



EPSRC Centre for Doctoral Training  
**Quantum Engineering**



University of  
**BRISTOL**

DOCTORATE OF PHILOSOPHY

---

# Schrödinger's Catwalk

---

BRIAN FLYNN

UNIVERSITY OF BRISTOL

January, 2021

# CONTENTS

---

## Appendix

A	FIGURE REPRODUCTION	2
B	FUNDAMENTALS	6
B.1	Linear algebra . . . . .	6
B.2	Postulates of quantum mechanics . . . . .	7
B.3	States . . . . .	8
B.3.1	Mulitpartite systems . . . . .	9
B.3.2	Registers . . . . .	10
B.4	Entanglement . . . . .	11
B.5	Unitary Transformations . . . . .	11
B.6	Dirac Notation . . . . .	12
C	EXAMPLE EXPLORATION STRATEGY RUN	17
C.1	Custom exploration strategy . . . . .	20

## LIST OF TABLES

---

Table A.1	Figure implementation details . . . . .	4
Table A.2	Figure implementation details continued . . . . .	5
Table B.1	Linear algebra defintions . . . . .	6

## LIST OF FIGURES

---

Figure C.1	Terminal running redis-server . . . . .	18
Figure C.2	Quantum Model Learning Agent (QMLA) plot: learning_summary . . . .	24
Figure C.3	QMLA plot: dynamics . . . . .	25

## LISTINGS

---

A.1	"QMLA Launch script" . . . . .	2
C.1	QMLA codebase setup . . . . .	17
C.2	Launch redis database . . . . .	18
C.3	local_launch script . . . . .	18
C.4	Launch QMLA . . . . .	19
C.5	QMLA results directory . . . . .	20
C.6	QMLA codebase setup . . . . .	20
C.7	QMLA codebase setup . . . . .	20
C.8	QMLA codebase setup . . . . .	21
C.9	local_launch configuration for QHL . . . . .	22

## ACRONYMS

---

$^{13}\text{C}$	carbon-13. 118, 122
$^{14}\text{N}$	nitrogen-14. 118, 120–122
AI	artificial intelligence. 15
AIC	Akaike information criterion. 103, 104
AICC	Akaike information criterion corrected. 103
BF	Bayes factor. 49–51, 57, 59, 72, 76, 80, 82, 87–89, 96, 98, 99, 104–109, 112, 113, 126, 127, 132
BFEER	Bayes factor enhanced Elo ratings. 98, 99, 111, 112
BIC	Bayesian information criterion. 104
CLE	classical likelihood estimation. 37, 123
CPU	central processing unit. 15, 22
EDH	experiment design heuristic. 42–45, 47, 50, 59, 69, 80, 81, 101, 128, 129
ES	exploration strategy. 51–55, 57–59, 64, 67, 69, 73, 74, 76, 87, 90, 92, 97, 118, 121, 123, 125–129, 132, 138
ET	exploration tree. 52, 53, 55, 57–59, 71, 92, 126, 127, 132
FH	Fermi-Hubbard. 83
FN	false negatives. 95
FP	false positives. 19, 95
GA	genetic algorithm. v–vii, 22, 23, 27, 29, 30, 58, 92, 93, 98, 100, 101, 106, 108, 111, 114, 115, 138, 140, 141
GES	genetic exploration strategy. 92, 96, 111, 112, 116
GPU	graphics processing unit. 15, 22
HPD	high particle density. 42
IQLE	interactive quantum likelihood estimation. 37, 38, 111, 112

LTL	log total likelihood. 40
ML	machine learning. 15–17, 19, 21, 22, 50, 51, 95
MS	model search. 51–53, 55, 59, 67, 69
MVEE	minimum volume enclosing ellipsoid. 42
NV	nitrogen-vacancy. 33, 118
NVC	nitrogen-vacancy centre. v, vii, 38, 118–122, 124–128, 132, 133, 136, 138–141
OF	objective function. vi, 17, 23, 26, 30, 92, 93, 96, 99, 101, 103, 106, 109–111
PGH	particle guess heuristic. 43, 44, 69, 128, 129
PL	photoluminescence. 120, 125, 130
QC	quantum computer. 7, 13–15, 22
QHL	quantum Hamiltonian learning. vi, 32–38, 40, 42, 44–46, 49–51, 55–59, 64, 67, 72, 80, 81, 92, 93, 111, 112, 125, 128
QL	quadratic loss. 41
QLE	quantum likelihood estimation. 37, 56, 120
QM	quantum mechanics. 4, 6, 8, 9, 145, 155
QML	quantum machine learning. 15, 21, 22
QMLA	Quantum Model Learning Agent. v–vii, 32, 37, 48, 49, 51–55, 58, 59, 64, 67–74, 76, 77, 85–90, 92–95, 98, 101, 104–106, 108, 109, 111, 112, 114–116, 118, 121–123, 125–141
SMC	sequential monte carlo. 35–37, 39, 42, 43, 47, 55
SVM	support vector machine. 16
TLTL	total log total likelihood. 40, 49–51, 59, 101, 103, 109
TN	true negatives. 95
TP	true positives. 19, 95

## GLOSSARY

---

Jordan Wigner transformation (JWT)	Jordan Wigner transformation . 85, 86, 89
Loschmidt echo (LE)	Quantum chaotic effect described. . 38
chromosome	A single candidate in the space of valid solutions to the posed problem in a genetic algorithm. . 23, 25
expectation value	Average outcome expected by measuring an observable of a quantum system many times, ??.. i, 12
gene	Individual element within a chromosome. . 23, 25
hyperparameter	Variable within an algorithm that determines how the algorithm itself proceeds.. 35
instance	a single implementation of the QMLA algorithm. vi, 72, 112, 114, 115, 132
likelihood	Value that represents how likely a hypothesis is.. 12, 34, 37, 39, 42, 55, 57, 58, 60, 105, 120, 125
model	The mathematical description of some quantum system. 48
model space	Abstract space containing all descriptions (within defined constraints such as dimension) of the system as models. 53, 138
probe	Input probe state, $ \psi\rangle$ , which the target system is initialised to, before unitary evolution. plural. 37, 39, 42–46, 79
results directory	Directory to which the data and analysis for a given run of QMLA are stored. . 73



run	collection of QMLA instances. iii, vi, vii, 72, 73, 89, 90, 112, 115, 116, 129, 132, 139, 140
spawn	Process by which new models are generated by combining previously considered models.. 53
success rate	. 73
term	Individual constituent of a model, e.g. a single operator within a sum of operators, which in total describe a Hamiltonian. . 48
volume	Volume of a parameter distribution's credible region.. 42, 80, 81, 88, 129
win rate	. 73

## APPENDIX

## FIGURE REPRODUCTION

---

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

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

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

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

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

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

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

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

exploration_strategy='AnalyticalLikelihood'
```

Listing A.1: "QMLA Launch script"

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

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

Figure	Exploration Strategy	$N_E$	$N_P$	Data
??	DemoBayesFactorsByFscore	500	2500	Dec_09/12_29
	DemoFractionalResourcesBayesFactorsByFscore	500	2500	Dec_09/12_31
	DemoBayesFactorsByFscore	1000	5000	Dec_09/12_33
	DemoBayesFactorsByFscoreEloGraphs	500	2500	Dec_09/12_32
??	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
??	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
	HeisenbergGeneticXYZ	500	2500	Dec_10/14_40
??	HeisenbergGeneticXYZ	500	2500	Dec_10/16_12
	HeisenbergGeneticXYZ	500	2500	Dec_10/16_12
??	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
??	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
??	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
??	SimulatedExperimentNVCentre	1000	3000	2019/Oct_02/18_16
	NVCentreExperimentalData	1000	3000	2019/Oct_02/18_01
??	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
??	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00
	NVCentreGenticAlgorithmPrelearnedParameters	2	5	Sep_09/12_00

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

## FUNDAMENTALS

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

## B.1 LINEAR ALGEBRA

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

- Notation

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

Table B.1: Linear algebra definitions.

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

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

$$M^\dagger = \begin{pmatrix} M_{0,0} & M_{0,1} \\ M_{1,0} & M_{1,1} \end{pmatrix}^\dagger = \begin{pmatrix} M_{0,0}^* & M_{1,0}^* \\ M_{0,1}^* & M_{1,1}^* \end{pmatrix}^T = \begin{pmatrix} M_{0,0}^* & M_{1,0}^* \\ M_{0,1}^* & M_{1,1}^* \end{pmatrix} \quad (\text{B.1})$$

<sup>1</sup> Much of this description is reproduced from my undergraduate thesis [1].

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

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

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

## B.2 POSTULATES OF QUANTUM MECHANICS

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

- 1 Every moving particle in a conservative force field has an associated wave-function,  $|\psi\rangle$ . From this wave-function, it is possible to determine all physical information about the system.
- 2 All particles have physical properties called observables (denoted  $q$ ). In order to determine a value,  $q$ , for a particular observable, there is an associated *operator*  $\hat{Q}$ , which, when acting on the particles wavefunction, yields the value times the wavefunction. The observable  $q$  is then the eigenvalue of the operator  $\hat{Q}$ .

$$\hat{Q} |\psi\rangle = q |\psi\rangle \quad (\text{B.3})$$

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

$$\hat{Q}^\dagger = \hat{Q} \quad (\text{B.4})$$

- 4 The set of eigenfunctions for any operator  $\hat{Q}$  forms a complete set of linearly independent functions.
- 5 For a system with wavefunction  $|\psi\rangle$ , the expectation value of an observable  $q$  with respect to an operator  $\hat{Q}$  is denoted by  $\langle q \rangle$  and is given by

$$\langle q \rangle = \langle \psi | \hat{Q} | \psi \rangle \quad (\text{B.5})$$



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

$$i\hbar \frac{\partial \psi}{\partial t} = \hat{H}\psi, \quad (\text{B.6})$$

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

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

### B.3 STATES

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

$$|\psi\rangle = \sum_x a_x |x\rangle \quad (\text{B.7a})$$

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

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

For example, photons can be polarised horizontally ( $\leftrightarrow$ ) or vertically ( $\updownarrow$ ); take those two conditions as observable states to define the eigenstates of a two-level system, so we can designate the photon as a qubit. Then we can map the two states to a 2-dimensional,  $x$ - $y$  plane:

a general vector on such a plane can be represented by a vector with coordinates  $\begin{pmatrix} x \\ y \end{pmatrix}$ . These polarisations can then be thought of as standard basis vectors in linear algebra. Denote  $\leftrightarrow$  as the eigenstate  $|0\rangle$  and  $\updownarrow$  as  $|1\rangle$

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

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

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

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

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

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

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

These particular superpositions are of significance:

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

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

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

### B.3.1 Mulitpartite systems

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

$$|\psi\rangle = |\psi_A\rangle |\psi_B\rangle = |\psi_A\psi_B\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \quad (\text{B.12})$$

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

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

<sup>2</sup> We will later discuss entangled states, which can not be described thus.

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

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

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

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

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

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

### B.3.2 Registers

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

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

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

#### B.4 ENTANGLEMENT

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

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

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

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

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

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

#### B.5 UNITARY TRANSFORMATIONS

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

$$U(a_1|\psi_1\rangle + a_2|\psi_2\rangle + \dots a_k|\psi_k\rangle) = a_1(U|\psi_1\rangle) + a_2(U|\psi_2\rangle) + \dots + a_k(U|\psi_k\rangle). \quad (\text{B.17})$$

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

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

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

$$UU^\dagger = I \Rightarrow U^\dagger = U^{-1} \quad (\text{B.18})$$

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

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

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

## B.6 DIRAC NOTATION

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

$$\begin{aligned} |x\rangle &= |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ |y\rangle &= |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \quad (\text{B.20})$$

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

$$\begin{aligned} \langle\psi|^\dagger &= |\psi\rangle \\ |\psi\rangle^\dagger &= \langle\psi| \end{aligned} \quad (\text{B.21})$$

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} \Rightarrow \langle\psi| = (\psi_1^* \quad \psi_2^* \quad \dots \quad \psi_n^*) \quad (\text{B.22})$$

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

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

**Example B.6.1.**

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

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

**Example B.6.2.**

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

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

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

$$|0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad (\text{B.26b})$$

$$|1\rangle\langle 0| = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad (\text{B.26c})$$

$$|1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad (\text{B.26d})$$

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

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

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

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

Which is equivalent to the outer product representation

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

For clarity, here we will prove this operation

**Example B.6.3.**

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

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

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

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

To demonstrate how Dirac notation simplifies this:

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

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

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

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

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

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

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

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

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

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



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

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

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

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

We can express the above as

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

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

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

## EXAMPLE EXPLORATION STRATEGY RUN

---

A complete example of how to run the Quantum Model Learning Agent (QMLA) framework, including how to implement a custom exploration strategy (ES), and generate/interpret analysis, is given.

First, *fork* the QMLA codebase from [2]<sup>1</sup>. Now, we must download the code base and ensure it runs properly; these instructions are implemented via the command line<sup>2</sup>.

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

# make directory for QMLA
cd
mkdir qmla_test
cd qmla_test

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

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

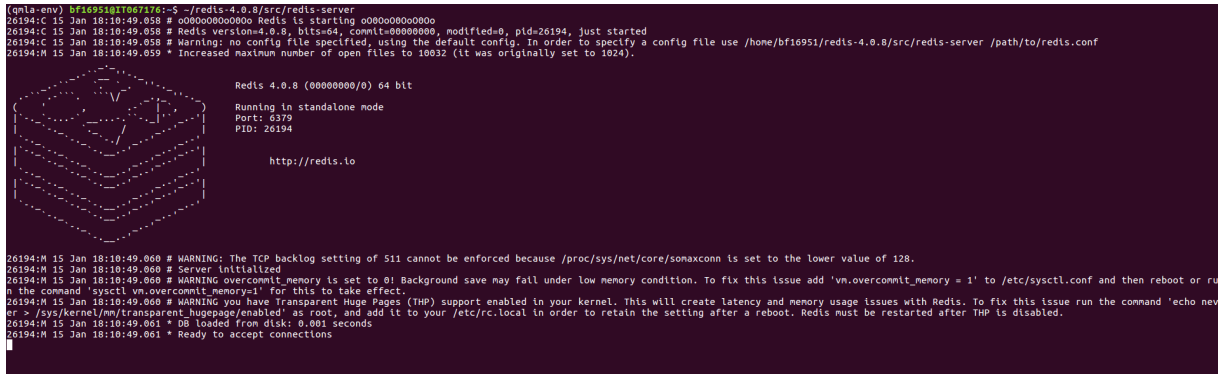
# Install dependencies
cd QMLA
pip install -r requirements.txt # note there may be a problem
    with matplotlib version; just pip install it directly
```

Listing C.1: QMLA codebase setup

---

<sup>1</sup> This will require a Github account.

<sup>2</sup> Note: these instructions are tested for Linux and presumed to work on Mac, but untested on Windows. It is likely some of the underlying software (redis servers) can not be installed on Windows.



```

(qmla-env) bf16951g17067176:~$ ~/redis-4.0.8/src/redis-server
26194:C 15 Jan 18:10:49.058 # 000000000000 Redis is starting 000000000000
26194:C 15 Jan 18:10:49.058 # Redis version=4.0.8, bits=64, commit=00000000, modified=0, pid=26194, just started
26194:C 15 Jan 18:10:49.058 # Warning: no config file specified, using the default config. In order to specify a config file use /home/bf16951/redis-4.0.8/src/redis-server /path/to/redis.conf
26194:M 15 Jan 18:10:49.059 * Increased maximum number of open files to 10032 (it was originally set to 1024).

Redis 4.0.8 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 26194

http://redis.io

26194:M 15 Jan 18:10:49.060 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
26194:M 15 Jan 18:10:49.060 # Server initialized
26194:M 15 Jan 18:10:49.060 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
26194:M 15 Jan 18:10:49.060 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
26194:M 15 Jan 18:10:49.061 * DB loaded from disk: 0.001 seconds
26194:M 15 Jan 18:10:49.061 * Ready to accept connections

```

Figure C.1: Terminal running redis-server.

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

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

Listing C.2: Launch redis database

which should give something like Appendix C.

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

```

#!/bin/bash

#####
# QMLA run configuration
#####
num_instances=1
run_qhl=1 # perform QHL on known (true) model
run_qhl_multi_model=0 # perform QHL for defined list of models.
exp=5 # number of experiments
prt=20 # number of particles

#####
# QMLA settings - user
#####

```

```

plot_level=5
debug_mode=0

#####
# QMLA settings - default
#####
do_further_qhl=0 # QHL refinement to best performing models
q_id=0 # instance ID can start from other ID if desired
use_rq=0
further_qhl_factor=1
further_qhl_num_runs=$num_instances
plots=0
number_best_models_further_qhl=5

#####
# Choose an exploration strategy
# This will determine how QMLA proceeds.
#####

exploration_strategy="ExampleBasic"

```

Listing C.3: local\_launch script

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

```

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

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

```

Listing C.4: Launch QMLA

There may be numerous warnings, but they should not affect whether QMLA has succeeded; QMLA will raise any significant error. Assuming the run has completed successfully, QMLA stores the run's results in a subdirectory named by the date and time it was started. Plots/analy-

sis are generated at the instance and model level; each results directory has an `analyse.sh` script to generate plots at the run level. For examples, if the run was initialised on January 1<sup>st</sup> at 01:23 navigate to that directory and analyse it by

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

Listing C.5: QMLA results directory

For now it is sufficient to notice that the code has run successfully and the analysis has generated several new files and directories; in the following sections we will describe the outputs.

## C.1 CUSTOM EXPLORATION STRATEGY

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

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

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

Listing C.6: QMLA codebase setup

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

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

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

## Listing C.7: QMLA codebase setup

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

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

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

$$\text{pauliSet\_1J2\_zJz\_d4} + \text{pauliSet\_2J3\_zJz\_d4} + \text{pauliSet\_3J4\_zJz\_d4}.$$

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

```
class ExampleBasic(
    exploration_strategy.ExplorationStrategy
):

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

```

self.initial_models = None
self.max_spawn_depth = 1
self.true_model_terms_params = {
    'pauliSet_1J2-zJz-d3' : 2.5,
    'pauliSet_2J3-zJz-d3' : 7.5,
    'pauliSet_4J5-zJz-d3' : 3.5,
}
self.min_param = 0
self.max_param = 10

def generate_models(self, **kwargs):

    self.log_print(["Generating models; spawn step {}".format
        (self.spawn_step)])
    if self.spawn_step == 0:
        # chains up to 4 sites
        new_models = [
            'pauliSet_1J2-zJz-d4',
            'pauliSet_1J2-zJz-d4+pauliSet_2J3-zJz-d4',
            'pauliSet_1J2-zJz-d4+pauliSet_2J3-zJz-d4+
            pauliSet_3J4-zJz-d4',
        ]

    elif self.spawn_step == 1:
        new_models = [
            'pauliSet_1J2-zJz-d4+pauliSet_1J4-zJz-d4+
            pauliSet_2J3-zJz-d4+pauliSet_3J4-zJz-d4', #
            ring
            'pauliSet_1J2-zJz-d4+pauliSet_1J3-zJz-d4+
            pauliSet_2J4-zJz-d4+pauliSet_3J4-zJz-d4', #
            square
        ]

    return new_models

```

Listing C.8: QMLA codebase setup

To run<sup>3</sup> the example ES, return to the local launch of Listing C.3, but change some of the settings:

```

prt=2000
exp=500
run_ghl=1
exploration_strategy=ExampleBasic

```

Listing C.9: local\_launch configuration for QHL

Run locally again as in Listing C.4, move to the results directory and analyse its contents as in Listing C.5.

QMLA stores results, and generates plots, over the entire range of the algorithm, i.e. the run<sup>4</sup>, instance and models. The depth of analysis performed automatically is the user control `plot_level` in `local_launch.sh`; for `_level=1`, only the most basic figures are generated, while `_level=6` generates plots for every individual model considered. For model searches across large model spaces and/or considering many candidates, excessive plotting can cause considerable slow-down, so users should be careful to generate plots only to the degree they will be useful.

We have just run quantum Hamiltonian learning (QHL) for the model in Eq. (C.1) for a single instance, using a reasonable number of particles and experiments, so we expect to have trained the model well. Instance-level results are stored (e.g. for the instance with `qmla_id=1`) in `Jan_01/01_23/instances/qmla_1`. Individual models' insights can be found in `model_training`, e.g. the training is summarised as in Fig. C.2, with the resultant dynamics in Fig. C.3.

In this case, because we are running QHL, there is little analysis of interest besides the single model's training, as shown already. It is of interest, however, to consider analysis on the run level, i.e. the average result of the training of the true model over several instances. This can be achieved by setting `num_instances` in `local_launch.sh`. For example, set `num_instances=10`, `prt=50`, `exp=10`, then launch and analyse the results (Listing C.5). In this case, since we are using few resources, the training will be ineffective, but ensure run-level analyses are generated in the results directory, e.g. in subdirectories `performance` and `champion_models`.

<sup>3</sup> Note this will take up to 20 minutes to run.

<sup>4</sup> Recall that a single implementation of QMLA is called an instance, and a series of instances which share the same target model is called the run.



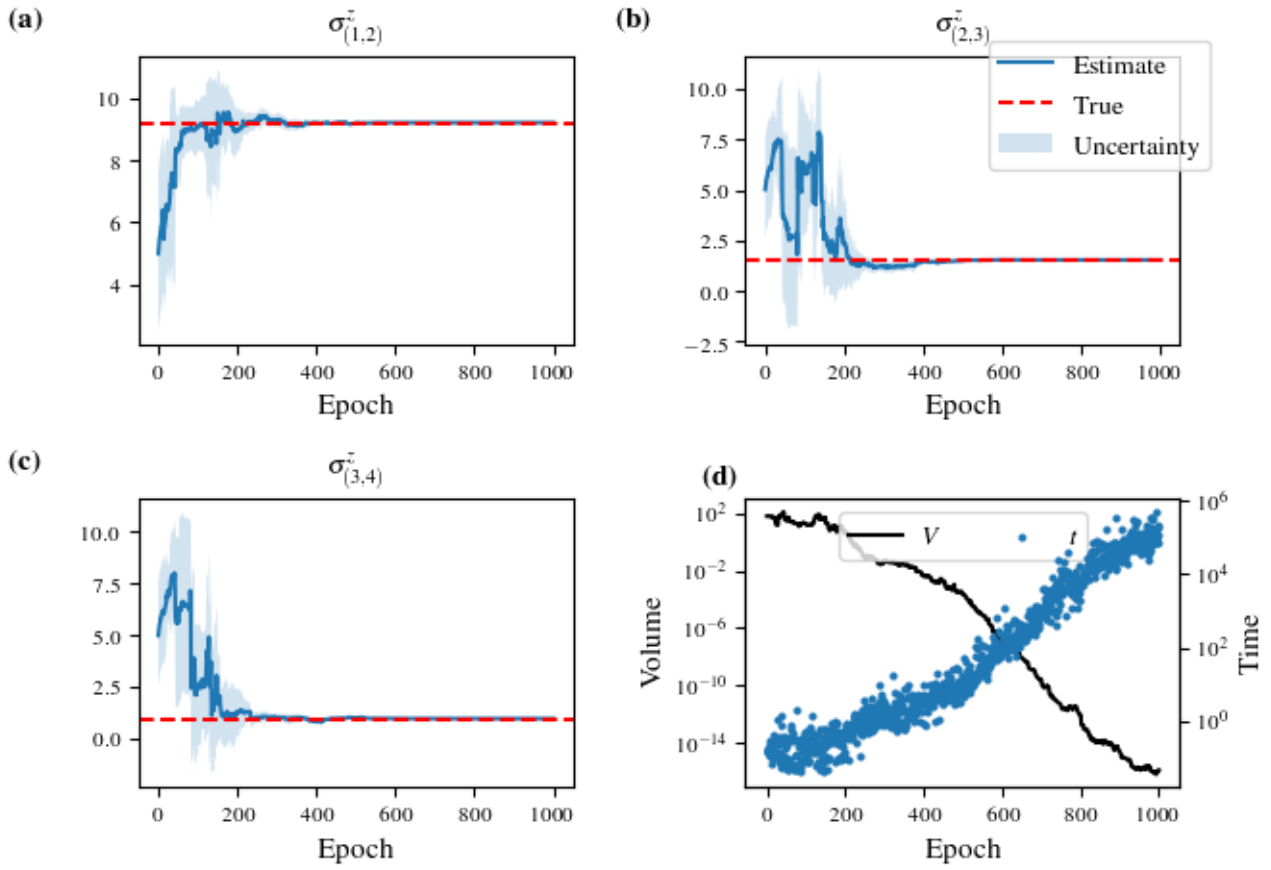


Figure C.2: QMLA plot: `model.trainings/learning_summary`. Displays the outcome of QHL for the given model: **(a)-c** show the estimates of the parameters; **(d)** shows the total parameterisation volume against experiments trained upon, along with the evolution times used for those experiments.

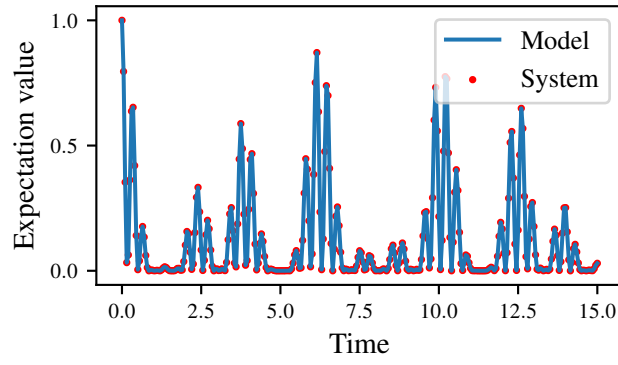


Figure C.3: QMLA plot: model\_trainings/dynamics. Displays the outcome of QHL for the given model: its attempt at reproducing data from  $\hat{H}_0$ .

## BIBLIOGRAPHY

---

- [1] Brian Flynn. Mathematical introduction to quantum computation, 2015. Undergraduate thesis.
- [2] Brian Flynn. Quantum model learning agent. <https://github.com/flynnbr11/QMLA>, 2021.