Continuous Error Correction in Quantum Dot Qubits

Stasiu Wolanski, Jesus College

2022 - 2023

Abstract

Woop!

Contents

0.1 Introduction . . . 1

1 Background **2**

1.1 Quantum error correction . . . 2

1.1.1 Quantum computing . . . 2

1.1.2 Error correction . . . 2

1.1.3 Continuous error correction . . . 2

1.2 Quantum Dot Qubits . . . 2

1.2.1 Semiconductor Heterostructures . . . 2

2 Theory **3**

2.1 Rapidly repeated error correction schemes . . . 3

2.2 Truly continuous error correction in quantum dot systems . . . 3

2.2.1 Scheme summary . . . 3

2.2.2 Theoretical treatment of scheme with second order perturbation theory . . . 4

2.2.3 Further questions related to tunnelling physics . . . 5

3 Methods **7**

3.1 Simulating the hubbard model . . . 7

3.1.1 Python extension: `FastHamiltoniser` . . . 7

3.1.2 Python class for Hubbard model construction: `Hamiltonian` . . . 9

3.1.3 Python class for time evolution of statevector: `Evolution` . . . 9

3.2 Truly continuous error correction prototype . . . 9

4 Discussion **11**

4.1 Conclusions . . . 11

Appendices **13**

A Detecting errors in ancilla qubits. **14**

B Locked states **15**

0.1 Introduction

Chapter 1

Background

1.1 Quantum error correction

1.1.1 Quantum computing

1.1.2 Error correction

1.1.3 Continuous error correction

1.2 Quantum Dot Qubits

1.2.1 Semiconductor Heterostructures

The Hubbard Model

Chapter 2

Theory

2.1 Rapidly repeated error correction schemes

2.2 Truly continuous error correction in quantum dot systems

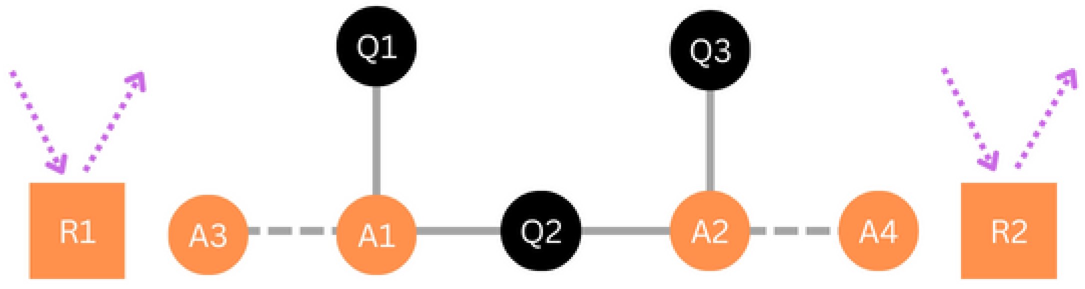

Figure 2.1: Schematic of the layout of qubits for the continuous error correction scheme. Q1, Q2 and Q3 comprise a single logical qubit. A1 to A4 are ancilla qubits used to detect bit flip errors in the logical qubits. Q1 and Q2 are continuously weakly coupled to A1 through a tunnel barrier, and likewise Q2 and Q3 to A2. R1 and R2 are resonators, which are additional quantum dots held at a higher electron occupancy than the qubits. These are probed using RF signals, and their coupling to the charge on the ancillas then allows the singlet/triplet state of the ancillas, as in [4].

2.2.1 Scheme summary

In this section a proposal for truly continuous error correction is developed. I thank Chris Long, Dr Mertig, Giovanni Oakes and my supervisor Dr Arvidsson-Shukur for contributing variously to the development of the ideas presented here. Recent developments in the field of RF reflectometry measurements enable the measurement of the singlet/triplet state of two adjacent qubits coupled via a tunnel barrier, by monitoring the rf reflection characteristics of a neighbouring electron reservior [4]. The codespace qubits are to be coupled pairwise to two ancilliary qubits whose function is to perform parity measurements, in such a way that any deviation from the codespace will result in a change of state of one or both of the ancilla qubits, depending on the error syndrome. The rate at which the error propogates to the ancilla qubit(s) is in the tens of nanoseconds, a faster timescale than either gate times (hundreds of nanoseconds) or readout (the state of the art [4] achives 6µs).

This scheme is potentially more efficient than a repeated error correction scheme like those analysed in section 2.1. One reason for this is that the 'dead time' after an error has occurred during which the system is vulnerable to a second error is limited by the time for the ancilla to be reset, which is also on the order of microseconds (e.g. see [2]). A second, more significant advantage is that the scheme does not require any quantum gates to be performed for the purposes of error sydrome detection, as such detection happens continuously and simultaneously with circuit gates. This reduces circuit complexity, and avoids the error accumulation associated with performing the syndrome measurements.

TODO flesh this out.

The proposed qubit layout is given in figure 2.1. The nature of the circuit requires a 2D array of quantum dot structures, of which there is not yet any published experimental demonstration, but a proposed architecture was proposed in [**Tadokoro2021˙2**]. In the scheme, the logical qubit is encoded in the *odd parity subspace* of three qubits Q1, Q2, and Q3. There are two pairs of ancilla qubits: A1, A3, and A2, A4. Each of these is initially prepared in the T_0 triplet state. As shown in the figure, each pair of qubits Q1 and Q2, Q2 and Q3, is weakly coupled to one ancilla qubit through a voltage-controlled tunnel barrier. If the couplings are sufficiently well-tuned, the ancilla qubits will remain as triplets and no tunneling will occurr. When a bit-flip error occurrs on one of Q1, Q2 or Q3, this will cause one or both (depending on the error syndrome) of the ancilla pairs to rotate into a singlet state. This allows (subject to complexities discussed in section 2.2.3) tunneling between the electrons in the affected pair, which is registered as a change in dispersive response of the nearby resonator. Thus the error syndrome is immediately detected as a signal in the continuously monitored reflectometry circuit.

2.2.2 Theoretical treatment of scheme with second order perturbation theory

Suppose the system is initally in an arbitrary codespace state, with each of the ancilla pairs starting in triplet states:

$$|\psi(t=0)\rangle = (\alpha\,|\uparrow_{Q1}\downarrow_{Q2}\uparrow_{Q3}\rangle + \beta\,|\downarrow_{Q1}\uparrow_{Q2}\downarrow_{Q3}\rangle)(|\uparrow_{A1}\downarrow_{A3}\rangle + |\downarrow_{A1}\uparrow_{A3}\rangle)(|\uparrow_{A2}\downarrow_{A4}\rangle + |\downarrow_{A2}\uparrow_{A4}\rangle)/2$$

First consider the case where no coupling is applied between the qubits. If we can engineer our system so that $\delta^z_{1,3} = E^z_{A1} - E^z_{A3} \ll 1/T_1$, and similarly $\delta^z_{2,4} \ll 1/T_1$ (with T_1 the timescale of bit flip errors), then the energy difference between the two branches of each qubit singlet is sufficiently small that rate at which the triplets rotate into triplets, and so register a tunnelling event on the reflectometry apparatus, is long compared to the error time and so not limiting. We can see this by considering a single ancilla branch:

$$|\psi(t)\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle + \exp(-2i\delta t)\,|\downarrow\uparrow\rangle) = \frac{1}{2}\,|T_0\rangle\,(1+\exp(-2i\delta t)) + \frac{1}{2}\,|S_0\rangle\,(1-\exp(-2i\delta t))) \tag{2.1}$$

, so that the amplitude of the singlet state goes as

$$|\langle S_0|\psi(t)\rangle|^2 = \sin(\delta t)^2$$

.

Now we apply a coupling between the following four pairs of qubits: Q1 and A1, Q2 and A1, Q2 and A2, Q3 and A2. We consider the four branches of the state in the computational basis of the ancilla qubits seperately, and apply second order perturbation theory to approximate the energy shift of each state due to the adiabatic application of a weak coupling, of strength t. The limits of this approximate treatment will be tested using a numerical simulation developed in chapter 3.

The standard formula for second order perturbation theory is (there is no first-order shift in energy as the perturbing terms are off-diagonal):

$$E^1_i \approx E^0_i + \sum_j \frac{|\langle\psi_i|V|\psi_j\rangle|^2}{E^0_i - E^0_j}$$

where E^0_i and E^1_i are the unperturbed and perturbed energies of state i, and H_0 and V are the basic and perturbative hamiltonians. If we set V to the hopping term in the hubbard model, then we obtain, for example,

$$\begin{aligned} E^1_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow} - E^0_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow} &= -\frac{t^2}{E^0_{\uparrow\updownarrow\uparrow,\varnothing\downarrow,\uparrow\downarrow} - E^0_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow}} - \frac{t^2}{E^0_{\uparrow\varnothing\uparrow,\updownarrow\downarrow,\uparrow\downarrow} - E^0_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow}} - \frac{t^2}{E^0_{\uparrow\updownarrow\uparrow,\uparrow\downarrow,\varnothing\downarrow} - E^0_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow}} - \frac{t^2}{E^0_{\uparrow\varnothing\uparrow,\uparrow\downarrow,\updownarrow\downarrow} - E^0_{\uparrow\downarrow\uparrow,\uparrow\downarrow,\uparrow\downarrow}} \\ &= -t^2\left(\frac{1}{\Delta_1} + \frac{1}{\Delta_2} + \frac{1}{\Delta_3} + \frac{1}{\Delta_4}\right) \end{aligned}$$

with

$$\begin{aligned} \Delta_1 &= U_{Q2} + \mu_{Q2} - \mu_{A1} + \mathrm{E}^z_{Q2} - \mathrm{E}^z_{A1} \\ \Delta_2 &= U_{A1} + \mu_{A1} - \mu_{Q2} + \mathrm{E}^z_{Q2} - \mathrm{E}^z_{A1} \\ \Delta_3 &= U_{Q2} + \mu_{Q2} - \mu_{A2} + \mathrm{E}^z_{Q2} - \mathrm{E}^z_{A2} \\ \Delta_4 &= U_{A2} + \mu_{A2} - \mu_{Q2} + \mathrm{E}^z_{Q2} - \mathrm{E}^z_{A2} \end{aligned}$$

Computational branch		Ancilla branch			
Subspace	branch	↑↓↑↓	↑↓↓↑	↓↑↑↓	↓↑↓↑
Codespace	↓↑↓	$2\Lambda_t$	$2\Lambda_t$	$2\Lambda_t$	$2\Lambda_t$
	↑↓↑				
E1	↑↑↓	$1\Lambda_t$	$1\Lambda_t$	$3\Lambda_t$	$3\Lambda_t$
	↓↓↑	$3\Lambda_t$	$3\Lambda_t$	$1\Lambda_t$	$1\Lambda_t$
E2	↓↓↓	$4\Lambda_t$	$2\Lambda_t$	$2\Lambda_t$	0
	↑↑↑	0			$4\Lambda_t$
E3	↓↑↑	$1\Lambda_t$	$3\Lambda_t$	$1\Lambda_t$	$3\Lambda_t$
	↑↓↓	$3\Lambda_t$	$1\Lambda_t$	$3\Lambda_t$	$1\Lambda_t$

Table 2.1: Table of the (negative) energy shifts due to coupling of each ancilla branch, of each computational branch, of each subspace.

, where the terms describe parameters of the Hubbard model described in section 1.2.1.

We note that $U_i \gg \Delta\mu_i, E_i^z$, and that the magnitude of U_i (order of a few meV) varies little from site to site , so that we may define $\Lambda_t = 2t^2/\langle U_i \rangle$. Then we have

$$E^1_{\mathrm{C}\uparrow\downarrow,\uparrow\downarrow} - E^0_{\mathrm{C}\uparrow\downarrow,\uparrow\downarrow} = \Delta E_{\mathrm{C}\uparrow\downarrow,\uparrow\downarrow} \approx \Delta E_{\mathrm{C}\uparrow\downarrow,\downarrow\uparrow} \approx \Delta E_{\mathrm{C}\downarrow\uparrow,\uparrow\downarrow} \approx \Delta E_{\mathrm{C}\downarrow\uparrow,\downarrow\uparrow} \approx -2\Lambda_t$$

, where we note the shift in energy is the same, within the above approximations, for both of the codespace branches, $|\uparrow\downarrow\uparrow\rangle$ and $|\downarrow\uparrow\downarrow\rangle$, and denote the energies of both branches by the subscript C. All branches of the ancilla state will therefore accure phase at the same rate as long as the state code qubits remains within the codespace.

When a bit-flip error occurs, the state will aquire a component in one of the error subspaces, denoted E1, E2 and E3 depending on which qubit has flipped. Applying the above analysis, we calculate the energy shifts of each of the ancilla branches of each computational branch in each codespace. These are given in table 2.1. These shifts result in the A1, A3, or A2, A4 ancilla pairs rotating from triplets to singlets, or both, depending on the error subspace. To see this, consider the E1 error subspace, into which the state moves when the Q1 qubit experiences a bit-flip error. If both ancilla pairs are in triplet states at the point when the system transitions into the E1 subspace from the codespace, the state undergoes the following evolution [1]:

$$\begin{aligned}
|\psi(t)\rangle =& \alpha\,|\downarrow\downarrow\uparrow\rangle\,(e^{-3i\Lambda_t t}\,|\uparrow\downarrow\uparrow\downarrow\rangle + e^{-3i\Lambda_t t}\,|\uparrow\downarrow\downarrow\uparrow\rangle + e^{-i\Lambda_t t}\,|\downarrow\uparrow\uparrow\downarrow\rangle + e^{-i\Lambda_t t}\,|\downarrow\uparrow\downarrow\uparrow\rangle)/2 \\
+e^{-i\delta_{Q1,Q3}t}&\beta\,|\uparrow\uparrow\downarrow\rangle\,(e^{-i\Lambda_t t}\,|\uparrow\downarrow\uparrow\downarrow\rangle + e^{-i\Lambda_t t}\,|\uparrow\downarrow\downarrow\uparrow\rangle + e^{-3i\Lambda_t t}\,|\downarrow\uparrow\uparrow\downarrow\rangle + e^{-3i\Lambda_t t}\,|\downarrow\uparrow\downarrow\uparrow\rangle)/2 \\
&\approx e^{-3i\Lambda_t t}\alpha\,|\downarrow\downarrow\uparrow\rangle\,(|\uparrow\downarrow\rangle + e^{2i\Lambda_t t}\,|\downarrow\uparrow\rangle)(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle)/2 \\
&+ e^{-i\Lambda_t t}\beta\,|\uparrow\uparrow\downarrow\rangle\,(|\uparrow\downarrow\rangle + e^{-2i\Lambda_t t}\,|\downarrow\uparrow\rangle)(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle)/2
\end{aligned}$$

, where the $\delta_{Q1,Q3}$ phase term accounts for the difference in energy between the two branches, which is small compared to Λ_t, and irrelevant to our conclusion, so we ignore it. By comparison with equation 2.2.2, we see that the A1,A3 ancilla pair in both branches will rotate into a triplet state with a characteristic timescale $1/\Lambda_t$. The same analysis yields for E2 error subspace:

$$\begin{aligned}
|\psi(t)\rangle \approx& \alpha\,|\downarrow\downarrow\uparrow\rangle\,(|\uparrow\downarrow\rangle + e^{-2i\Lambda_t t}\,|\downarrow\uparrow\rangle)(|\uparrow\downarrow\rangle + e^{-2i\Lambda_t t}\,|\downarrow\uparrow\rangle)/2 \\
+e^{-4i\Lambda_t t}&\beta\,|\downarrow\downarrow\uparrow\rangle\,(|\uparrow\downarrow\rangle + e^{2i\Lambda_t t}\,|\downarrow\uparrow\rangle)(|\uparrow\downarrow\rangle + e^{2i\Lambda_t t}\,|\downarrow\uparrow\rangle)/2
\end{aligned}$$

, so that both pairs of ancillas rotate into singlets. The E3 case is identical to the E1 case with the appropriate qubits exchanged, and leads to only the A2, A4 qubit pair rotating into a singlet.

2.2.3 Further questions related to tunnelling physics

Much work has been done optimising the implementation and and examining the classical physics of reflectometry as a tool for quantum measurement. For a review see, e.g. [5], especially section VIII. During the project the author has not had time to study the physics behind this technique in detail. It appears, however, that present analyses treat the system semi-classically in assuming that an electron tunnels, or does not, within the measurement period (i.e. the period of time that the tunneling barrier between the pair of qubits is lowered). They then seek to maximise

[1] We assume that the resultant state is approximately a stationary state of the perturbed hamiltonian.

the fidelity with which this charge-shifting event, or lack thereof, is detected. This tunneling event is interpreted as a projective measurement of the singlet/triplet system at the point of measurement, with probabilities of either state given by Born's rule. In the present case this is an insufficient analysis: at any given time, there is necessarily some finite component of the electron pair state that is a singlet, and we need to establish a rate at which tunnelling occurs as a function of the amplitude of this component. A full treatment of this problem probably needs to be formulated in terms of quantum master equations and the theory of quantum decoherence [6].

Time has not permitted me to look into this issue, so at the suggestion of my supervisor I will make the simplifying assumption that any ancilla pair that aquires more than a 10% square-amplitude of singlet immediately tunnels and is hence registered by the dispersive readout device with empirically determined fidelity.

Chapter 3

Methods

3.1 Simulating the hubbard model

This section will discuss the methodology used to perform simulations of scheme described in section 2.2 on a hubbard model of a multiple quantum dot system. The treatment of the error correction scheme described in section 2.2 is necessarily highly approximate due to the complexity of the high-dimensional hilbert space involved (for 7 electrons on 7 sites, the occupation basis is of dimension 3432) and the large number of parameters of the Hubbard model, so makes multiple simplifying assumptions, chiefly the application of second-order perturbation theory and ignoring the site-to-site variance in certain model parameters. It is therefore valuable to simulate the Hubbard model numerically to test the validity of these assumptions in various parameter regimes.

My supervisor Dr. Arvidsson-Shukur have developed a code (yet unpublished), `DotHamiltoniser`, designed to generate Hubbard hamiltonians in an occupation basis given a set of physical parameters (E_i^z, U_i, etc.), as well as to perform low-dimensional time-evolution simulations. This was very helpful at the start of my project to visualise the energy levels of the system and their dependence on coupling, but was not designed for higher dimensional time evolution, and could not be used for this purpose due to performance limitations.

I therefore designed a Hamiltonian simulation code, inspired by the work of Dr Arvidsson-Shuker et al., with the ability to handle higher dimensional systems. The system has three main components:

1. A python extension, `FastHamiltoniser`, written in C, C++ and CUDA, which provides highly performant specialised sparse matrix operations for use in Hamiltonian time evolution;
2. `Hamiltonian`, a python class that builds a Hubbard model Hamiltonian from physical parameters; and
3. `Evolution`, a python class which takes as input a Hamiltonian and a set of continuous time instructions to produce a time evolution of a given starting statevector.

3.1.1 Python extension: `FastHamiltoniser`

Structure of extension

The purpose of the simulator is to solve the Schrödinger equation

$$\dot{\psi} = -iH\psi \tag{3.1}$$

(here expressed in natural units $\hbar = 1$) where ψ is a vector in the occupation basis of a Hubbard Hamiltonian. In order to use Runge-Kutta [1] methods of integration, we need to be able to calculate this derivative efficiently at an arbitrary time and with arbitrary input. H is typically very sparse, with the only off-diagonal terms given by magnetic drives and hopping couplngs, of which typically only a subset of size linear in the number of qubits are non-zero at a given time.

To perform this calculation efficiently, I implemented the following data structure in C. A `HamiltonianMatrix` object, illustrated in figure 3.1, contains basic data about the dimension of the array, as well as an array of pointers to objects of type `Permuter`. Each of these `Permuter` objects stores three arrays of equal length, two of integers representing source and target vector indices, and one of floats representing matrix coefficients, where each entry represents a matrix element. `Permuter`s also store a complex coefficient, which can be changed over time by writing to a single variable, and a boolean representing whether it is 'on' or 'off'. The internal arrays are determined in advance of the simulation by python code, and thereafter only the strength and activations of the `Permuter` objects need be

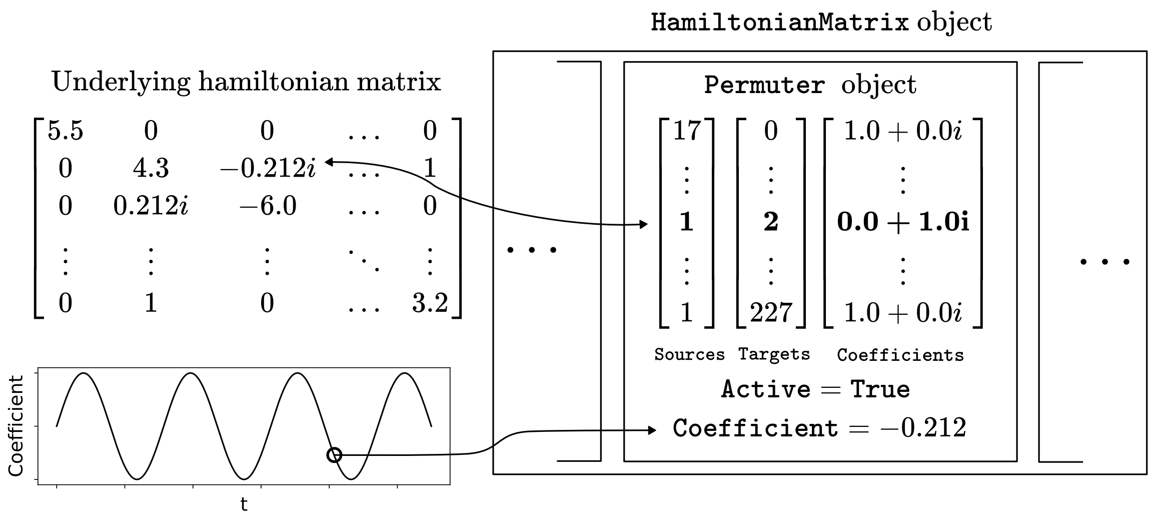

Figure 3.1: How sparse Hamiltonian matrices are represented in the custom python extension `FastHamiltoniser`. The non-zero matrix elements are grouped into objects of type `Permuter`, whose structure is calculated in advance but whose activation and multiplicative coefficients are modified efficiently as runtime.

modified during the simulation. For example, one such object provides all the on-diagonal elements of the matrix, and another might provide all the off-diagonal elements associated with a particular J-coupling between two qubits.

The calculation 3.1 can then be performed by iterating over the active permuters and building up the derivative vector according to the permutation and coefficient arrays.

Interaction picture calculation

A further requirement that favoured the development of a custom C backend was the ability to perform simulations in the interaction picture. Performing simulations in the interaction picture is necessary due to the large differences in energy between eigenstates, for example due to the U_i on-site repulsion in some states. With any choice of reference energy in a static basis, there are always states whose phase rotate much faster than the timescale of the evolutions that I wish to simulate, meaning there is no suitable step-size that captures the desired phenomena within reasonable computational time with the simulation remaining stable. Instead, we perform the following transformation to the interaction picture:

$$\begin{aligned}\dot{\psi}_i &= -i\sum_j H_{ij}\psi_j \\ \psi_i \rightarrow \phi_i &= e^{-iH_{ii}t}\psi_i \\ \Rightarrow \dot{\phi} &= -i\sum_{j\neq i} e^{-i(H_{jj}-H_{ii})t}H_{ij}\phi_j\end{aligned}$$

.

Since most off-diagonal terms H_{ij} are zero at most points in time, this results in most states being stationary and lowering the rate of rotation of affected states to a longer timescale that can be simulated. It comes at the cost, however, of introducing a time-dependence into the hamiltonian. I wrote a C function to perform the matrix multiplication, only calculating the phase factors as required.

GPU acceleration

The calcluation to be performed is parallel in nature, and thus a candidate for GPU acceleration. I wrote the requisite kernels[1] to perform the matrix multiplication in CUDA, a variant of C++ developed by Nvidia to control their graphics processing units ('GPU's) [3], one of which (a P620 Quadro) is installed in my laptop. On testing, the CUDA variant

[1]This presented a lot of technical challenges as the process for incorporating CUDA C++ into python C extensions and compiling them is not documented and very technical. I made a template project with the setup to use in the future: https://github.com/Stasiu51/TemplateCudaEx

outperformed the CPU calculation only for systems with more than around 15 qubits, more than required and where the speed of computation is too slow for practical use anyway (< 10 iterations per second, several orders of magnitude too slow for the simulations of the present project).

3.1.2 Python class for Hubbard model construction: `Hamiltonian`

This is a python class (and supporting code) that implements a Hubbard Hamiltonian, using `FastHamiltoniser` as a backend to store the underlying matrix. In particular it

1. creates a list of basis vectors, represented as bit strings (with convenience functions for printing them in canonical notation), that span the fock basis of a system with a specified number of sites and number of electrons;
2. defines creation and annhilation operators on these states, which it uses to
3. construct the diagonal `Permuter` object that represents the self-energy terms due to chemical potential, coulomb repulsion, and static magnetic z-fields; and to
4. construct the off diagonal `Permuter` objects generated by hopping terms and x and y magnetic fields. It also
5. provides transformations to and from the interaction picture; and
6. provides functions for creating projector and X matrices in the fock basis, which are more involved to calculate than in the computational (single electron per site) basis.

3.1.3 Python class for time evolution of statevector: `Evolution`

This class takes a reference to a `Hamiltonian` object, and a set[2] of objects that subclass the abstract base class `TimeStep` whose role is to provide instructions as to how to modify the parameters of the Hubbard model at given time. For example, one such object instructs the simulation to ramp a set of hopping couplings to a specified value over a given period of time, and another might oscillate the strength of the x component of the applied magnetic field at a given frequency and phase.

The `scipy` routine `solve_ivp` is used along with the calculations of the derivative described above in a Runge-Kutta 5th order [1] integration routine to calculate the evolution of a given initial statevector. The routine is designed to estimate the errors and dynamically choose a step size for the integration. This works poorly in the present use case, as the calculated (vector-norm) error scales with the number of non-zero elements of the vector. Instead, the code allows the user to manually specify a step size, typically around 1/10 of the smallest relevant timescale of the system, and checks the continued normalisation of the statevector as an imperfect check on the stability of the simulation.

Some demonstrations of the simulator and the results of a benchmarking exercise are shown in figure 3.2.

3.2 Truly continuous error correction prototype

This section explains how the software described above was used to simulate the 'truly continuous' error correction scheme developed in section 2.2. In particular, we want to explore suitable physical parameters, such that when we initialise the system in an initial codespace state, and adiabatically turn on a weak coupling between the relevant qubits

- the ancilla pairs remain in triplet states as long as the system remains in the codespace, for at least a time scale longer longer than ; and
- when a bit-flip error is artificially introduced, one or both of the ancilla pairs rotates into a singlet, as appropriate to the error syndrome.

Physical parameters

It is useful to survey the feasible ranges of the various physical paramters of the experiment based on experimental work to date. These can be found in table **??**

[2]There is in fact only one such object, as they are defined in a recursive tree structure.

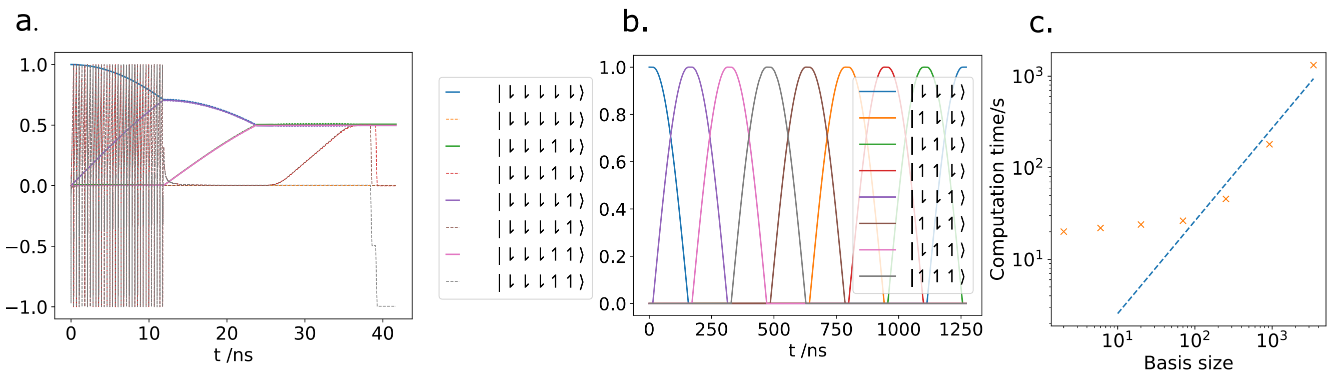

Figure 3.2: Testing the capabilities of the simulation software. **a.** A plot of 5-qubit system showing the amplitudes (solid lines) and phases (dashed lines) of significant basis states. The plot demonstrates of a CPHASE gate acting on the last two qubits, so that the $|{\downarrow\downarrow\downarrow\uparrow\uparrow}\rangle$ state acquires a phase shift of π. First an X gate is applied to each of the two qubits, then a coupling between them is applied, and finally a z-rotation is applied by increasing the static field (in practice the last step is unecessary). **b.** A sequence of X flips on a 3-qubit system. This circuit is generalised to N qubits to benchmark the simulator, with results given in **c.**. The results indicate a consistent speed of around 7.5 million statevector derivative elements calculated per second.

Chapter 4

Discussion

4.1 Conclusions

Bibliography

[1] J.C. Butcher. "A history of Runge-Kutta methods". In: *Applied Numerical Mathematics* 20.3 (1996), pp. 247–260. ISSN: 0168-9274. DOI: https://doi.org/10.1016/0168-9274(95)00108-5. URL: https://www.sciencedirect.com/science/article/pii/0168927495001085.

[2] Takashi Nakajima et al. "Quantum non-demolition measurement of an electron spin qubit". In: *Nature Nanotechnology* 14 (2019). DOI: 10.1038/s41565-019-0426-x. URL: https://doi.org/10.1038/s41565-019-0426-x.

[3] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[4] G A Oakes et al. "Fast High-Fidelity Single-Shot Readout of Spins in Silicon Using a Single-Electron Box". In: (2023). DOI: 10.1103/PhysRevX.13.011023.

[5] Florian Vigneau et al. "Probing quantum devices with radio-frequency reflectometry". In: (2023).

[6] Howard M Wiseman. "Quantum trajectories and quantum measurement theory". In: *Quantum and Semiclassical Optics: Journal of the European Optical Society Part B* 8.1 (1996), p. 205.

Appendices

Appendix A

Detecting errors in ancilla qubits.

Appendix B

Locked states