Doctorate of Philosophy

Schrödinger's Catwalk

Machine learning methods to distill models of quantum systems

Brian Flynn

University of Bristol

March, 2021

ABSTRACT

Quantum technologies exploit quantum mechanical processes to achieve outcomes beyond the reach of classical machinery. One of their most promising applications is quantum simulation, whereby particles, atoms and molecules can be examined thoroughly for the first time, having been beyond the scope of even the most powerful supercomputers.

Models have been useful tools in understanding physical systems: these are mathematical structures encoding physical interactions, which allow us to predict how the system will behave under various conditions. Models of quantum systems are particularly difficult to design and test, owing to the huge computational resources required to represent them accurately. In this thesis, we introduce and develop an algorithm to characterise quantum systems efficiently, by inferring a model consistent with their observed dynamics. The *Quantum Model Learning Agent* (QMLA) is an extensible framework which permits the study of any quantum system of interest, by combining quantum simulation with state of the art machine learning. QMLA iteratively proposes candidate models and trains them against the target system, finally declaring a single model as the best representation for the system of interest.

We describe QMLA and its implementation through open source software, before testing it under a series of physical scenarios. First, we consider idealised theoretical systems in simulation, verifying the core principles of QMLA. Next, we incorporate strategies for generating candidate models by exploiting the information QMLA has gathered to date; by incorporating a genetic algorithm within QMLA, we explore vast spaces of valid candidate models, with QMLA reliably identifying the precise target model. Finally, we apply QMLA to *realistic* quantum systems, including operating on experimental data measured from an electron spin in a nitrogen vacancy centre.

QMLA is shown to be effective in all cases studied in this thesis; however, of greater interest is the platform it provides for examining quantum systems. QMLA can aid engineers in configuring experimental setups, facilitate calibration of near term quantum devices, and ultimately enable complete characterisation of natural quantum structures. This thesis marks the beginning of a new line of research, into automating the understanding of quantum mechanical systems.

DECLARATION OF AUTHORSHIP

This dissertation is submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctorate of Philosophy in the Faculty of Science.

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's *Regulations and Code of Practice for Research Degree Programmes* and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Signed:

_____

Date:

_____

Word count: $\sim 50,000$

ACKNOWLEDGEMENTS

For

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LISTINGS

ACRONYMS

CLE	classical likelihood estimation, Section 1.3
CPU	central processing unit
EDH	experiment design heuristic, Section 1.6
ES	exploration strategy, Section 2.4
GA	genetic algorithm, **??**
IQLE	interactive quantum likelihood estimation, Section 1.3.1
LE	Loschmidt echo, Section 1.3.1
LTL	log total likelihood, Section 1.4
NVC	nitrogen-vacancy centre, **??**
PGH	particle guess heuristic, Section 1.6.1
QHL	quantum Hamiltonian learning, Chapter 1.
QL	quadratic loss, Eq. (1.19)
QLE	quantum likelihood estimation, Section 1.3
QM	quantum mechanics, **??**
QMLA	Quantum Model Learning Agent, Chapter 2.
SMC	sequential Monte Carlo, Section 1.2
TL	total likelihood, Eq. (1.15)
TLTL	total log total likelihood, Eq. (1.18)

GLOSSARY

N_E	Number of experiments to perform during model training through QHL.
N_P	Number of particles used when training a model through QHL.
Q	Quantum system which is the target of QMLA, i.e. the system to be characterised.
$\hat{H}_0$	True model for the target system, Q; i.e. the Hamiltonian model form which QMLA is attempting to retrieve in a given instance.
champion	See champion model.
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, **??**.
experiment	Experiment performed upon Q when training a model through QHL, Section 1.2.
Hamiltonian	Mathematical structure which captures all interactions to which a quantum system is subject, **??**.
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, Section 1.3.
model	The mathematical description of some quantum system, Section 2.1.
model space	Abstract space containing all descriptions (within defined constraints such as dimension) of the system as models, Section 2.4.1.

particle	Sampled from prior distribution during model training through QHL, each particle gives a parameterisation corresponding to a unique hypothesis about Q, Section 1.2.
probe	Input state, $\|\psi\rangle$, into which the target system is initialised, before unitary evolution, Section 1.7.
run	Collection of QMLA instances, usually targeting the same system with the same initial conditions.
true model	The model which correctly describes the target system. $\hat{H}_0$ is known for simulated Q but not known precisely for experimental systems.
volume	Volume of a parameter distribution's credible region, Section 1.5.1.

LIST OF PUBLICATIONS

The work presented in this thesis has appeared[1] publicly in several formats.

Papers

1. *Learning models of quantum systems from experiments.* A.A. Gentile, Brian Flynn, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, R. Santagati and A. Laing. arXiv preprint arXiv:2002.06169 (2020); accepted *Nature Physics* (2021); referred to throughout as [1].
2. *Quantum Model Learning Agent: quantum systems' characterisation through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe and A. Laing. Reporting outcomes of studies on theoretical systems as described in this thesis. In Preparation (2021); referred to throughout as [2].
3. *Quantum Model Learning Agent: Python framework for characterising quantum systems.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe and A. Laing. Technical manuscript detailing software implementation. In Preparation (2021).

Software

4. *QMLA: Python framework for the reverse engineering of Hamiltonian models of quantum systems through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe, S. Paesani, C. E. Granade, and A. Laing. Codebase; open sourced via Github repository; referrred to throughout as [3].
5. *Quantum Model Learning Agent*. Brian Flynn, A.A. Gentile, R. Santagati, N. Wiebe, S. Paesani, C. E. Granade, and A. Laing. Documentation for codebase; referrred to throughout as [4].

Conference Proceedings - Talks [2]

6. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Techniques in Machine Learning, Online, 2020.

1 Or will appear in the near future.
2 Note: only conferences proceedings presented by the author are included.

7. *Learning models of quantum systems from experiments.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Bristol Quantum Information Technologies, Online, 2020.
8. *Quantum Model Learning: characterizing quantum systems through machine learning*. Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Techniques in Machine Learning, Daejeon, South Korea, 2019.
9. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Engineering Centre for Doctoral Training Conference, Bristol, UK, 2019. Awarded *Talk prize*.

Conference Proceedings - Posters

10. *Quantum Model Learning Agent.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Machine Learning for Quantum, IOP Conference, Online, 2021. Awarded *Poster prize*.
11. *Quantum Model Learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Machine Learning for Quantum Technologies, Erlangen, Germany, 2019.
12. *Quantum Model Learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Bristol Quantum Information Technologies, Bristol, UK, 2019.
13. *Quantum Model Learning: characterizing quantum systems through machine learning.* Brian Flynn, A.A. Gentile, R. Santagati, S, Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, and A. Laing. Quantum Engineering Centre for Doctoral Training Conference, Bristol, UK, 2018. Awarded *Poster prize*.

INTRODUCTION

Part I

CONTEXTUAL REVIEW

Part II

ALGORITHMS

1

QUANTUM HAMILTONIAN LEARNING

First suggested in [5] and since developed [6,7] and implemented [8], quantum Hamiltonian learning (QHL) is a machine learning algorithm for the optimisation of a given Hamiltonian parameterisation against a quantum system whose model is known a priori. Given a target quantum system, Q, known to be described by some Hamiltonian $\hat{H}(\vec{\alpha})$, QHL optimises $\vec{\alpha}$. This is achieved by interrogating Q and comparing its outputs against proposals $\vec{\alpha}_p$. In particular, an experiment is designed, consisting of an input state, $|\psi\rangle$, and an evolution time, t. This experiment is performed on Q, whereupon its measurement yields the datum $d \in \{0,1\}$ – i.e. the eigenstate $|d\rangle \in \{|0\rangle, |1\rangle\}$ is observed – according to the expectation value $|\langle\psi| e^{-i\hat{H}_0 t} |\psi\rangle|^2$. Then, on a trusted (quantum) simulator, proposed parameters $\vec{\alpha}_p$ are encoded to the known Hamiltonian, and the same probe state is evolved for the chosen t and projected on to $|d\rangle$, i.e. $|\langle d| e^{-i\hat{H}(\vec{\alpha}_p)t} |\psi\rangle|^2$ is computed. The task for QHL is then to find $\vec{\alpha}'$ for which this quantity is close to 1 for all values of $\{|\psi\rangle, t\}$, i.e. the parameters input to the simulation produce dynamics consistent with those measured from Q.

The procedure is as follows. A *prior* probability distribution $\Pr(\vec{\alpha})$ in a parameter space of dimension $|\vec{\alpha}|$ is initialised to represent the constituent parameters of $\vec{\alpha}$. $\Pr(\vec{\alpha})$ is typically a multivariate normal (Gaussian) distribution; it is therefore necessary to pre-suppose some mean and standard deviation for each parameter in $\vec{\alpha}$. This imposes prior knowledge on the algorithm whereby the programmer must decide the range in which parameters are *likely* to fit: although QHL is generally robust and capable of finding parameters outside of this prior, the prior must at least capture the order of magnitude of the target parameters. It is important to understand, then, that QHL removes the prior knowledge of the precise parameter representing an interaction in Q, but does rely on a *ball-park* estimate thereof from which to start.

In short, QHL samples parameter vectors $\vec{\alpha}_p$ from $\Pr(\vec{\alpha})$, simulates experiments by computing the *likelihood* $|\langle d| e^{-i\hat{H}(\vec{\alpha}_p)t} |\psi\rangle|^2$ for experiments $\{|\psi\rangle, t\}$ designed by a QHL heuristic subroutine, and iteratively improves the probability distribution of the parameterisation $\Pr(\vec{\alpha})$ through standard *Bayesian inference*. A given set of $\{|\psi\rangle, t\}$ is called an *experiment*, since it corresponds to preparing, evolving and measuring Q once[1]. QHL iterates for N_E experiments. The parameter vectors sampled are called *particles*: there are N_P particles used per experiment. Each particle used incurs one further calculation of the likelihood function – this calculation, on a classical computer, is exponential in the number of qubits of the model under consideration (because each unitary evolution relies on the exponential of the $2^n \times 2^n$ Hamiltonian matrix of n qubits), **??**. Likewise, each additional experiment incurs the cost of calculation of N_P particles, so the total cost of running QHL to train a model is $\propto N_e N_p$. It is therefore preferable to use as few

1 Experimentally, this may involve repeating a measurement many times to determine a majority result and to mitigate noise.

particles and experiments as possible, though it is important to include sufficient resources that the parameter estimates have the opportunity to converge. Access to a fully operational, trusted quantum simulator admits an exponential speedup by simulating the unitary evolution instead of computing the matrix exponential classically.

1.1 BAYES' RULE

Bayes' rule is used to update a probability distribution describing hypotheses, $\Pr(\text{hypothesis})$, when presented with new information (data). That is, the probabilty that a hypothesis is true is replaced by the initial probability that it *was* true, $\Pr(\text{hypothesis})$, multiplied by the likelihood that the new data would be observed were that hypothesis true, $\Pr(\text{data}|\text{hypothesis})$, normalsied by the probability of observing that data in the first place, $\Pr(\text{data})$. It is stated as

$$\Pr(\text{hypothesis}|\text{data}) = \frac{\Pr(\text{data}|\text{hypothesis}) \times \Pr(\text{hypothesis})}{\Pr(\text{data})}. \tag{1.1}$$

We wish to represent our knowledge of Hamiltonian parameters with a distribution, $\Pr(\vec{\alpha})$: in this case hypotheses $\vec{\alpha}$ attempt to describe data, $\mathcal{D}$, measured from the target quantum system, from a set of experiments $\mathcal{E}$, so we can rewrite Bayes' rule as

$$\Pr(\vec{\alpha}|\mathcal{D};\mathcal{E}) = \frac{\Pr(\mathcal{D}|\vec{\alpha};\mathcal{E})\ \Pr(\vec{\alpha})}{\Pr(\mathcal{D}|\mathcal{E})}. \tag{1.2}$$

We can consider Eq. (1.2) at the level of single *particles*, i.e. individual vectors $\vec{\alpha}$ in the parameter space, sampled from $\Pr(\vec{\alpha})$:

$$\Pr(\vec{\alpha}_p|d;e) = \frac{\Pr(d|\ \vec{\alpha}_p;\ e)\ \Pr(\vec{\alpha}_p)}{\Pr(d|e)} \tag{1.3}$$

where

- e are the experimental controls of a single experiment, e.g. evolution time and input probe state;
- d is the datum, i.e. the (usually) binary outcome of measuring Q under conditions e;
- $\vec{\alpha}_p$ is the *hypothesis*, i.e. a single parameter vector, called a particle, sampled from $\Pr(\vec{\alpha})$;
- $\Pr(\vec{\alpha}_p|d;e)$ is the *updated* probability of this particle following the experiment e, i.e. accounting for new datum d, the probability that $\vec{\alpha} = \vec{\alpha}_0$;
- $\Pr(d|\vec{\alpha}_p;e)$ is the likelihood function, i.e how likely it is to have measured the datum d from the system assuming $\vec{\alpha}_p$ are the true parameters and the experiment e was performed;
- $\Pr(\vec{\alpha}_p)$ is the probability that $\vec{\alpha}_p = \vec{\alpha}_0$ according to the prior distribution $\Pr(\vec{\alpha})$, which we can immediately access;

- $\Pr(d|e)$ is a normalisation factor, the chance of observing d from experiment e irrespective of the underlying hypothesis such that $\sum_{\{d\}} \Pr(d|e) = 1$.

In order to compute the updated probability for a given particle, then, all that is required is a value for the likelihood function. This is equivalent to the expectation value of projecting $|\psi\rangle$ onto $|d\rangle$, after evolving $\hat{H}(\vec{\alpha}_p)$ for t, i.e.

$$\Pr(d|\vec{\alpha}; e) = |\langle d| e^{-i\hat{H}(\vec{\alpha}_p)t} |\psi\rangle|^2, \tag{1.4}$$

which can be simulated clasically or using a quantum simulator (see Section 1.3). It is necessary first to know the datum d (either 0 or 1) which was projected by Q under experimental conditions. Therefore we first perform the experiment e on Q (preparing the state $|\psi\rangle$ evolving for t and projecting again onto $\langle\psi|$) to retrieve the datum d. d is then used for the calculation of the likelihood for each particle sampled from $\Pr(\vec{\alpha})$. Each particle's probability can be updated by Eq. (1.3), allowing us to redraw the entire probability distribution. We can hence compute a *posterior* probability distribution by performing this routine on a set of N_P particles: we hypothesise N_P parameterisations $\vec{\alpha}_i$ sampled from $\Pr(\vec{\alpha})$, and update their $\Pr(\vec{\alpha}_i)$ in proportion to their likelihood. In effect, hypotheses (particles) which are found to be highly likely are given increased credence, while those with low likelihood have their credence decreased.

1.2 SEQUENTIAL MONTE CARLO

In practice, QHL samples from and updates $\Pr(\vec{\alpha})$ via sequential Monte Carlo (SMC). SMC samples the N_P particles from $\Pr(\vec{\alpha})$, and assigns each particle a weight, $w_0 = 1/N_p$. Each particle corresponds to a unique position in the parameters' space, i.e. $\vec{\alpha}_p$. Following the calculation of the likelihood, $\Pr(d|\vec{\alpha}_p; e)$, the weight of particle p is updated from its initial value of w_p^{old} by Eq. (1.5).

$$w_p^{\text{new}} = \frac{\Pr(d|\vec{\alpha}_p; e) \times w_p^{\text{old}}}{\sum_p w_p \Pr(\vec{\alpha}_p|d; e)}. \tag{1.5}$$

In this way, strong particles – with high $\Pr(d|\vec{\alpha}_p; e)$ – have their weight increased, while weak particles (low $\Pr(d|\vec{\alpha}_p; e)$) have their weights decreased, and the sum of weights remains normalised. Within a single experiment, the weights of all N_P particles are updated: we *simultaneously* update sampled particles' weights as well as $\Pr(\vec{\alpha})$. The procedure of updating particles' weights iterates for the subsequent experiment, using the *same* particles: we do *not* redraw N_P particles for every experiment. Eventually, the weights of most particles fall below a threshold, r_t, meaning that only that fraction of particles have reasonable likelihood of being $\vec{\alpha}_0$. At this stage, SMC *resamples*[2], i.e. selects new particles, according to the updated $\Pr(\vec{\alpha})$. Then, the new particles are in the range of parameters which is known to be more likely, while particles in the region of low-weight are effectively discarded. Usually, we set $r_t = 0.5$, although this hyperparameter can have a large impact on the rate of learning, so can be optimised in particular circumstances, see Fig. 1.2.

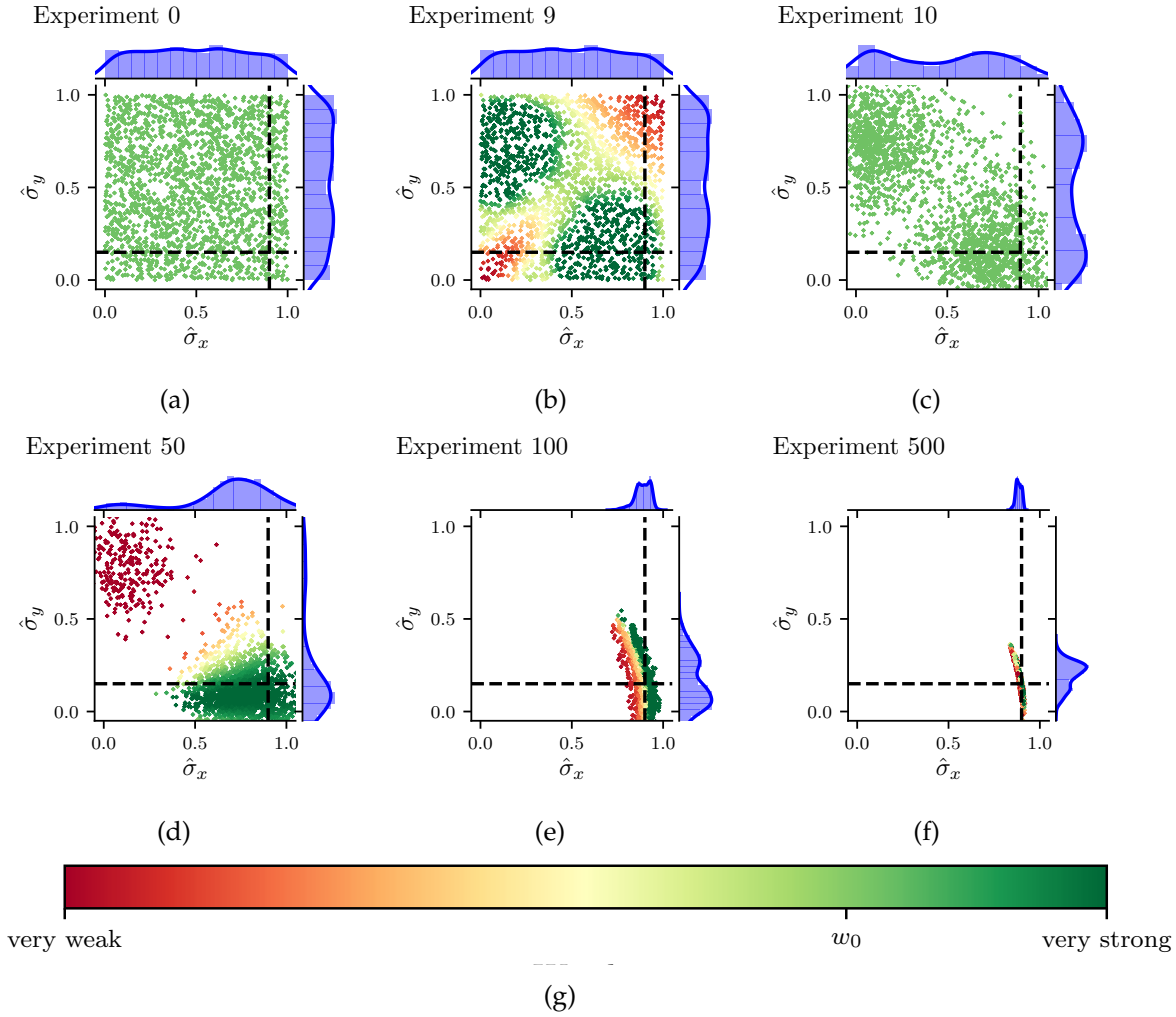

Figure 1.1: Quantum Hamiltonian learning (QHL) via sequential Monte Carlo (SMC). The studied model has two terms, $\{\hat{\sigma}_x, \hat{\sigma}_y\}$ with true parameters $\alpha_x = 0.9, \alpha_y = 0.15$ (dahsed lines), with resources $N_e = 500, N_p = 2000$ for training the model. Crosses represent particles, while the distribution $\Pr(\alpha_p)$ for each parameter can be seen along the top and right-hand-sides of each subplot. Both parameters are assigned a uniform probability distribution $\mathcal{U}(0,1)$, representing our prior knowledge of the system. **a,** SMC samples N_P particles from the initial joint probability distribution, with particles uniformly spread across the unit square, each assigned the starting *weight* w_0. At each experiment e, each of these particles' likelihood is computed according to Eq. (1.3) and its weight is updated by Eq. (1.5). **b,** after 9 experiments, the weights of the sampled particles are sufficiently informative that we know we can discard some particles while most likely retaining the true parameters. **c,** SMC resamples according the current $\Pr(\vec{\alpha})$, i.e. having accounted for the experiments and likelihoods observed to date, a new batch of N_P particles are drawn, and each reassigned weight w_0, irrespective of their weight prior to resampling. **d-e,** After further experiments and resamplings, SMC narrows $\Pr(\vec{\alpha})$ to a region around the true parameters. **f,** The final *posterior* distribution consists of two narrow distributions centred on α_x and α_y. By taking the mean of the posterior distribution, we approximate the parameters of interest as $\vec{\alpha}'$.

This procedure is easiest understood through the example presented in Fig. 1.1, where a two-parameter Hamiltonian is learned starting from a uniform distribution. $N_p = 2000$ particles are used to propose hypotheses distributed evenly throughout the parameter space, each of which are subject to weight updates as outlined above. In this example, after 9 experiments the particles around the diagonal ($x = y$) are deemed unlikely, while clusters form in the opposite corners where the algorithm finds the hypotheses credible. Before the tenth experiment, the algorithm resamples, i.e. reassigns weights based on the present $\Pr(\vec{\alpha})$. The algorithm iteratively reassigns weight to particles based on their likelihoods, redraws $\Pr(\vec{\alpha})$ and resamples. We show the state of the particles after 50, 100 and 500 experiments, with the overall result of a highly peaked parameter distribution, whose centre is near the target parameters.

1.3 LIKELIHOOD

The fundamental step within QHL is the calculation of likelihood, which enables updates of the probability distribution in Eq. (1.3). The key to the learning algorithm is that likelihood can be retrieved from the Born rule, which captures how likely a given a quantum system is to be measured in an eigenstate. When we have retrieved a datum, d, from Q, we can compute the probability that Q would be measured in the corresponding eigenstate $|d\rangle$ – this probability serves as the likelihood, and is given by Eq. (1.4).

In some cases, it is feasible to derive the closed form of the likelihood, for example as a simple expression in terms of the Hamiltonian parameters, which we will exemplify in Section 1.3.2. Closed form likelihoods allow for rapidly testing hypothetical parameters for comparison against the observed data, so QHL can feasibly be run with high N_E, N_P. In general, however, it is not possible to derive the closed form of the likelihood, and instead the likelihood must be computed through Eq. (1.4), which can be done either on a classical or quantum simulator. The case where the likelihood is computed on a quantum simulator is referred to as quantum likelihood estimation (QLE) [7, 8], and can leverage any algorithm for the calculation of Hamiltonian dynamics to achieve *quantum speedup* [10–12].

In this thesis, we do not implement the presented algorithms on quantum hardware, instead investigating their performance using idealised classical simulations, i.e. classical likelihood estimation (CLE). The reliance on classical resources demands that Eq. (1.4) be computed explicitly, notably involving the matrix exponential $e^{-i\hat{H}(\vec{\alpha}_p)t}$. Since the Hamiltonian matrix scales with the size of the simulated system, running QHL for an n-qubit systems requires exponentiation of its $2^n \times 2^n$ Hamiltonian matrix, in order to compute the exact likelihoods required for learning. This overhead restricts the applicability of CLE: $n = 11$-qubit systems' Hamiltonians exhaust the memory capacity of most conventional classical computers. In practice, QHL is limited by the computation of the total $N_e N_p$ matrix exponentials required for training:

2 Particles are *resampled* according to a resampling algorithm. Throughout this thesis, we always use the Liu-West resampling algorithm [9].

we will only entertain systems which can be represented by Hamiltonians of up to $n = 8$ qubits. In principle, larger systems could be condensed for simulation on available classical resources, or those resources used more efficiently [13], but the remit of this thesis can be fulfilled with demonstrations in the domain $n \leq 8$ qubits, so we do not endeavour to find the most effective classical strategies.

Adopting the notation used by QInfer [14], upon which our software builds, the expectation value for a the unitary operator is given by

$$\Pr(0) = |\langle\psi| e^{-i\hat{H}_p t} |\psi\rangle|^2 = l(d = 0|\hat{H}_p; e). \tag{1.6}$$

In Eq. (1.6), the input basis is assigned the measurement label $d = 0$, and this $\Pr(0)$ is the probability of measuring $d = 0$, i.e. measuring the same state as was prepared as input. We assume a binary outcome model[3], i.e. that the system is measured either in $|\psi\rangle$ (labelled $d = 0$), or it is not ($|\psi_\perp\rangle, d = 1$); the likelihood for the latter case is

$$\Pr(1) = l(d = 1|\hat{H}_p; e) = \sum_{\{|\psi_\perp\rangle\}} |\langle\psi_\perp| e^{-i\hat{H}_p t} |\psi\rangle|^2 = 1 - \Pr(0). \tag{1.7}$$

Usually we will refer to the case where Q is projeced onto the input state $|\psi\rangle$, so the terms *likelihood*, *expectation value* and $\Pr(0)$ are synonymous, unless otherwise stated.

1.3.1 *Interactive quantum likelihood estimation*

A fundamental result in quantum mechanics (QM) – the Loschmidt echo (LE) – shows that marginally differing Hamiltonians produce exponentially diverging evolutions, undermining the basis of QLE, i.e. that the likelihood function can inform Bayesian updates to a paramater distribution. The LE concerns the result when Q is prepared in some initial state $|\psi\rangle$, evolved forward in time by some $\hat{H}_+$, then evolved *backwards*[4] in time by $\hat{H}_-$, and projected back onto $|\psi\rangle$. The LE – or the *fidelity* – is given by

$$M(t) = \left|\langle\psi|e^{+i\hat{H}_- t} e^{-i\hat{H}_+ t}|\psi\rangle\right|^2. \tag{1.8}$$

$M(t)$ is dictated by the *similarity* between the two Hamiltonians. If $\hat{H}_+ = \hat{H}_-$, then $M(t) = 1$, while $\|\hat{H}_+ - \hat{H}_-\|_2 > 0$ yields $M(t) < 1$, indicating disagreement between the two Hamiltonians. The fidelity is characterised by a number of distinct regions, depending on the evolution time, t:

$$M(t) \sim \begin{cases} 1 - \mathcal{O}(t^2), & t \leq t_c \\ e^{-\mathcal{O}(t)}, & t_c \leq t \leq t_s \\ 1/\|\hat{H}\|, & t \geq t_s \end{cases} \tag{1.9}$$

3 In principle the output does not have to be binary, so we sum over the general set $\{|\psi_\perp\rangle\}$ of eigenstates orthogonal to $|\psi\rangle$ in Eq. (1.7).

4 Equivalently and in practice, evolved forward in time for $(-\hat{H}_-)$.

where $\|\hat{H}\|$ is the dimension of the Hamiltonians, and t_c, t_s are bounds on the evolution time marking the transition between the *parabolic decay*, *asymptotic decay* and *saturation* of the echo [15]. t_c and t_s generally depend on the similarity between $\hat{H}_+$ and $\hat{H}_-$: intuitively, as $\|\hat{H}_+ - \hat{H}_-\|_2$ decreases, the echo does not saturate until higher evolution times.

Recall that the Bayesian updates to the parameter distribution relies on good hypotheses receiving likelihood $l_e \sim 1$, and weak hypotheses receiving $l_e \sim 0$. The LE tells us that there is a small range of evolution times ($t \lesssim t_c$) for which even good particles may expect $l_e \sim 1$. We can exploit this effect, however: by designing experiments with $t \sim t_c$, the likelihood is extremely sensitive to the parameterisation, in that only particles close to the precise parameters will give a high likelihood in this regime. This is the basis of the particle guess heuristic, described in Section 1.6.1.

We can relate the LE to the likelihood, Eq. (1.4), by supposing $\hat{H}_- = \hat{\mathbb{1}}$. It is inescapable that the likelihoods are exponentially small if the evolution times are not short; experimentally, exponentially small expectation values demand an exponential number of measurements to approximate accurately. Furthermore, short-time experiments are known to be uninformative [6, 16]. Together, these problems render QLE unscalable. We overcome these inherent problems by using a modification of QLE: interactive quantum likelihood estimation (IQLE), the key to which is invoking a likelihood function other than Eq. (1.4).

In effect, the LE guarantees that, for most t, if $\hat{H}_- \not\approx \hat{H}_+$, then $M(t) \ll 1$, while $\hat{H}_- \approx \hat{H}_+$ gives $M(t) \approx 1$. This can be exploited for learning: by taking $\hat{H}_+$ as either $\hat{H}_0$ (the true system) or $\hat{H}(\vec{\alpha})$ (particle/hypothesis), and sampling $\hat{H}_-$ from $\Pr(\vec{\alpha})$, we can adopt Eq. (1.8) as the likelihood function. Thus, both $\hat{H}_0$ and $\hat{H}(\vec{\alpha})$ have been evolved for arbitrary t, and unevolved by a common unitary, $e^{i\hat{H}_+ t}$. The likelihood that they are both measured in the same eigenstate is still a function of the overlap between the hypothesis and the true parameters, but here the informative difference between them is not drowned out by the chaotic effects captured by the LE, as it had been in QLE.

Importantly, IQLE can only be used where we can *reliably* reverse the evolution for the system under study. In order that the reverse evolution is relibable, it must be performed on a trusted simulator, restricting IQLE to cases where a coherent quantum channel exists between the target system and a trusted simulator. This automatically excludes any open quantum systems, as well as most realistic experimental setups, although such channels can be achieved [17]. The remaining application for IQLE, and correspondingly QHL, is in the characterisation of untrusted quantum simulators, which can realise such coherent channels [8].

1.3.2 *Analytical likelihood*

For some Hamiltonians, we can derive an analytical likelihood function to describe their dynamics [18, 19]. For instance, the Hamiltonian for an oscillating electron spin in a nitrogen-vacancy centre is given by

$$\hat{H}(\omega) = \frac{\omega}{2}\hat{\sigma}_z, \tag{1.10}$$

where ω is the Rabi frequency of the spin. Then, recalling that $\hat{\sigma}_z\hat{\sigma}_z = \hat{\mathbb{1}}$, so $\hat{\sigma}_z^{2k} = \hat{\mathbb{1}}$ and $\hat{\sigma}_z^{2k+1} = \hat{\sigma}_z$, using MacLaurin expansion, the unitary evolution of Eq. (1.10) is given by

$$\begin{aligned} U = e^{-i\hat{H}(\omega)t} = e^{-i\frac{\omega t}{2}\hat{\sigma}_z} &= \cos\left(\frac{\omega t\hat{\sigma}_z}{2}\right) - i\sin\left(\frac{\omega t\hat{\sigma}_z}{2}\right) \\ &= \left(\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k)!}\left(\frac{\omega t}{2}\right)^{2k}\hat{\sigma}_z^{2k}\right) - i\left(\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)!}\left(\frac{\omega t}{2}\right)^{2k+1}\hat{\sigma}_z^{2k+1}\right) \\ &= \left(\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k)!}\left(\frac{\omega t}{2}\right)^{2k}\right)\hat{\mathbb{1}} - i\left(\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)!}\left(\frac{\omega t}{2}\right)^{2k+1}\right)\hat{\sigma}_z \\ &= \cos\left(\frac{\omega t}{2}\right)\hat{\mathbb{1}} - i\sin\left(\frac{\omega t}{2}\right)\hat{\sigma}_z \end{aligned} \tag{1.11}$$

Then, evolving a probe $|\psi_0\rangle$ and projecting onto a state $|\psi_1\rangle$ gives

$$\langle\psi_1|\,U\,|\psi_0\rangle = \cos\left(\frac{\omega t}{2}\right)\langle\psi_1|\psi_0\rangle - i\sin\left(\frac{\omega t}{2}\right)\langle\psi_1|\hat{\sigma}_z|\psi_0\rangle\,. \tag{1.12}$$

By initialising and projecting into the same state, say $|\psi_0\rangle = |\psi_1\rangle = |+\rangle$, and reaclling $\hat{\sigma}_z\,|+\rangle = |-\rangle$, we have

$$\begin{aligned} \langle\psi_1|\psi_0\rangle &= \langle+|+\rangle = 1 \\ \langle\psi_1|\hat{\sigma}_z|\psi_0\rangle &= \langle+|-\rangle = 0 \\ \implies \langle\psi_1|U|\psi_0\rangle &= \cos\left(\frac{\omega t}{2}\right). \end{aligned} \tag{1.13}$$

If the system measures in $|+\rangle$, we set the datum $d = 0$, otherwise $d = 1$. From Born's rule, and in analogy with Eq. (1.4), we can formulate the likelihood function, where the hypothesis is the single parameter ω, and the sole experimental control is t,

$$\Pr(d = 0|\omega;t) = |\langle\psi_1|U|\psi_0\rangle|^2 = \cos^2\left(\frac{\omega t}{2}\right); \tag{1.14a}$$

$$\Pr(d = 1|\omega;t) = 1 - \cos^2\left(\frac{\omega t}{2}\right) = \sin^2\left(\frac{\omega t}{2}\right). \tag{1.14b}$$

This analytical likelihood will underly the simulations used in the following introductions, except where explicitly mentioned.

1.4 TOTAL LOG TOTAL LIKELIHOOD

We have already used the concept of likelihood to update our parameter distribution during SMC; we can consolidate the likelihoods of all particles with respect to a single datum, d, from a single experiment e, in the *total likelihood (TL)*,

$$l_e = \sum_{p \in \{p\}} \Pr(d|\vec{\alpha}_p; e) \times w_p^{\text{old}}, \tag{1.15}$$

where w_p^{old} are the particle *weights* for the particle with parameterisation $\vec{\alpha}_p$. For each experiment, we use TL as a measure of how well the *distribution* performed overall, i.e. we care about how well all particles, $\{p\}$, perform as a collective, representative of how well $\Pr(\vec{\alpha})$ approximates the system, equivalent to the noramlisation factor in Eq. (1.5), [20].

l_e are strictly positive, and because the natural logarithm is a monotonically increasing function, we can equivalently work with the log total likelihood (LTL), since $ln(l_a) > ln(l_b) \iff l_a > l_b$. LTL are also beneficial in simplifying calculations, and are less susceptible to system underflow, i.e. very small values of l will exhaust floating point precision, but $ln(l)$ will not.

Note, we know the initial weights are normalised,

$$w_p^0 = \frac{1}{N_p} \implies \sum_p^{N_p} w_p^0 = 1, \tag{1.16}$$

so we can see

$$\begin{aligned} \Pr(d|\vec{\alpha}_p; e) \leq 1 &\implies \Pr(d|\vec{\alpha}_p; e) \times w_p^{\text{old}} \leq w_p^{\text{old}} \\ &\implies \sum_{\{p\}} \Pr(d|\vec{\alpha}_p; e) \times w_p^{\text{old}} \leq \sum_{\{p\}} w_p^{\text{old}} \leq \sum_p^{N_p} w_p^0; \\ &\implies l_e \leq 1. \end{aligned} \tag{1.17}$$

Eq. (1.17) essentially says that a good batch of particles, where on average particles perform well, will mean that most w_i are high, so $l_e \approx 1$. Conversely, a poor batch of particles will have low average w_i, so $l_e \approx 0$.

In order to assess the quality of a *model*, $\hat{H}_i$, we can consider the performance of a set of particles throughout a set of experiments $\mathcal{E}$, through its total log total likelihood (TLTL),

$$\mathcal{L}_i = \sum_{e \in \mathcal{E}} ln(l_e). \tag{1.18}$$

The set of experiments on which $\mathcal{L}_i$ is computed, $\mathcal{E}$, as well as the particles whose sum constitute each l_e, can be the same experiments on which $\hat{H}_i$ is trained, $\mathcal{E}_i$, but in general need not be. That is, $\hat{H}_i$ can be evaluated by considering different experiments than those on which it was trained. For example, $\hat{H}_i$ can be trained with $\mathcal{E}_i$ to optimise $\vec{\alpha}_i'$, and thereafter be evaluated

using a different set of experiments $\mathcal{E}_v$, such that $\mathcal{L}_i$ is computed using particles sampled from the distribution after optimising $\vec{\alpha}$, $\Pr(\vec{\alpha}'_i)$, and may use a different number of particles than the training phase.

Perfect agreement between the model and the system would result in $l_e = 1 \Rightarrow ln(l_e) = 0$, as opposed to imperfect agreement where $l_e < 1 \Rightarrow ln(l_e) < 0$. In all cases Eq. (1.18) is negative, and across a series of experiments, strong agreement gives low $|\mathcal{L}_i|$, whereas weak agreement gives large $|\mathcal{L}_i|$.

1.5 PARAMETER ESTIMATION

QHL is a parameter estimation algorithm, so here we introduce some methods to evaluate its performance, which we can reference in later sections of this thesis. The most obvious meausre of the progression of parameter estimation is the error between the true parameterisation, $\vec{\alpha}_0$, and the approximation $\vec{\alpha} = \text{mean}\left(\Pr(\vec{\alpha})\right)$, which can be captured by a large family of loss functions. Among others, we use the quadratic loss (QL), which captures this error through the sum of the square difference between each parameters' true and estimated values symetrically. We can record the QL at each experiment of our training regime and hence track its over- or under-estimation. The QL is given by

$$L_Q(\vec{\alpha}) = \|\vec{\alpha}_0 - \vec{\alpha}\|^2 \tag{1.19}$$

where $\vec{\alpha}_0$ is the true parameterisation and $\vec{\alpha}$ a hypothesis distribution. An example of the progression of QL throughout QHL is shown in Fig. 1.2.

1.5.1 *Volume*

We also care about the range of parameters supported by $\Pr(\vec{\alpha})$ at each experiment: the volume of the particle distribution can be seen as a proxy for our certainty that the approximation $\text{mean}\left(\Pr(\vec{\alpha})\right)$ is accurate. For example, for a single parameter ω, our best knowledge of the parameter is $\text{mean}\left(\Pr(\omega)\right)$, and our belief in that approximation is the standard deviation of $\Pr(\omega)$; we can think of volume as an n-dimensional generalisation of this intuition [14, 21].

In general, a confidence region, defined by its confidence level κ, is drawn by grouping particles of *high particle density*, $\mathcal{P}$, such that $\sum_{p \in \mathcal{P}} w_p \geq \kappa$. We use the concept of *minimum volume enclosing ellipsoid* to capture the confidence region [21], calculated as in [22], which are characterised by their covariance matrix, Σ, which allows us to calculate the volume,

$$V(\Sigma) = \frac{\pi^{|\vec{\alpha}|/2}}{\Gamma(1 + \frac{|\vec{\alpha}|}{2})} \det\left(\Sigma^{-\frac{1}{2}}\right), \tag{1.20}$$

where Γ is the Gamma function, and $|\vec{\alpha}|$ is the cardinality of the parameterisation. This quantity allows us to meaningfully compare distributions of different dimension, but we must be cautious

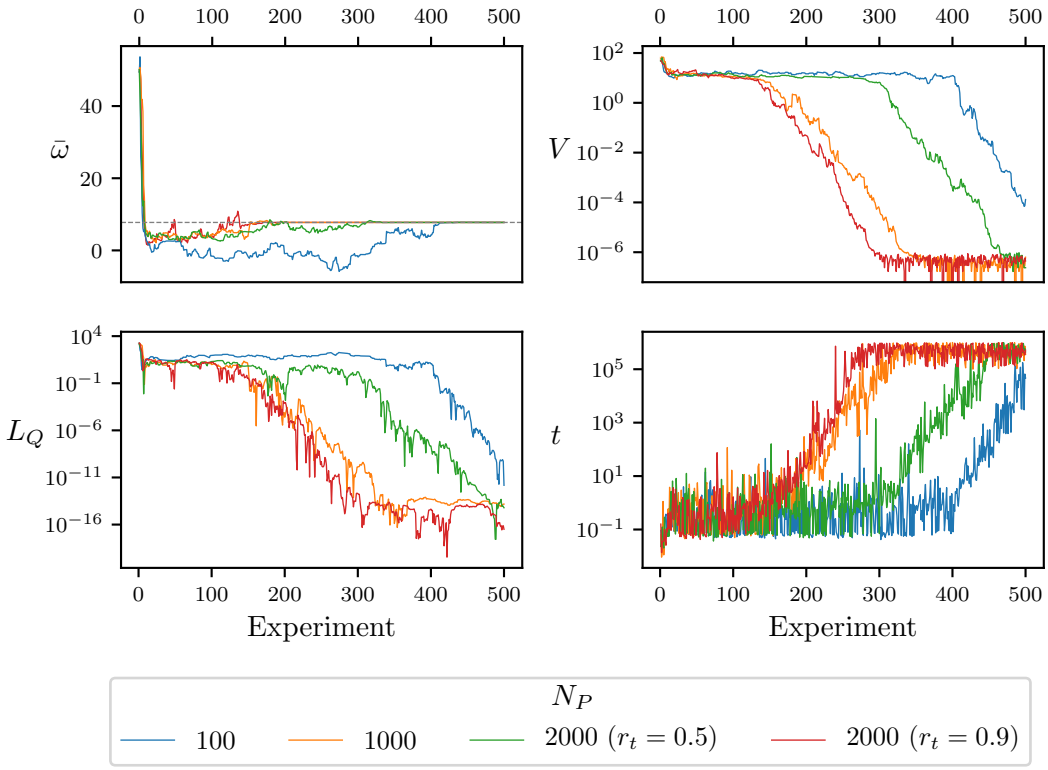

Figure 1.2: Parameter learning for the analyitcal likelihood, Eq. (1.14), for varying numbers of particles N_P, with $N_e = 500$. For $N_p = 2000$, we show the resampler threshold set to $r = 0.5$ and $r = 0.9$. The parameter estimate, i.e. $\bar{\omega}$, the mean of the posterior distribution after each experiment, approaching $\omega_0 = 7.75$ (dashed line), where the prior is centred on $\omega = 50 \pm 25$. For the same experiments, the volume, V, quadratic loss, L_Q, and evolution time, t, are shown. Implementation details are listed in **??**

of drawing strong comparisons between models based on their volume alone, for instance because they may have started from vastly different prior distributions.

Within SMC, we assume the credible region is simply the posterior distribution, such that we can take $\Sigma = \text{cov}(\Pr(\vec{\alpha}))$ after each experiment, and hence track the uncertainty in our parameters across the training experiments [5]. We use volume as a measure of the learning procedure's progress: slowly decreasing or static volume indicates poor learning, e.g. the blue and green models in Fig. 1.2, possibly highlighting poor experiment design. Exponentially decreasing volume indicates that the parameters' estimation is improving, e.g. the first 300 experiments for the red model in Fig. 1.2, whereas converged volume (the latter 200 experiments) indicate the learning has saturated and there is little benefit to running further experiments.

1.6 EXPERIMENT DESIGN HEURISTIC

A key consideration in QHL is the choice of experimental controls implemented in attempt to learn from the system. The experimental controls required are dictated by the choice of likelihood function used within SMC, though typically there are two primary controls we will focus on: the evolution time, t, and the *probe* state evolved, $|\psi\rangle$. The design of experiments is handled by an experiment design heuristic (EDH), whose structure can be altered to suit the user's needs, with respect to the individual target system. Usually, the EDH attempts to exploit the information available, adaptively accounting for some aspects of the inference process performed already. In some cases, however, there may be justification to employ a non-adaptive schedule, for instance to force QHL to train upon a full set of experimental data rather than a subset, as an adaptive method may advise. We can categorise each EDH as either *online* of *offline*, depending on whether it accounts for the current state of the inference procedure, i.e. the posterior. The EDH is modular and can be replaced by any method that returns a valid set of experimental controls, so we can consider numerous approaches, for instance those described in [23, 24].

1.6.1 *Particle guess heuristic*

The default EDH is the particle guess heuristic (PGH) [7], an online method which attempts to design the optimal evolution time based on the posterior at each experiment. Note PGH does not specify the probe, so is coupled with a probe selection routine to comprise a complete EDH.

The principle of PGH is that the uncertainty of the posterior limits how well the Hamiltonian is currently approximated[5], and therefore limits the evolution time for which the posterior can be expected to reasonably mimic $\hat{H}_0$. For example, consider Eq. (1.10) with a single parameter $\omega_0 = 10$, and current $\{\text{mean}\left(\Pr(\omega)\right) = 9, \text{std}\left(\Pr(\omega)\right) = 2\}$: we can expect that the approximation $\omega' = \text{mean}\left(\Pr(\omega)\right)$ is valid up to $t_{max} \approx {}^{1}/_{\text{std}(\Pr(\omega))}$. It is sensible, then, to use $t \sim t_{max}$ for two reasons: (i) smaller times are already well explained by the posterior, so offer little opportunity to learn; (ii) t_{max} is at or near the threshold which particles sampled from the posterior can comfortably explain, so it will expose the relative difference in likelihood between the posterior's better and worse particles, providing a capacity to learn. Informally, as the uncertainty in the posterior shrinks, PGH selects larger times to ensure the training is based on informative experiments while simultaneously increasing certainty about the paraemters. In the one-dimensional case, this logic can be used to find an optimal time heuristic, where experiment k is assigned $t_k = {}^{1.26}/_{\text{std}(\Pr(\omega))}$ [19].

For a general multidimensional parameterisation, rather than directly using the inverse of the standard deviation of $\Pr(\vec{\alpha})$, which relies on the expensive calculation of the covarinace matrix,

5 The reasoning behind limiting the evolution time according to the posterior distribution is rooted in the effect of the Loschmidt echo, described in Section 1.3.1.

PGH uses a proxy whereby two particles are sampled from $\Pr(\vec{\alpha})$. The experimental evolution time for experiment k is then given by

$$t_k = \frac{1}{\|\vec{\alpha}_i - \vec{\alpha}_j\|}, \tag{1.21}$$

where $\vec{\alpha}_i, \vec{\alpha}_j$ are distinct particles sampled from $\mathcal{P}$ where $\mathcal{P}$ is the set of particles under consideration by SMC after experiment $k-1$, which had been recently sampled from $\Pr(\vec{\alpha})$.

1.6.2 *Alternative experiment design heuristics*

The EDH can be specified to the requirements of the target system; we test four examples of customised EDHs against four target Hamiltonians. Here the EDH must only design the evolution time for the experiment, with probe design discussed in the next section. The heuristics tested are:

- Random$(0, t_{max})$: Randomly chosen time up to some arbitrary maximum, we set $t_{max} = 1000$ (arbitrary units). This approach is clearly subobtimal, since it does not account whatsoever for the knowledge of the training so far, and demands the user choose a suitable t_{max}, which can be not gauranteed to be meaningful.
- t list: forcing the training to consider a set of times decided in advance. For instance, when only a small set of experimental measurements are available, it is sensible to train on all of them, perhaps repeatedly. We test uniformly spaced times $t \in (0, t_{max}]$, and cycle through the list twice, aiming first to broadly learn the region of highest likelihood for all times, and then to refine the approximation. Again this EDH fails to account for the performance of the trainer so far, so may use times either far above or below the ability of the parameterisation.
- $(9/8)^k$: An early attempt to match the expected exponential decrease in volume from the training, was to set $t_k = (9/8)^k$ [5]. Note we increment k after 10 experiments in the training regime, rather than after each experiment, which would result in extremely high times which flood CPU memory.
- PGH: as described in Section 1.6.1.

We demonstrate the influence of the EDH on the training procedure by testing models[6] of various complexity and dimension in Fig. 1.3. In particular, we first test a simple 1-qubit model, Eq. (1.22a); followed by more complicated 1-qubit model, Eq. (1.22b); as well as randomly generated 5-qubit Ising, Eq. (1.22c), and 4-qubit Heisenberg models, Eq. (1.22d). Each $\hat{H}_i$ have randomly chosen parameters implicitly assigned to each term.

$$\hat{H}_1 = \hat{\sigma}_1^z \tag{1.22a}$$

$$\hat{H}_2 = \hat{\sigma}_1^x + \hat{\sigma}_1^y + \hat{\sigma}_1^z \tag{1.22b}$$

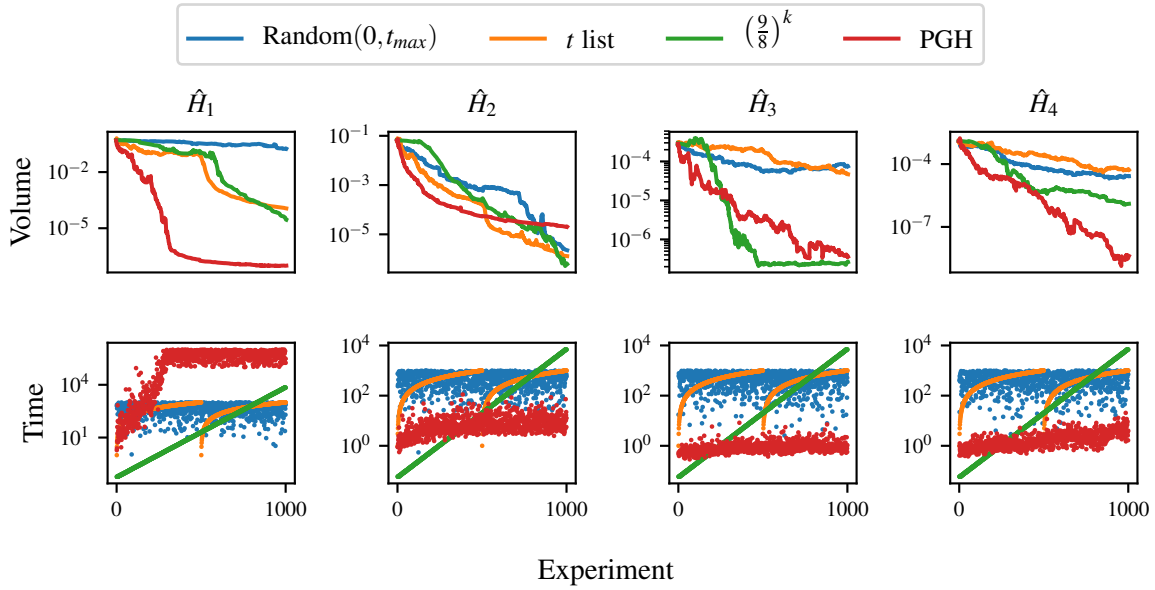

Figure 1.3: The volume (top) and evolution times (bottom) of various models when trained through QHL using different EDHs. We show models of various complexity and dimension, each trained using four heuristics, outlined in the main text. Implementation details are listed in **??**

$$\hat{H}_3 = \hat{\sigma}_1^z\hat{\sigma}_3^z + \hat{\sigma}_1^z\hat{\sigma}_4^z + \hat{\sigma}_1^z\hat{\sigma}_5^z + \hat{\sigma}_2^z\hat{\sigma}_4^z + \hat{\sigma}_2^z\hat{\sigma}_5^z + \hat{\sigma}_3^z + \hat{\sigma}_4^z + \hat{\sigma}_3^z + \hat{\sigma}_5^z \tag{1.22c}$$

$$\hat{H}_4 = \hat{\sigma}_1^z\hat{\sigma}_2^z + \hat{\sigma}_1^z\hat{\sigma}_3^z + \hat{\sigma}_2^x\hat{\sigma}_3^x + \hat{\sigma}_2^z\hat{\sigma}_3^z + \hat{\sigma}_2^x\hat{\sigma}_4^x + \hat{\sigma}_3^z\hat{\sigma}_4^z \tag{1.22d}$$

We show the performance of each of the listed EDHs in Fig. 1.3. The general trend reveals that, although some individual models benefit from bespoke EDHs, the PGH is generically applicable and usually facilitates a reasonable level of training, without providing advantage to any model. We will have cause to use alternative EDHs in particular circumstances, but we adopt PGH as the default EDH throughout this thesis, unless otherwise stated.

1.7 PROBE SELECTION

A final consideration about training experiments within QHL is the choice of input probe state, $|\psi\rangle$, which is evolved in the course of finding the likelihood used during the Bayesian update. We can consider the choice of probe as an output of the EDH, although previous work has usually not considered optimising the probe, instead usually setting $|\psi\rangle = |+\rangle^{\otimes n}$ for n qubits [8, 19]. In

6 Note the models designed here are not intended to represent physically meaningful situations, but merely to serve as examples of simulatable Hamiltonians.

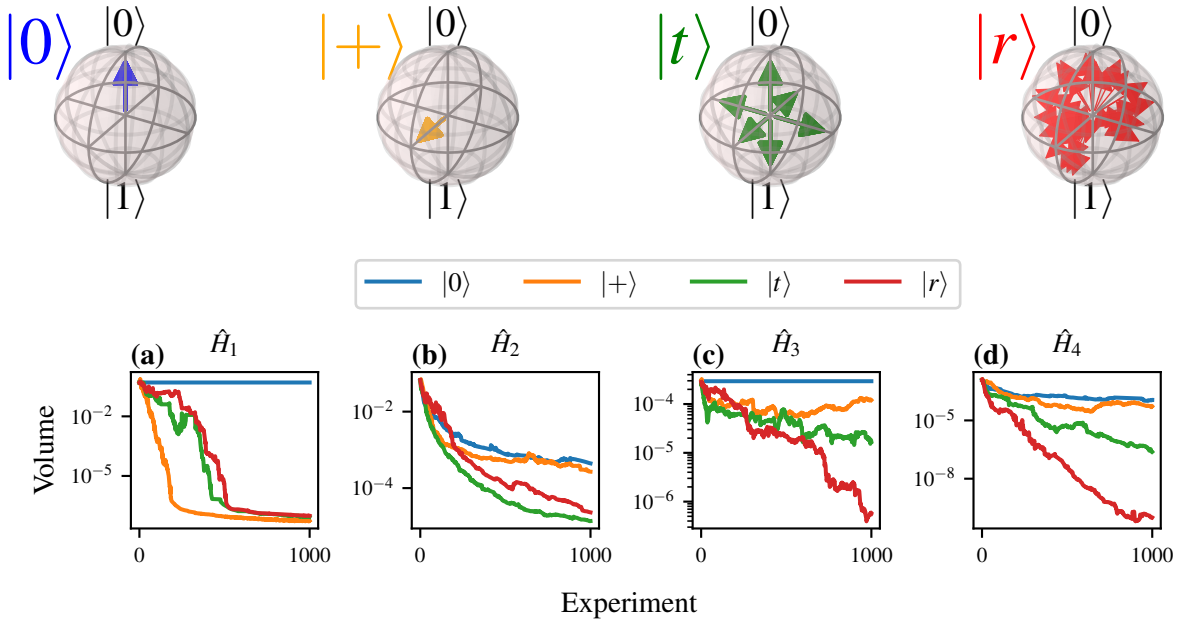

Figure 1.4: Training through QHL with varying probes. **Top,** Probes used, $\Psi = \{|0\rangle^{\otimes n}\}$ (blue); $\Psi = \{|+\rangle^{\otimes n}\}$ (orange); Ψ constructed from tomographic probes (green); Ψ random (red). We show the 1-qubit probes on the Bloch sphere, though probes are constructed up to n-qubits in each case. **Bottom,** Volume of various models, listed in Eq. (1.22), trained through QHL using different initial probe sets. In each case the probes are generated for arbitrary numbers of qubits; for $|0\rangle, |+\rangle$, the number of probes generated is $N_\psi = 1$, and for $|t\rangle, |r\rangle$, $N_\psi = 40$. Implementation details are listed in **??**

principle it is possible for the EDH to design a new probe at each experiment, although a more straightforward approach is to compose a set of probes offline, $\Psi = \{|\psi\rangle\}$, of size $N_\psi = |\Psi|$. Then, a probe is chosen at each experiment from Ψ, allowing for the same $|\psi\rangle$ to be used for multiple experiments within the training, e.g. by iterating over Ψ. Ψ can be generated with respect to the individual learning problem as we will examine later[7], but it is usually sufficient to use generic strategies which should work for all models; some straightforward examples are

i. $|0\rangle$: $\Psi = \{|0\rangle^{\otimes n}\}$, $N_\psi = 1$;

ii. $|+\rangle$: $\Psi = \{|+\rangle^{\otimes n}\}$, $N_\psi = 1$;

iii. $|t\rangle$: Ψ is a random subset of probes generated by combining tomographic basis states, $N_\psi = 40$;

iv. $|r\rangle$: $|\psi\rangle$ are random, separable probes, $N_\psi = 40$.

7 In **??**

Recalling the set of models from Eq. (1.22), we test each of these probe construction strategies in Fig. 1.4. We can draw a number of useful observations from these simple tests:

- Training on an eigenstate – as in the case for $\hat{H}_1$ and $\hat{H}_3$ using $|0\rangle$ – yields no information gain. This is because all particles give likelihoods $l = 1$, so no weight update can occur, meaning the parameter distribution does not change when presented new evidence.
- Training on an even superposition of the model's eigenstates – e.g. $|+\rangle$ for $\hat{H}_1$ – is maximally informative: any deviations from the true parameterisation are registered most dramatically in this basis, providing the optimal training probe for this case.
- These observations are reinforced by Fig. 1.4**c**, where a 5-qubit Ising model also fails to learn from one of its eigenstates, $|0\rangle^{\otimes 5}$. Of note, however, is that $|+\rangle^{\otimes 5}$ is not the strongest probe here: the much larger Hilbert space here can not be scanned sufficiently using a single probe; using a larger number of probes is more effective, even if those are randomly chosen.
- In general the tomographic and random probe sets perform reliably, even for complex models.

It is an open challenge to identify the optimal probe for training any given model; the design of informative probes could be built into the EDH in principle, e.g. a set of probes could be generated of even superpositions of the candidate's eigenstates. However, for model comparison purposes in general, it is helpful to have a universal set of probes, Ψ, upon which all models are trained. The use of Ψ minimises systematic bias towards particlar models, which might arise from probes which serves as favourable bases for a subset of models, for example $|+\rangle$ in Fig. 1.4**(a)**. Careful consideration should be given to N_ψ in the choice of the probe generator, since it is important to ensure probes robustly test the parameterisation across the entire Hilbert space. It is also necessary that SMC has sufficient opportunity to learn within a given subspace before moving to the next, so that slight deviations in $\Pr(\vec{\alpha})$ due to a single probe are not immediately reversed because a distant probe is immediately invoked. We can mitigate this concern by instructing the EDH to repeatedly select a probe from Ψ for a batch of successive experiments, before moving to the next available probe. Unless otherwise stated, for the remainder of this thesis we will adopt the random probe generator as the defualt mechanism for selecting probes, iterating between probes after batches of 5 experiments.

2

QUANTUM MODEL LEARNING AGENT

A *model* is the mathematical description of a quantum system of interest, Q. In Chapter 1, we discussed a number of systems in terms of their Hamiltonian descriptions, although in general the description of a quantum system need not be Hamiltonian, e.g. Lindbladian models describe open quantum systems, so we will generically refer to the *model* of Q throughout.

Quantum Model Learning Agent (QMLA) is an algorithm that builds upon the concept at the heart of Chapter 1, i.e. applying machine learning (ML) to the characterisation of Hamiltonians. The extension, and central question of QMLA is: if we do not know the structure of the model which describes a target quantum system, can we still learn about the physics of the system? That is, we remove the assumption about the form of the Hamiltonian model, and attempt to uncover which *terms* constitute the Hamiltonian, and in so doing, learn the interactions the system is subject to.

For the remainder of this thesis, our objective is to learn the model underlying a series of target quantum systems. We will first introduce some concepts which will prove useful when discussing QMLA, before describing the protocol in detail in Section 2.3.

2.1 MODELS

Models are simply the mathematical objects which can be used to predict the behaviour of a system. In this thesis, models are synonymous with Hamiltonians, composed of a set of *terms*, $\mathcal{T} = \{\hat{t}\}$, where each $\hat{t}$ is a matrix. Each term is assosiated with a multiplicative scalar, which may be referred to as that term's *parameter*: we impose order on the terms and parameters such that we can succinctly summarise any model as

$$\hat{H} = \begin{pmatrix} \alpha_0 & \dots & \alpha_n \end{pmatrix} \begin{pmatrix} \hat{t}_1 \\ \vdots \\ \hat{t}_n \end{pmatrix} = \vec{\alpha} \cdot \vec{T}, \tag{2.1}$$

where $\vec{\alpha}, \vec{T}$ are the model's parameters and terms, respectively.

For example, a model which is the sum of the (non-identity) Pauli operators is given by

$$\begin{aligned} \hat{H} &= \begin{pmatrix} \alpha_x & \alpha_y & \alpha_z \end{pmatrix} \cdot \begin{pmatrix} \hat{\sigma}_x \\ \hat{\sigma}_y \\ \hat{\sigma}_z \end{pmatrix} \\ &= \alpha_x \hat{\sigma}_x + \alpha_y \hat{\sigma}_y + \alpha_z \hat{\sigma}_z \\ &= \begin{pmatrix} \alpha_z & \alpha_x - i\alpha_y \\ \alpha_x + i\alpha_y & \alpha_z \end{pmatrix}. \end{aligned} \tag{2.2}$$

Through this formalism, we can say that the sole task of quantum Hamiltonian learning (QHL) was to optimise $\vec{\alpha}$, given $\vec{T}$. The principal task of QMLA is to identify the terms $\vec{T}$ which are supported by the most statistical evidence as describing a target system Q. In short, QMLA proposes *candidate models*, $\hat{H}_i$, as hypotheses to explain Q; we *train* each model independently through a parameter learning routine, and finally nominate the model with the best performance after training. In particular, QMLA uses QHL as the parameter learning *subroutine*, but in principle this step can be performed by any algorithm which learns $\vec{\alpha}$ for given $\vec{T}$, [25–33]. While discussing a model $\hat{H}_i$, their *training* then simply means the implementation of QHL[1], where $\hat{H}_i$ is *assumed* to represent Q, such that $\vec{\alpha}_i$ is optimised as well as it can be, even in the case it is entirely inaccurate, $\hat{H}_i \not\approx \hat{H}_0$.

2.2 BAYES FACTORS

We can use the tools introduced in Section 1.4 to *compare* candidate models. Of course it is first necessary to ensure that each model has been adequately trained: while inaccurate models are unlikely to strongly capture the system dynamics, they should first train on the system to determine their best attempt at doing so, i.e. they should undergo the process in Chapter 1. It is statistically meaningful to compare models via their total log total likelihood (TLTL), $\mathcal{L}_i$, if and only if they have considered the same data, i.e. if models have each attempted to account for the same set of experiments, $\mathcal{E}$ [34].

We can then exploit direct pairwise comparisons between models, by imposing that both models' TLTL are computed based on *any* shared set of experiments $\mathcal{E}$, with corresponding measurements $\mathcal{D} = \{d_e\}_{e\in\mathcal{E}}$. Pairwise comparisons can then be quantified by the Bayes factor (BF),

$$B_{ij} = \frac{\Pr(\mathcal{D}|\hat{H}_i;\mathcal{E})}{\Pr(\mathcal{D}|\hat{H}_j;\mathcal{E})}. \tag{2.3}$$

Intuitively, we see that the BF is the ratio of the likelihood, i.e. the performance, of model $\hat{H}_i$'s attempt to account for the data set $\mathcal{D}$ observed following the experiment set $\mathcal{E}$, against the same likelihood for model $\hat{H}_j$. BFs are known to be statistically significative of the stronger model from a pair when both models attempt to explain observed data, while favouring models of low cardinality, thereby supressing overfitting models.

We have that, for independent experiments, and recalling Eq. (1.15),

$$\begin{aligned}\Pr(\mathcal{D}|\hat{H}_i;\mathcal{E}) &= \Pr(d_n|\hat{H}_i;e_n)\times\Pr(d_{n-1}|\hat{H}_i;e_{n-1})\times\cdots\times\Pr(d_0|\hat{H}_i;e_0)\\ &= \prod_{e\in\mathcal{E}}\Pr(d_e|\hat{H}_i;e)\\ &= \prod_{e\in\mathcal{E}}(l_e)_i.\end{aligned} \tag{2.4}$$

1 Or the chosen parameter learning subroutine.

We also have, from Eq. (1.18)

$$\begin{aligned} \mathcal{L}_i &= \sum_{e\in\mathcal{E}} \ln\left((l_e)_i\right) \\ \implies e^{\mathcal{L}_i} &= \exp\left(\sum_{e\in\mathcal{E}} \ln\left[(l_e)_i\right]\right) = \prod_{e\in\mathcal{E}} \exp\left(\ln\left[(l_e)_i\right]\right) = \prod_{e\in\mathcal{E}} (l_e)_i. \end{aligned} \tag{2.5}$$

So we can write

$$B_{ij} = \frac{\Pr(\mathcal{D}|\hat{H}_i;\mathcal{E})}{\Pr(\mathcal{D}|\hat{H}_j;\mathcal{E})} = \frac{\prod_{e\in\mathcal{E}} (l_e)_i}{\prod_{e\in\mathcal{E}} (l_e)_j} = \frac{e^{\mathcal{L}_i}}{e^{\mathcal{L}_j}} \tag{2.6}$$

$$\implies B_{ij} = e^{\mathcal{L}_i - \mathcal{L}_j} \tag{2.7}$$

This is simply the exponential of the difference between two models' TLTLs when presented the same set of experiments. Intuitively, if $\hat{H}_i$ performs well, and therefore has a high TLTL, $\mathcal{L}_i = -10$, and $\hat{H}_j$ performs worse with $\mathcal{L}_j = -100$, then $B_{ij} = e^{-10-(-100)} = e^{90} \gg 1$. Conversely for $\mathcal{L}_i = -100, \mathcal{L}_j = -10$, then $B_{ij} = e^{-90} \ll 1$. Therfore $|B_{ij}|$ is the strength of the statistical evidence in favour of the interpretation

$$\begin{cases} B_{ij} > 1 & \Rightarrow \hat{H}_i \text{ favoured over } \hat{H}_j \\ B_{ij} < 1 & \Rightarrow \hat{H}_j \text{ favoured than } \hat{H}_i \\ B_{ij} = 1 & \Rightarrow \hat{H}_i, \hat{H}_j \text{ equally favoured.} \end{cases} \tag{2.8}$$

Throughout this thesis, Eq. (2.8) will be used to inform algorithmic preferences towards models based on pairwise comparisons. For example, for a fixed set of models, we can compute BFs between all pairs of models, and allocate a single point to the favoured model from each comparison, after which we can deem the model with most points as the best model from the set.

2.2.1 *Experiment sets*

As mentioned, it is necessary for the TLTL of both models in a BF calculation to refer to the same set of experiments, $\mathcal{E}$. There are a number of ways to achieve this, which we briefly summarise here for reference later.

During training (the QHL subroutine), candidate model $\hat{H}_i$ is trained against $\mathcal{E}_i$, designed by an experiment design heuristic (EDH) to optimise parameter learning specifically for $\hat{H}_i$; likewise $\hat{H}_j$ is trained on $\mathcal{E}_j$. The simplest method to compute the BF is to enforce $\mathcal{E} = \mathcal{E}_i \cup \mathcal{E}_j$ in Eq. (2.3), i.e. to cross-train $\hat{H}_i$ using the data designed specifically for training $\hat{H}_j$, and vice versa. This is a valid approach because it challenges each model to attempt to explain experiments designed explicitly for its competitor, at which only truly accurate models are likely to succeed.

A second approach builds on the first, but incorporates *burn–in* time in the training regime: this is a standard technique in the evaluation of ML models whereby its earliest iterations are discounted for evaluation so as not to skew its metrics, ensuring the evaluation reflects the strength of the model. In BF, we achieve this by basing the TLTL only on a subset of the training experiments. For example, the latter half of experiments designed during the training of $\hat{H}_i$, $\mathcal{E}_i'$. This does not result in less predictive BF, since we are merely removing the noisy segments of the training for each model, e.g. the first half of experiments in Fig. 1.2. Moreover it provides a benefit in reducing the computational requirements: updating each model to ensure the TLTL is based on $\mathcal{E}' = \mathcal{E}_i' \cup \mathcal{E}_j'$ requires only half the computation time, which can be further reduced by lowering the number of particles used during the update, N_p', which will give a similar result as using N_P, assuming the posterior has mostly converged[2].

A final option is to design a set of *evaluation* experiments, $\mathcal{E}_v$, that are valid for a broad variety of models, and so will not favour any particular model. Again, this is a common technique in ML: to use one set of data for training models, and a second, unseen dataset for evaluation. This is a favourable approach: provided for each model we compute Eq. (1.18) using $\mathcal{E}_v$, we can automatically select the strongest model based solely on their TLTLs, meaning we do not have to perform further computationally-expensive updates, as required to cross-train on opponents' experiments during BF calculation. However, it does impose on the user to design a *fair* $\mathcal{E}_v$, requiring unbiased probe states $\{|\psi\rangle\}$ and times $\{t\}$ on a timescale which is meaningful to the system under consideration. For example, experiments with $t > T_2$, where T_2 is the decoherence time of the system, would result in measurements which offer little information, and hence it would be difficult to extract evidence in favour of any model from experiments in this domain. It is difficult to know, or even estimate, such meaningful time scales a priori, so it is difficult for a user to design $\mathcal{E}_v$. Additionally, the training regime each model undergoes during QHL is designed to provide adaptive experiments that take into account the specific model entertained, to choose an optimal set of evolution times, so it is likely that the set of times in $\mathcal{E}_i$ is *reasonable* by default. This approach would be favoured in principle, in the case where such constraints can be accounted for, e.g. an experiment repeated in a laboratory where the available probe states are limited and the timescale achievable is understood.

2.3 QUANTUM MODEL LEARNING AGENT PROTOCOL

Given a target quantum system, Q, described by some *true model*, i.e. its Hamiltonian $\hat{H}_0$, QMLA distills a model $\hat{H}' \approx \hat{H}_0$. We can think of QMLA as a forest search algorithm[3]: consisting of a number of trees, each of which can have an arbitrary number of branches, where each leaf on each branch is an individual model. QMLA is the search for the leaf in the forest with the strongest statistical evidence of representing Q. Each tree in the QMLA forest corresponds

2 We will verify this claim in **??**, in the context of real examples.

3 Note QMLA is not a random forest, where decision trees are added at random, because in QMLA trees are highly structured and included manually.

to an independent *model search* , structured according to a bespoke exploration strategy (ES), which we detail in Section 2.4.

In short, the components of the iterative model search for a given ES, depicted in Fig. 2.1**(a-d)**, are

BRANCHES

A set of candidate models, $\{\hat{H}_i\}$, are held together on a branch, μ.

TRAINING

Each model $\hat{H}_i \in \mu$ is trained according to a parameter learning subroutine.

CONSOLIDATION

The performance of candidates in μ are ranked relative to each other, such that some models are favoured over others, for instance through selection of a *branch champion*, $\hat{H}^{\mu}_{C}$. Consolidation can rely on any statistical test, with BFs providing a robust platform to disinguish any pair of candidates.

SPAWN

A set of new models are constructed, accounting for the consolidation stage immediately beforehand, i.e. leveraging the best-yet-known models to construct improved hypotheses.

Multiple model searches can proceed in parallel, and they are each assigned an independent exploration tree (ET), S. Following the iterative model generation procedure, the protocol selects the strongest considered candidate, for instance by consolidating the set of branch champions, $\{\hat{H}^{\mu}_{C}\}$, resulting in the nomination of a single *tree champion*, $\hat{H}'_{S}$, Fig. 2.1**(e)**. The final step of QMLA is then to consolidate the set of champion models from all ETs, $\{\hat{H}'_{S}\}$, in order to delcare a *global* champion model, $\hat{H}'$, Fig. 2.1**(f)**.

2.4 EXPLORATION STRATEGIES

QMLA is implemented by running N_t ETs concurrently, where each ET corresponds to a unique model search and ultimately nominates a single model as its favoured approximation of $\hat{H}_0$. An exploration strategy (ES) is the set of rules which guide a single ET throughout its model search. We elucidate the responsibilities of ESs in the remainder of this section, but in short they can be summarised as:

i. model generation: combining the knowledge progressively acquired on the ET to construct new candidate models;

ii. decision criteria for the model search phase: instructions for how QMLA should respond at predefined junctions, e.g. whether to cease the model search after a branch has completed;

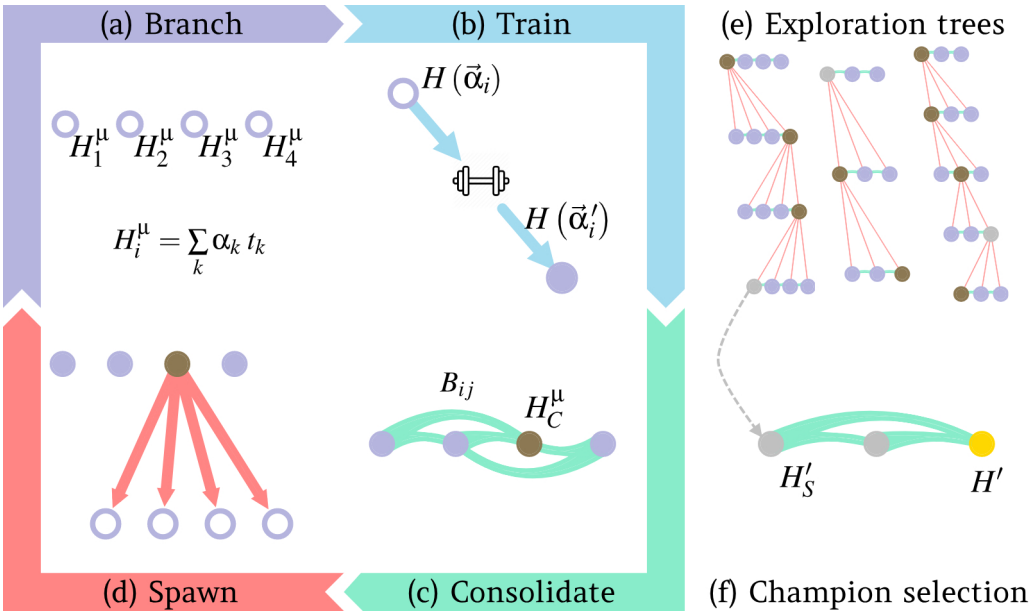

Figure 2.1: Schematic of Quantum Model Learning Agent (QMLA). **a-d,** Model search phase within an exploration strategy (ES). **a,** Models are placed as (empty, purple) nodes on the *active branch* μ, where each model is a sum of terms $\hat{t}_k$ multiplied by corrresponding scalar parameters α_k. **b,** Each active model is trained according to a subroutine such as quantum Hamiltonian learning to optimise $\vec{\alpha}_i$, resulting in the trained $\hat{H}(\vec{\alpha}_i')$ (filled purple node). **c,** μ is consolidated, i.e. models are evaluated relative to other models on μ, according to the consolidation mechanism specificed by the ES. In this example, pairwise Bayes factors, B_{ij}, between $\hat{H}_i, \hat{H}_j$ are computed, resulting in the election of a single branch champion $\hat{H}_C^\mu$ (bronze). **d,** A new set of models are *spawned* according to the chosen ES's model generation strategy. In this example, models are spawned from a single parent. The newly spawned models are placed on the next branch, $\mu + 1$, iterating back to **(a)**. **e-f,** Higher level of entire QMLA procedure. **e,** The model search phase for a unique ES is presented on an *exploration tree*. Multiple ES can operate in parallel, e.g. assuming different underlying physics, so the overall QMLA procedure involves a *forest search* across multiple ETs. Each ES nominates a champion, $\hat{H}_S'$ (silver), after consolidating its branch champions (bronze). **f,** $\hat{H}_S'$ from each of the above ETs are gathered on a single branch, which is consolidated to give the final champion model, $\hat{H}'$ (gold).

iii. true model specification: detailing the terms and parameters which constitute $\hat{H}_0$ (in the case where Q is simulated);

iv. modular functionality: subroutines called throughout QMLA are interchangeable such that each ES specifies the set of functions to achieve its goals.

QMLA acts in tandem with one or more ESs, through the process depicted in Fig. 2.2. In summary: QMLA sends a request to the ES for a set of models; the ES designs models and places them as leaves on a new branch on its ET, and returns the set $\mathbb{H}$; QMLA trains the models

in $\mathbb{H}$; QMLA consolidates $\mathbb{H}$; QMLA informs the ES of the results of training/consolidation of $\mathbb{H}$; ES decides whether to continue the search, and informs QMLA.

2.4.1 *Model generation*

The main role of any ES is to design candidate models to test against $\hat{H}_0$. This can be done through any means deemed appropriate, although in general it is sensible to exploit the information gleaned so far in the ET, such as the performance of previous candidates and their comparisons, so that successful models are seen to *spawn* new models, e.g. by combining previously successful models, or by building upon them. Conversely, model generation can be completely determined in advance, or entirely random. This alludes to the central design choice in composing an ES: how broad and deep should the searchable *model space* be, considering that adequately training each model is expensive, and that model comparisons are similarly expensive. The size of the model space can usually be easily found by assuming that terms are binary – either the interaction they represent is present or not. If all possible terms are accounted for, and the total set of terms is $\mathcal{T}$, then there are $2^{|\mathcal{T}|}$ available candidates in the model space. The model space encompasses the closed[4] set of models construable by the set of terms considered by an ES. Because training models is slow in general, a central aim of QMLA is to search this space efficiently, i.e. to minimise the number of models considered, while retaining high quality models and providing a reasonable prospect of uncovering the true model, or a strong approximation thereof.

2.4.2 *Decision criteria for the model search phase*

Further control parameters, which direct the growth of the ET, are set within the ES. At several junctions within Algorithms 1 to 2, QMLA queries the ES in order to decide what happens next. Here we list the important cases of this behaviour.

PARAMETER-LEARNING SETTINGS

- such as the prior distribution to assign each parameter during QHL, and the parameters needed to run sequential Monte Carlo (SMC);
- the time scale on which to examine Q;
- the input probes to train upon, Ψ, described in Section 1.7.

BRANCH CONSOLIDATION STRATEGY

4 It is feasible to define an ES which uses an open model space, that is, there is no pre-defined $\mathcal{T}$, but rather the ES determines models through some other heuristic mechanism. In this thesis, we do not propose any such ES, but note that the QMLA framework facilitates the concept, see **??**.

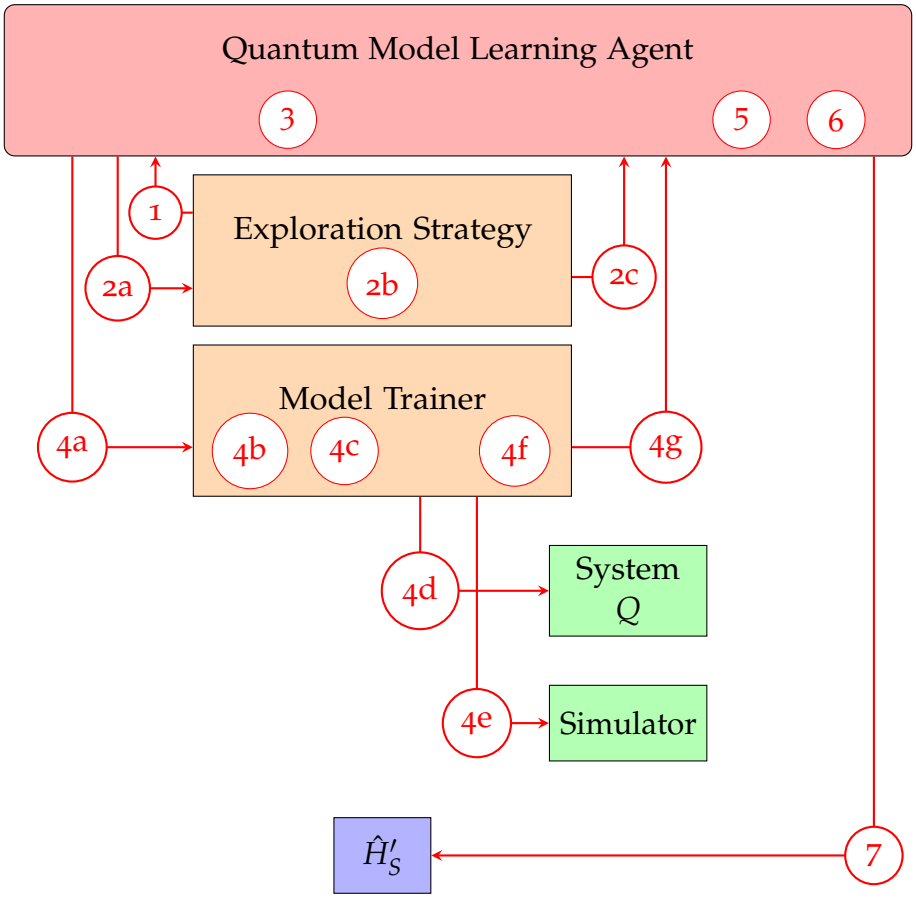

Figure 2.2: Interface between Quantum Model Learning Agent (QMLA) and a single exploration strategy (ES). The main components are the ES, model training subroutine, target quantum system Q, and (quantum) simulator. The main steps of the algorithm, shown in red with arrows denoting data transferred during that step, are as follows. **1**, QMLA retrieves decision infrastructure from ES, such as the consolidation mechanism and termination criteria. **2**, models are designed/spawned; **2a**, QMLA signals to ES requesting a set of models, passing the results of the previous branch's models if appropriate. **2b**, ES spawns new models, $\mathbb{H}$; **2c**, ES passes $\mathbb{H}$ to QMLA. **3**, QMLA assigns a new branch $(\mu \leftarrow \mu + 1)$ and places the newly proposed models upon it. **4**, Model training subroutine (here quantum Hamiltonian learning), performed independently for each model $\hat{H}_i \in \mu$; **4a**, QMLA passes $\hat{H}_i$ to the model trainer; **4b**, construct a prior distribution $\Pr_i(\vec{\alpha})$ describing the model's parameterisation $\vec{\alpha}_i$; **4c**, design experiment e to perform on Q to optimise $\vec{\alpha}_i$; **4d**, perform e on Q to retrieve a datum d; **4e**, simulate e for particles $\{\vec{\alpha}_1, \ldots, \vec{\alpha}_{N_p}\}$ sampled from $\Pr_i(\vec{\alpha})$ to retrieve likelihoods for each particle $\{l_e^j\}_{j\in(1,\ldots N_P)}$; **4e**, update the prior $\Pr_i(\vec{\alpha})$ based on $\{(d, l_e^j)\}_{j\in(1,\ldots,N_P)}$. **5**, Evaluate and rank $\hat{H}_i \in \mu$ according to the ES's consolidation mechanism. **6**, Check ES's termination criteria; if reached, proceed to **(7)**, otherwise return to **(2)**. **7**, Nominate champion model, $\hat{H}'_S$.

– How to consolidate models within a branch. Some examples used in this work are:

* a points-ranking, where all candidates are compared via BF, and points are assigned to the favoured model in each case, according to Eq. (2.8);

* ranking reflecting each model's log-likelihood (Eq. (2.5)) after training;

* models are ranked according to some objective function, as in the case of genetic algorithms (GAs) which we detail in **??**.

MODEL SEARCH TERMINATION CRITERIA

– For example, instruction to stop after a fixed number of iterations, or when a certain fitness has been reached.

CHAMPION NOMINATION

– when a single ET is explored, identify a single champion model from the branch champions, $\{\hat{H}_C^\mu\}$;

– if multiple ETs are explored, the mechanism to compare champions across trees, $\{\hat{H}_S'\}$.

2.4.3 *True model specification*

It is necessary also to specify details about the true model, $\hat{H}_0$, at least in the case where QMLA acts on simulated data. Within the ES, we can set $\vec{T}_0$ as well as $\vec{\alpha}_0$. For example where the target system is an untrusted quantum simulator to be characterised, S_u, by interfacing with a trusted (quantum) simulator S_t, we decide some $\hat{H}_0$ in advance: the model training subroutine calls for likelihoods, those corresponding to $\hat{H}_0$ are computed S_u, while particles' likelihoods are computed on S_t.

2.4.4 *Modular functionality*

Finally, there are a number of fundamental subroutines which are called upon throughout the QMLA algorithm. These are written independently such that each subroutine has a number of available implementations. These can be chosen to match the requirements of the user, and are set via the ES.

MODEL TRAINING PROCEDURE

Subroutine used to optimise parameters for candidate models.

– i.e. whether to use QHL or quantum process tomography, etc.

– In this work we always use QHL.

LIKELIHOOD FUNCTION

The method used to estimate the likelihood for use during quantum likelihood estimation (QLE) within QHL, which ultimately depends on the measurement scheme.

- The role of these functions is to compute the probability of measuring each experimental outcome.
- These functions compute the *expectation value* of the unitary operator, $e^{-i\hat{H}t}$, corresponding to the dynamics of either Q or the hypothesis model.
- By default, we use projective measurement back onto the input probe state, $|\langle\psi| e^{-i\hat{H}t} |\psi\rangle|^2$.
- In the usual case where Q has binary outcomes, we label one outcome – say, measurement in the $|+\rangle$ state – as $d = 0$ and compute $\Pr(0)$ so that the likelihood, expectation value and $\Pr(0)$ refer to the same quantity, see Section 1.3.
- It is possible instead to implement any measurement procedure, for example an experimental procedure where the environment is traced out, as we address in **??**.

PROBE

Defining the input probes to be used during training, Ψ, see Section 1.7.

- In general it is preferable to use numerous probes in order to avoid biasing particular terms.
- In some cases we are restricted to a small number of available input probes, e.g. to match experimental constraints.

EXPERIMENT DESIGN HEURISTIC

The method through which to design bespoke experiments to maximise the information on which models are individually trained, described in Section 1.6.

- In particular, in this work the experimental controls consist solely of $\{|\psi\rangle, t\}$.
- Currently, probes are generated offline, but in principle it is feasible to choose optimal probes based on available or hypothetical information. For example, probes can be chosen as a normalised sum of the candidate model's eigenvectors.
- Choice of t has a large effect on how well the model can train. By default, times are chosen proportional to the inverse of the current uncertainty in $\vec{\alpha}$ to maximise Fischer information, through the multi-particle guess heuristic described in Section 1.6.1 [7].

∗ Alternatively, evolution times may be chosen from a fixed set in order to force QHL to reproduce the dynamics within those times' scale. For instance, if a small amount of

experimental data is available offline, it is sensible to train all candidate models against the entire dataset.

MODEL TRAINING PRIOR

Specify the structure of the prior distribution, e.g. Fig. 1.1**(a)**.

– Set the initial mean and standard deviation of each parameter separately to define the prior multi-dimensional $\Pr(\vec{\alpha})$.

2.4.5 *Exploration strategy examples*

To solidifiy the concept of ESs, and how they affect the overall reach and runtime of a given ET, consider the following examples, where each strategy specifies how models are generated, as well as how trained models are consolidated within a branch. Recall that all of these strategies rely on QHL as the model training subroutine, so that the run time for training, is $t_{\text{QHL}} \sim N_e N_p t_{U(n)}$, where $t_{U(n)}$ is the time to compute the unitary evolution via the matrix exponential for an n-qubit model All models are trained using the default likelihood in Eq. (1.4). Assume the conditions

- all models considered are represented by 4-qubit models;
  - $t_{U(4)} \sim 10^{-3}$sec.
- each model undergoes a reasonable training regime;
  - $N_e = 1000, N_p = 3000$;
  - $\implies t_{\text{QHL}} = N_e \times N_p \times t_{U(4)} = 3000s \sim 1\text{h}$;
- Bayes factor calculations use
  - $N_e = 500, N_p = 3000$
  - $\implies t_{\text{BF}} \sim 2 \times 500 \times 3000 \times 10^{-3} \sim 1\text{h}$;
- there are 12 available terms
  - allowing any combination of terms, this admits a model space of size $2^{12} = 4096$
- access to 16 computer cores to parallelise calculations over
  - i.e. we can train 16 models or perform 16 BF comparisons in 1h.

Then, consider the following model generation/comparison strategies.

a. Predefined set of 16 models $\mathbb{H}$, with BF compareisons between every pair of models
   (i) Training takes 1h, and there are $\binom{16}{2} = 120$ comparisons spread across 16 processes, requiring 8h
   (ii) total time is 9h.

b. Generative procedure for model design, comparing every pair of models, running for 12 branches

(i) One branch takes 9h $\implies$ total time is $12 \times 9 = 108$h;

(ii) total number of models considered is $16 \times 12 = 192$.

c. Generative procedure for model design, where less model comparisons are needed (say one third of all model pairs are compared), running for 12 branches

(i) Training time is still 1h

(ii) One third of comparisons, i.e. 40 BF to compute, requires 3h

(iii) One branch takes $4h$ $\implies$ total time is 36h

(iv) total number of models considered is also 192.

These examples illustrate some of the design decisions involved in composing an ES, namely whether timing considerations are more important than thoroughly exploring the model space. They also show considerable time–savings in cases where it is acceptable to forego all model comparisons. The approch in (a) is clearly limited in its applicability, mainly in that there is a heavy requirement for prior knowledge, and it is only useful in cases where we either know $\hat{H}_0 \in \mathbb{H}$, or would be satisfied with approximating $\hat{H}_0$ as the closest available $\hat{H}_j \in \mathbb{H}$. On the opposite end of this spectrum, (c) is an excellent approach with respect to minimising prior knowledge required by the algorithm, although at the significant expense of testing a much larger number of candidate models. There is no optimal strategy: each use case demands particular considerations, and the amount of prior information available informs how wide the model search should reach.

2.5 GENERALITY

Several aspects of QMLA are deliberately vague in order to facilitate generality.

Model

Can mean any description of a quantum system which captures the interactions it is subject to.

– Here we exclusively consider Hamiltonian models, but Lindbladian models can also be considered as generators of quantum dynamics.

model training

Any subroutine which can train a given model, i.e. optimise a given parameterisation under the assumption that it represents the target system.

– Currently only QHL has been implemented, although for example tomography is valid in principle, with its own advantages and disadvantages. Overall QHL is found to fulfil the remit of model training with a balance of efficiency and rigour [1].

– QHL relies on the calculation of a characteristic likelihood function; this too is not restricted to the generic form of Eq. (1.4) and can be replaced by any form which represents the

likelihood that experimental conditions e result in measurement datum d. We will see examples of this in **??** where we trace out part of the system in order to represent open systems.

CONSOLIDATION

Can be as rigourous as desired by the user.

- Consolidation occurs at the branch level of each ET, but also in finding the tree champion, and ultimately the global champion.
- In practice, we use either BF or a related concept such as TLTL which are statistically significative. However, in **??** we will consider a number of alternative schemes for discerning the strongest models.

2.5.1 *Agency*

While the concept of *agency* is contentious [35], we can view our overall protocol as a multi-agent system [36], or even an agent based evolutionary algorithm [37], because any given ES satisfies the definition, *the population of individuals can be considered as a population of agents*, where we mean the population of models present on a given ET. More precisely, we can view individual models as *learning agents* according to the criteria of [38], i.e. that a learning agent has

- a *problem generator*: designs actions in an attempt to learn about the system – this is precisely the role of the EDH;
- a *performance element*: implements the designed actions and measures the outcome – the measurement of a datum following the experiment designed by the EDH;
- a *critic*: the likleihood function informs whether the designed action (experiment) was successful;
- a *learning element*: the updates to the weights and overall parameter distribution improve the model's performance over time.

We depict this analogy in Fig. 2.3. Overall, the QMLA model search can be regarded as an unsupervised ML algorithm, since the true model is unknown. However, the flexibility of the ES paradigm permits any model design mechanism: in general we can say the model search – and the model training subroutine – is *guided* by the environment, i.e. it learns whether subspaces of candidates are effective from interaction with the system. We are therefore justified in labelling the entire procedure as the *quantum model learning agent*.

2.6 ALGORITHMS

We conclude this chapter by listing the algorithms used most frequently, in order to clarify each of their roles, and how they interact. Algorithm 1 shows the overall QMLA algorithm, which is

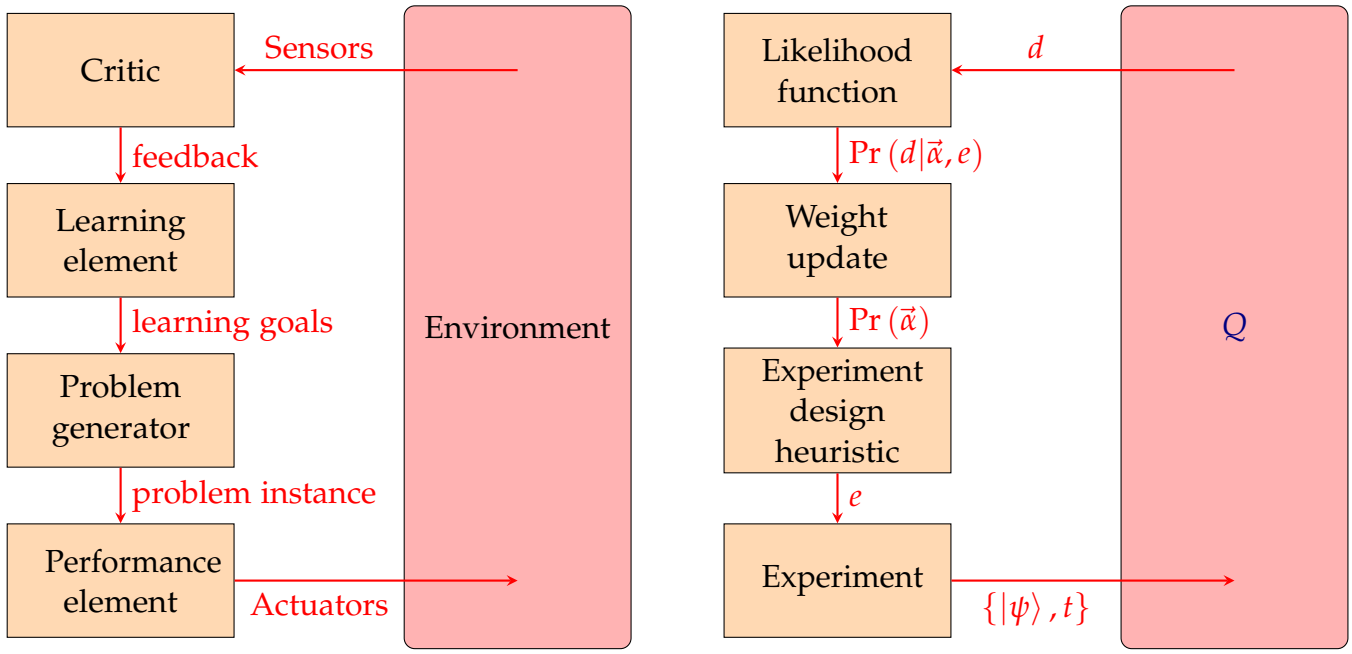

Figure 2.3: Learning agents. **Left,** definition of a learning agent, where an *environment* is affected by *actuators* which realise a *problem instance*, designed by a *problem generator*, through some *performance element*. The result of the agent's action is detected by *sensors*, which the *critic* interprets with respect to the agent's *learning goals*, by providing *feedback* to the *learning element*. **Right,** mapping of the concept of a learning agent on to an individual model. A target quantum system, Q, is queried by performing some experiment e, designed by an experiment design heuristic, and implemented by evolving a probe state $|\psi\rangle$ for time t. The systems is measured, and the datum d is sent to the likelihood function, which sends the likelihood $\Pr(d|\vec{\alpha}, t)$ to the weight update (and the parameter distribution update), before designing another experiment.

simplified greatly to a loop over the model search of each ES. The model search itself is listed in Algorithm 2, which contains calls to subroutines for model learning (QHL, Algorithm 5), branch consolidation (which can be based upon BF, Algorithm 6) and centers on the generation of new models, an example of which – based on a *greedy search* prerogative – is given in Algorithm 4.

Algorithm 1: Quantum Model Learning Agent

Input: Q // some physically measurable or simulateable quantum system
Input: $\mathbb{S}$ // set of exploration strategies

Output: $\hat{H}'$ // champion model

$\mathbb{H}_c \leftarrow \{\}$
for $S \in \mathbb{S}$ **do**
  $\hat{H}'_S \leftarrow$ model_search(Q, S) // run model search for this ES, e.g. Algorithm 2
  $\mathbb{H}_c \leftarrow \mathbb{H}_c \cup \{\hat{H}'_S\}$ // add ES champion to collection
end
$\hat{H}' \leftarrow$ final_champion($\mathbb{H}_c$)
return $\hat{H}'$

Algorithm 2: Exploration strategy subroutine: model_search

Input: Q // some physically measurable or simulateable quantum system
Input: S // exploration strategy: collection of rules/subroutines

Output: $\hat{H}'_S$ // exploration strategy's nominated champion model

$\nu \leftarrow \{\}$
$\mathbb{H}_c \leftarrow \{\}$
while !S.terminate() **do**
  $\mu \leftarrow$ S.generate_models(ν) // spawn new models, e.g. Algorithm 4

  for $\hat{H}_i \in \mu$ **do**
    $\hat{H}'_i \leftarrow$ S.train($\hat{H}_i$) // train candidate model, e.g. Algorithm 5
  end
  $\nu \leftarrow$ S.consolidate(μ) // consolidate set of models, e.g. pairwise via Algorithm 3
  $\hat{H}^{\mu}_c \leftarrow$ S.branch_champion(ν) // use ν to select a branch champion
  $\mathbb{H}_c \leftarrow \mathbb{H}_c \cup \{\hat{H}^{\mu}_c\}$ // add branch champion to collection
end
$\hat{H}'_S \leftarrow$ S.nominate_champion($\mathbb{H}_c$)
return $\hat{H}'_S$

Algorithm 3: Exploration strategy subroutine: consolidate (points per Bayes factor win)

Input: μ // information about models considered to date
Input: b // threshold for sufficient evidence that one model is favoured
Input: BF() // function to compute the BF between $\hat{H}_j$ and $\hat{H}_k$, Algorithm 6
Output: $\hat{H}_{k'}$ // favoured model within μ

$\mathbb{H} \leftarrow$ extract_models(μ)
for $\hat{H}_j \in \mathbb{H}$ **do**
  $s_j = 0$ // initialise score for each model
end
for $\hat{H}_j, \hat{H}_k \in \mathbb{H}$ // pairwise Bayes factor between all models in the set
do
  $B \leftarrow \mathrm{BF}(\hat{H}_j, \hat{H}_k)$
  if $B > b$ // increase score of winning model
  then
    $s_j \leftarrow s_j + 1$
  else if $B < 1/b$ **then**
    $s_k \leftarrow s_k + 1$
end
$k' \leftarrow \max_k\{s_k\}$ // find which model has most points
return $\hat{H}_{k'}$

Algorithm 4: Exploration strategy subroutine: generate_models (greedy spawn)

Input: ν // information about models considered to date
Input: $\mathcal{T}$ // set of terms to search
Output: $\mathbb{H}$ // set of candidate models

$\hat{H}_C^{\mu} \leftarrow$ top_model(ν) // find previous branch champion
$\{\hat{t}_c^{\mu}\} \leftarrow$ get_terms($\hat{H}_C^{\mu}$) // extract terms of branch champion
$\mathcal{T}' \leftarrow \mathcal{T} \setminus \{\hat{t}_c^{\mu}\}$ // remove terms of branch champion

$\mathbb{H} \leftarrow \{\}$
for $\hat{t} \in \mathcal{T}'$ **do**
  $\hat{H}_i \leftarrow \hat{H}_C^{\mu} + \hat{t}$ // add one term to the branch champion
  $\mathbb{H} \leftarrow \mathbb{H} \cup \{\hat{H}_i\}$
end
return $\mathbb{H}$

Algorithm 5: Quantum Hamiltonian Learning

Input: Q // some physically measurable or simulatable quantum system, described by $\hat{H}_0$
Input: $\hat{H}_i$ // Hamiltonian model attempting to reproduce data from $\hat{H}_0$
Input: $\Pr(\vec{\alpha})$ // probability distribution for $\vec{\alpha} = \vec{\alpha}_0$
Input: N_E // number of experiments to iterate learning procedure for
Input: N_P // number of particles to draw from $\Pr(\vec{\alpha})$
Input: $\Lambda(\Pr(\vec{\alpha}))$ // experiment design heuristic
Input: $\mathrm{RS}(\Pr(\vec{\alpha}))$ // resampling algorithm for redrawing particles
Output: $\vec{\alpha}'$ // estimate of Hamiltonian parameters

$\mathcal{P} \leftarrow \mathrm{sample}(\Pr(\vec{\alpha}), N_p)$ // sample particles from prior
for $p \in \mathcal{P}$ **do**
  $w_p \leftarrow 1/N_p$ // set weights for each particle
end

for $e \in \{1 \rightarrow N_e\}$ **do**
  $t, |\psi\rangle \leftarrow \Lambda(\Pr(\vec{\alpha}))$ // design an experiment
  $d \leftarrow$ prepare Q in $|\psi\rangle$, evolve and measure after t // datum
  for $p \in \mathcal{P}$ **do**
    $\vec{\alpha}_p \leftarrow$ parameter vector corresponding to p
    $\Pr(d|\vec{\alpha}_p; t) \leftarrow |\langle d| e^{-iH(\vec{\alpha}_p)t} |\psi\rangle|^2$ // likelihood
    $w_p \leftarrow w_p \times \Pr(d|\vec{\alpha}_p; t)$ // weight update
  end

  if $1/\sum_p w_p^2 < N_p/2$ // check whether to resample (are weights too small?)
  then
    $\mathrm{RS}(\Pr(\vec{\alpha})) \leftarrow \mathcal{P}$ // redraw particles via resampling algorithm
    for $p \in \mathcal{P}$ **do**
      $w_p \leftarrow 1/N_p$ // set weights for each particle
    end
  end
end

$\vec{\alpha}'_i \leftarrow \mathrm{mean}(\Pr(\vec{\alpha})) \leftarrow \vec{\alpha}'$
return $\vec{\alpha}'$

Algorithm 6: Bayes Factor calculation

Input: Q // some physically measurable or simulateable quantum system.
Input: $\hat{H}_j, \hat{H}_k$ // Hamiltonian models to compare
Input: $\Pr_j(\vec{\alpha}), \Pr_k(\vec{\alpha})$ // posterior distribution following training for $\hat{H}_j, \hat{H}_k$
Input: N'_p // number of particles to draw from posteriors for evaluation
Input: $\mathcal{E}_j, \mathcal{E}_k$ // experiments on which $\hat{H}_j$ and $\hat{H}_k$ were trained during QHL
Output: B_{jk} // Bayes factor between two candidate Hamiltonians

$\mathcal{E} = \{\mathcal{E}_j \cup \mathcal{E}_k\}$ // common experiments for fair comparison

for $\hat{H}_i \in \{\hat{H}_j, \hat{H}_k\}$ **do**
  $\mathcal{L}_i = 0$ // total log total likelihood for $\hat{H}_i$
  for $e \in \mathcal{E}$ **do**
    $e \leftarrow t, |\psi\rangle$ // assign evolution time and probe from experiment control set
    $d \leftarrow$ Prepare Q in $|\psi\rangle$, evolve and measure after t // datum

    $\mathcal{P} \leftarrow \text{sample}(\Pr_i(\vec{\alpha}), N'_p)$ // sample from $\hat{H}_i$'s posterior
    $l_e \leftarrow 0$ // total likelihood for $\hat{H}_i$ on e

    for $\vec{\alpha}_p \in \mathcal{P}$ **do**
      $\Pr(d|\hat{H}_i, t) \leftarrow \left|\langle d| e^{-i\hat{H}_i(\vec{\alpha}_p)t} |\psi\rangle\right|^2$ // likelihood for particle $\vec{\alpha}_p$
      $l_e \leftarrow l_e + \Pr(d|\hat{H}_i, t)$ // add l_e to total likelihood
    end
    $\mathcal{L}_i \leftarrow \mathcal{L}_i + \ln(l_e)$ // add $\ln(l_e)$ to total log total likelihood
  end
end
$B_{jk} \leftarrow \exp\left(\mathcal{L}_j - \mathcal{L}_k\right)$ // Bayes factor between models

return B_{jk}

Part III

THEORETICAL STUDY

Part IV

EXPERIMENTAL STUDIES

Part V

CONCLUSION

3

OUTLOOK FOR MODEL LEARNING METHODOLOGIES

Optimal control techniques are a crucial component in improving quantum technologies, such that imperfect near-term devices may be leveraged to achieve some meaningful quantum advantages. The developments presented in this thesis contribute to the growing interest in automatic characterisation and verification of quantum systems and devices. Namely, the introduction of the Quantum Model Learning Agent (QMLA) represents an important advancement, whereby quantum systems can be completely characterised starting with little prior knowledge. The majority of this thesis was dedicated to the rigourous testing of QMLA, gradually moving from ideal scenarios in simulation to genuine experimental quantum systems.

We described the implementation of QMLA as an open source software platform in Part II, detailing numerous tunable aspects of the protocol, and their impact on training candidate models in Chapter 1. QMLA facilitates customisation of its core elements and subroutines, such that it is applicable to a wide range of target quantum systems, as described in 2–**??**. This malleability enables users to easily adapt the framework to their own needs, and formed the basis for the cases studied in the remainder of the thesis: we tested QMLA by devising a series of exploration strategies, each corresponding to a different target quantum system.

In Part III we considered ideal theoretical quantum systems in simulation. Initial tests in **??** showed that QMLA could distinguish between different physical scenarios and internal configurations. In **??**, we explored much larger model spaces by incorporating a genetic algorithm (GA) into QMLA's model design; the GA showed promise for characterising complex quantum systems by successfully identifying the target model. The performance of the GA, however, came at the expense of relying on a restrictive subroutine – used for training individual candidate models – drastically reducing its applicability to realistic systems. However, the restriction is permitted in the scope of characterising *controlled* quantum systems, for example new, untrusted quantum simulators.

We concluded the thesis by considering realistic quantum systems in Part IV. Experimental data from an electron spin in a nitrogen-vacancy centre (NVC) was treated in **??**; this too relied upon tailoring QMLA's procedure with respect to the system under study. A theoretically justified Hamiltonian is proposed by QMLA to describe the decoherence of the electron spin, yielding a highly predictive model in agreement with the system's measured dynamics, albeit exploring a small model space. To overcome concerns that the model search was artifically constrained in the context of realistic systems, **??** exercised QMLA in a vast model space, spanning terms which represent plausible interactions for the same type of system. Here, again, QMLA achieved high success rates, but with caveats on the subroutines assumed for model training, and resorting to simulated data.

In summary, this thesis has provided extensive tests of the QMLA algorithm, but each may be undermined by its individual constraints. In outlook, near-term developments of model learning methodologies in the context of quantum systems must address these shortcomings, for instance by unifying the strategies described in this thesis. Further, we anticipate immediate application in the study of open quantum systems, by replacing the Hamiltoinian formalism examined here with a Lindbladian representation, permitted within the QMLA apparatus. Through the advancements presented herein, we hope to have provided a solid foundation upon which these constraints may be relaxed, ultimately with a view to providing an automated platform for the complete characterisation of quantum systems. We envision QMLA as a straighforward but powerful utility for quantum engineers in the design of near term quantum devices, expecting continued development of the framework alongside the burgeoning open-source quantum software eco-system.

BIBLIOGRAPHY

[1] 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. Accepted: Nature Physics.

[2] Brian Flynn, Antonio A. Gentile, Raffaele Santagati, Nathan Wiebe, and Anthony Laing. Quantum model learning agent: quantum systems' characterisation through machine learning. In preparation, 2021.

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

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

[5] Christopher E Granade, C Ferrie, N Wiebe, and D G Cory. Robust online Hamiltonian learning. *New Journal of Physics*, 14(10):103013, October 2012.

[6] Nathan Wiebe, Christopher Granade, Christopher Ferrie, and David Cory. Quantum hamiltonian learning using imperfect quantum resources. *Physical Review A*, 89(4):042314, 2014.

[7] N Wiebe, C Granade, C Ferrie, and D G Cory. Hamiltonian Learning and Certification Using Quantum Resources. *Physical Review Letters*, 112(19):190501–5, May 2014.

[8] Jianwei Wang, Stefano Paesani, Raffaele Santagati, Sebastian Knauer, Antonio A Gentile, Nathan Wiebe, Maurangelo Petruzzella, Jeremy L O'Brien, John G Rarity, Anthony Laing, et al. Experimental quantum hamiltonian learning. *Nature Physics*, 13(6):551–555, 2017.

[9] Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice*, pages 197–223. Springer, 2001.

[10] Seth Lloyd. Universal quantum simulators. *Science*, pages 1073–1078, 1996.

[11] Andrew M Childs, Dmitri Maslov, Yunseong Nam, Neil J Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

[12] Dominic W Berry, Andrew M Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809. IEEE, 2015.

[13] Alessandro Rudi, Leonard Wossnig, Carlo Ciliberto, Andrea Rocchetto, Massimiliano Pontil, and Simone Severini. Approximating hamiltonian dynamics with the nyström method. *Quantum*, 4:234, 2020.

[14] Christopher Granade, Christopher Ferrie, Steven Casagrande, Ian Hincks, Michal Kononenko, Thomas Alexander, and Yuval Sanders. QInfer: Library for statistical inference in quantum information, 2016.

[15] Arseni Goussev, Rodolfo A Jalabert, Horacio M Pastawski, and Diego Wisniacki. Loschmidt echo. *arXiv preprint arXiv:1206.6348*, 2012.

[16] Nathan Wiebe, Christopher Granade, and David G Cory. Quantum bootstrapping via compressed quantum hamiltonian learning. *New Journal of Physics*, 17(2):022005, 2015.

[17] Bas Hensen, Hannes Bernien, Anaïs E Dréau, Andreas Reiserer, Norbert Kalb, Machiel S Blok, Just Ruitenberg, Raymond FL Vermeulen, Raymond N Schouten, Carlos Abellán, et al. Loophole-free bell inequality violation using electron spins separated by 1.3 kilometres. *Nature*, 526(7575):682–686, 2015.

[18] Alexandr Sergeevich, Anushya Chandran, Joshua Combes, Stephen D Bartlett, and Howard M Wiseman. Characterization of a qubit hamiltonian using adaptive measurements in a fixed basis. *Physical Review A*, 84(5):052315, 2011.

[19] Christopher Ferrie, Christopher E Granade, and David G Cory. How to best sample a periodic probability distribution, or on the accuracy of hamiltonian finding strategies. *Quantum Information Processing*, 12(1):611–623, 2013.

[20] Christopher E Granade. Characterization, verification and control for large quantum systems. page 92, 2015.

[21] Christopher Ferrie. High posterior density ellipsoids of quantum states. *New Journal of Physics*, 16(2):023006, 2014.

[22] Michael J Todd and E Alper Yıldırım. On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

[23] Ian Hincks, Thomas Alexander, Michal Kononenko, Benjamin Soloway, and David G Cory. Hamiltonian learning with online bayesian experiment design in practice. *arXiv preprint arXiv:1806.02427*, 2018.

[24] Lukas J Fiderer, Jonas Schuff, and Daniel Braun. Neural-network heuristics for adaptive bayesian quantum estimation. *arXiv preprint arXiv:2003.02183*, 2020.

[25] Sheng-Tao Wang, Dong-Ling Deng, and Lu-Ming Duan. Hamiltonian tomography for quantum many-body systems with arbitrary couplings. *New Journal of Physics*, 17(9):093017, 2015.

[26] Stefan Krastanov, Sisi Zhou, Steven T Flammia, and Liang Jiang. Stochastic estimation of dynamical variables. *Quantum Science and Technology*, 4(3):035003, 2019.

[27] Emmanuel Flurin, Leigh S Martin, Shay Hacohen-Gourgy, and Irfan Siddiqi. Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations. *Physical Review X*, 10(1):011006, 2020.

[28] Murphy Yuezhen Niu, Vadim Smelyanskyi, Paul Klimov, Sergio Boixo, Rami Barends, Julian Kelly, Yu Chen, Kunal Arya, Brian Burkett, Dave Bacon, et al. Learning non-markovian quantum noise from moire-enhanced swap spectroscopy with deep evolutionary algorithm. *arXiv preprint arXiv:1912.04368*, 2019.

[29] Eliska Greplova, Christian Kraglund Andersen, and Klaus Mølmer. Quantum parameter estimation with a neural network. *arXiv preprint arXiv:1711.05238*, 2017.

[30] Andrey Y Lokhov, Marc Vuffray, Sidhant Misra, and Michael Chertkov. Optimal structure and parameter learning of ising models. *Science advances*, 4(3):e1700791, 2018.

[31] Giovanni Acampora, Vittorio Cataudella, Pratibha R Hegde, Procolo Lucignano, Gianluca Passarelli, and Autilia Vitiello. An evolutionary strategy for finding effective quantum 2-body hamiltonians of p-body interacting systems. *Quantum Machine Intelligence*, 1(3):113–122, 2019.

[32] Daniel Burgarth and Ashok Ajoy. Evolution-free hamiltonian parameter estimation through zeeman markers. *Physical Review Letters*, 119(3):030402, 2017.

[33] Agnes Valenti, Guliuxin Jin, Julian Léonard, Sebastian D Huber, and Eliska Greplova. Scalable hamiltonian learning for large-scale out-of-equilibrium quantum dynamics. *arXiv preprint arXiv:2103.01240*, 2021.

[34] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.

[35] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer, 1996.

[36] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[37] Ruhul A Sarker and Tapabrata Ray. Agent based evolutionary approach: An introduction. In *Agent-Based Evolutionary Search*, pages 1–11. Springer, 2010.

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