real-time feedback. *Nature Physics*, 10(3):189–193, 2014.

[11] Adam Gali, Maria Fyta, and Efthimios Kaxiras. Ab initio supercell calculations on nitrogen-vacancy center in diamond: Electronic structure and hyperfine tensors. *Physical Review B*, 77(15):155206, 2008.

[12] MV Gurudev Dutt, L Childress, L Jiang, E Togan, J Maze, F Jelezko, AS Zibrov, PR Hemmer, and MD Lukin. Quantum register based on individual electronic and nuclear spin qubits in diamond. *Science*, 316(5829):1312–1316, 2007.

[13] P-Y Hou, L He, F Wang, X-Z Huang, W-G Zhang, X-L Ouyang, X Wang, W-Q Lian, X-Y Chang, and L-M Duan. Experimental hamiltonian learning of an 11-qubit solid-state quantum spin register. *Chinese Physics Letters*, 36(10):100303, 2019.

[14] L Childress, MV Gurudev Dutt, JM Taylor, AS Zibrov, F Jelezko, J Wrachtrup, PR Hemmer, and MD Lukin. Coherent dynamics of coupled electron and nuclear spin qubits in diamond. *Science*, 314(5797):281–285, 2006.

[15] L_G Rowan, EL Hahn, and WB Mims. Electron-spin-echo envelope modulation. *Physical Review*, 137(1A):A61, 1965.

[16] Forrest T Charnock and TA Kennedy. Combined optical and microwave approach for performing quantum spin operations on the nitrogen-vacancy center in diamond. *Physical Review B*, 64(4):041201, 2001.

[17] Antonio Andreas Gentile. *Operating practical quantum devices in the pre-threshold regime*. PhD thesis, University of Bristol, 2020.

[18] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[19] David A Broadway, Jean-Philippe Tetienne, Alastair Stacey, James DA Wood, David A Simpson, Liam T Hall, and Lloyd CL Hollenberg. Quantum probe hyperpolarisation of molecular nuclear spins. *Nature communications*, 9(1):1–8, 2018.

[20] Brian Flynn. Mathematical introduction to quantum computation, 2015. Undergraduate thesis.

[21] Quantum model learning agent documentation. https://quantum-model-learning-agent.readthedocs.io/en/latest/, Jan 2021. [Online; accessed 12. Jan. 2021].

[22] Brian Flynn. Quantum model learning agent. https://github.com/flynnbr11/QMLA, 2021.