Final Report: Parallelizing the L-BFGS algorithm on GPUs for Quantum Control problems

Author: Max Chandler

Supervisor: DR Frank Langbein

Student Number:	C1125169
Module:	CM3203
Module Title:	One Semester Individual Project
Credits:	40

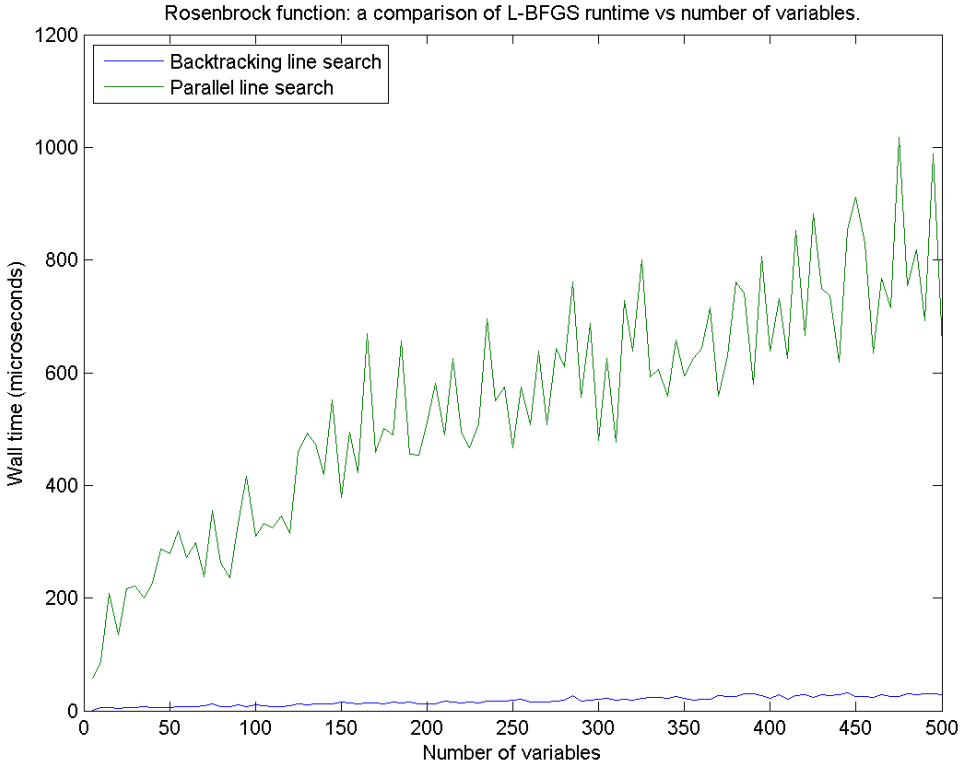
Abstract

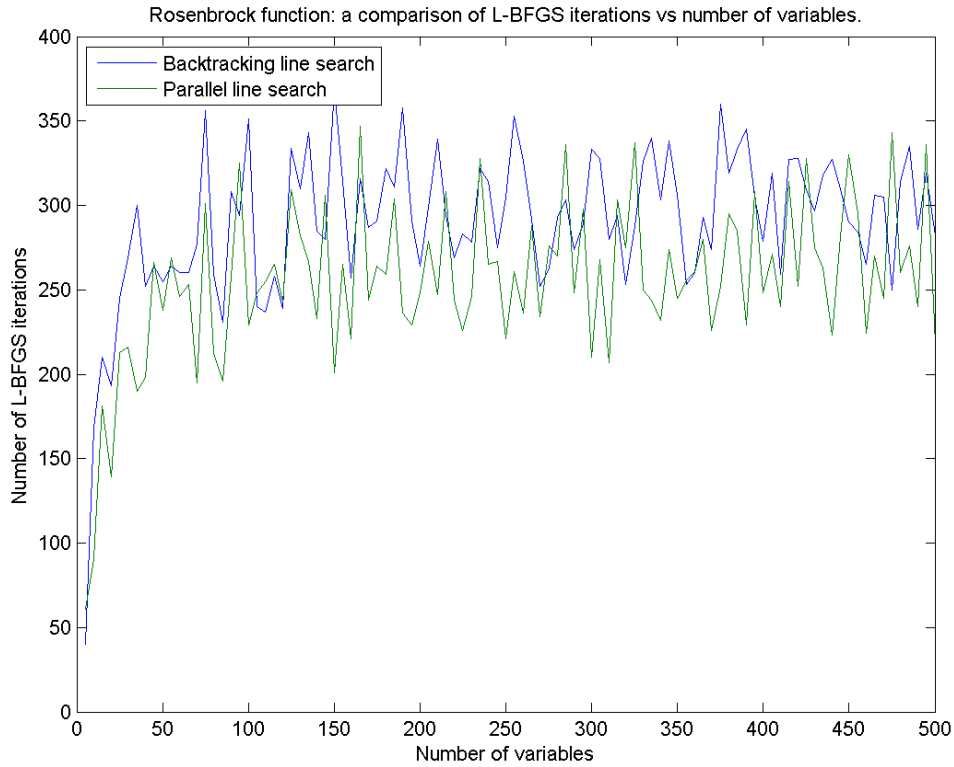
Control of quantum spin networks is an inherently difficult problem, where the search space has many local extremes. The difficulty of this problem is due to the quantum mechanical behavior of the spins, which are central to the operation of the network. Due to the nature of the problem, optimization of system controls is costly and time consuming. By improving on current techniques through the use of parallelization techniques, it is possible to reduce runtimes. This dissertation aims to investigate the areas for potential speedup of currently used methods.

Acknowledgements

I wish to express my sincere gratitude to Dr. Frank Langbein for his guidance on this project. His help and time during his sabbatical year was highly valued.

Table of Contents





Introduction

Control of quantum spin networks is an inherently difficult problem, where the search space has many local extremes. The difficulty of this problem is due to the quantum mechanical behavior of the spins, which are central to the operation of the network. Due to the nature of the problem, optimization of system controls is costly and time consuming. By improving on current techniques through the use of parallelization techniques, it is possible to reduce runtimes. This dissertation aims to investigate the areas for potential speedup of currently used methods. Progress will be monitored by evaluation of the improvement gained in terms of accuracy, precision, robustness and runtime.

The focus is on the control of spin chains, which simulates a router, where optimal controls maximize the probability of information transfer from one spin to another. Individual spins are controlled using magnetic control fields. The magnetic fields affect the spin of each individual element by aligning the spin with the magnetic field, the stronger the field the greater this effect is. When a relatively large control is placed on an individual spin, it effectively isolates it from the network, as there is a much-reduced chance of spin state transfer when it is aligned with the magnetic field.

The optimization part of this project is aiming to provide the best controls, which maximize the probability of successful data transfer between spins. For example, transfer from spin 1 to spin 3 is desired; the system is prescribed a fixed time t, where the controls are the only variable in the system. The probability of the system is then improved by optimizing the controls, in this case by a gradient based optimization algorithm; L-BFGS.

Parallelization of these methods will allow for research to be conducted in a timelier manner, as the current methods are extremely time consuming. In this paper, there is an exploration of improvement of accuracy as well as speed; parallelization allows for more functions and evaluations to be performed at the same time, providing a more detailed description of the search space.

Aims

The main aim of this project is to research the possibilities of parallelization of currently serial methods for achieving optimal system states. Through parallelization of current methods, the major improvement will be in runtime, where many processes can take place simultaneously rather than just one. There is also the possibility of increased accuracy, as current methods try to limit the number of evaluations to conserve time spent processing, where here is it possible to perform many evaluations in one time step.

By parallelizing current methods, it is possible to perform more calculations in less time that may yield more accurate results. The major idea here is to develop an understanding of how constant controls over time can be used to reach a high fidelity, which indicates the probability of information transfer.

This research can directly be applied to information transfer in quantum spin networks. Where current research is looking into the behavior of spin network, where

eventually it will be possible to model an entire network, where rings are used as routers to pass information down 'wires' (spin chains).

Background

In order to understand the problem addressed in this paper, it is first important to understand the background to this complex problem. As this system is based on quantum mechanical behavior, I have provided there is a brief introduction into the field of quantum mechanics. This is followed by an introduction to spintronics and finally optimization techniques.

The final system is a network where transfer of data is based on quantum mechanical phenomena. Data is stored in the spin state of a particle. For simplicity; consider each 'spin' as a spinning top, the rotation turning around its own axis. Finally, control is based around the use of magnetic fields, which are used to manipulate the probability of transfer of data around the network. However, due to the uncertainty of quantum mechanics, it is extremely difficult to find appropriate controls to maximize transfer probability. This is where an optimisaion algorithm comes in, where it searches a large space in an attempt to provide an answer to the value of these controls.

Quantum mechanics and Spintronics

Quantum mechanics is a major part of 'modern' physics, it aims to explain the incredibly small, where behavior is often counterintuitive and uncertain when compared to 'classical' physics; such as Newton's laws of motion. In quantum mechanics, the key principle is the notion of matter being both a wave and a particle, where the de Broglie wavelength λ is related to its motion through the Planck constant h.

$$\lambda = \frac{h}{p}$$

This theory applies to all matter, and can be used to explain quantum objects. It has been displayed in the double-slit experiment, where electrons created an interference pattern that is concurrent with a wave (Eichmann, et al., 1993).

Quantum mechanics is of particular interest to computer scientists as it utilizes quantum mechanical phenomena, such as superposition, the uncertainty principle or entanglement. These phenomena allow a qubit to represent an extremely large set of data; whereas in classical computing only 0s and 1s can be represented by one bit. The key idea here is that one qubit can represent a superposition of many states. At measurement, the wave function collapses into one single state out of the entire set of states in the superposition.

Shor's algorithm represents an efficient application of a quantum computer, whereby a classical computer fails to achieve the same outcome in reasonable time. The algorithm is focused on factoring integers to find their prime factors. This is a key component of many modern cryptography algorithms, by providing an algorithm that factors primes quickly Shor's algorithm breaks down the security of public-key cryptography, such as RSA (Vandersypen, et al., 2001).

The focus of this paper is on the transfer of information between spins in a network that could represent a Northbridge in a computer, passing qubits from memory

to the processor. Research into quantum control will aid the development of quantum devices, where in the future it may be possible to have a separate quantum processor in each personal computer for specialized tasks.

Bra and Ket notation

Throughout this paper, I have used special notation found in quantum mechanics called bra-ket notation. It was first introduced by the physicist Paul Dirac (1939). Bra-ket notation is used to represent an abstract state of a system, where $< \phi|\psi >$ represents the probability of ψ collapsing into state ϕ.

ψ represents the superposition of states, where the spin of a particle (or system of particles) is thought to be in every state possible up until the point of measurement. When measured the wave function collapses and the particle is in one of the possible states; the one being observed. For simplicity in this paper a bra $< |$represents a row vector and a ket $| >$ represents a column vector.

Schrödinger's equation

Schrödinger designed an equation that describes how the state of a quantum system evolves over time (1926). The quantum system model in this project is based around this equation.

$$i\hbar\frac{\partial}{\partial t}\Psi = \hat{\mathrm{H}}\,\Psi$$

$$\frac{\partial\psi(t)}{\partial t} = i\hbar H_t\psi(t)$$

The Schrödinger equation is based around calculating the wave function ψ, which describes the quantum state of one or more particles. In the above equations, i represents the imaginary part of the quantum equation; which in simplistic terms allows mathematics to accurately model quantum mechanics (Baylis, Hushilt, & Wei, 1992).

The constant $\hbar$ represents the reduced Planck constant, which describes the quantum of action in quantum mechanics. In this application, the reduced Planck constant is used, as we are investigating angular frequency it reduces the complexity of the mathematics by absorbing the value 2π. The reduced Planck's constant is equal to:

$$\hbar = \frac{h}{2\pi}$$

Finally, H represents the system Hamiltonian. That is the operator that describes the total energy of the system, and all of the possible states. This is often thought of as a spectrum of possible states, where only one exists at the point of measurement.

$$|\psi(t)> = expm(-i\hbar Ht)\cdot|\psi(0)>$$

The final equation above represents the calculation of wave function of a system at the given time. Later on in this paper I will engage in discussion of what this implies about the state of the system.

Spintronics

Spintronics is the study of an electrons spin state, where the spin and the electrons charge can be used to store information. The spin state of a particle usually has two positions, up or down, much like a bar magnet. A spintronic device can manipulate an electrons spin state by applying a magnetic field to align the spin to the

magnetic field. As an application, this can lead to energy efficient memory devices; such as magnetoresistive random-access memory (MRAM). Due to the efficiency of these devices they are thought to be the future of memory, eventually becoming the 'universal' memory for all computational devices (Åkerman, 2014). However, current spintronics devices are in most homes in the form of hard-drive read/write heads, which manipulate the spin state of ferromagnetic film to store data.

Spintronics is also the foundation of magnetic imaging techniques, where devices such as MRI and NMR machines manipulate the spin state of protons. These protons emit a radio frequency that can be used to identify the density or type of matter at a given location. Once a large dataset of these frequencies has been collected, it is possible to create a detailed image of the object.

In this paper, the focus is on modeling a spin network and attempting to transfer qubits of information by manipulating the magnetic field on the network. The study of spin transfer is modeled by transformation theory in quantum mechanics, where a state vector models a corresponding quantum system. Transformation theory can be applied to model the change of a system over time, where in this project it is employed in the form of the Schrödinger equation. Specifically, rings of spins are explored, which are designed to model the behavior of a qubit router.

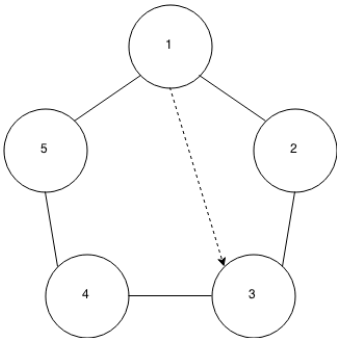


FIGURE 1

In figure 1, the sold lines represent the links in a 5-spin network; in a classical network, the information would pass in a logical manner around the network between neighbors (for example, if each node represented a computer rather than a spin). In quantum networks, data does not necessarily transfer to nearest neighbors; it tends to behave in a counterintuitive way where the links appear to be symbolic. If the arrow were to represent a target in the system, for example transferring a qubit of data from spin 1 to 3, what controls would this require? This is the core of this paper, and for simplicity the time of the transfer is fixed, so the main question to be answered is; *given a time 't', what controls would maximize the transfer probability of a qubit of information from spin 1 to 3?*

Spin Network Control

As discussed in the previous section, the focus of this project is to provide adequate controls for a closed quantum system that will maximize transfer at a pre-determined time. For this application, the quantum system is a ring, containing an odd number of spins. Each spin is a $-\frac{1}{2}$ spin particle, as these mimic a real world implementation if this was based around a proton's spin. The controls represent the bias of a magnetic field placed on each spin in the network; the values are arbitrary. The spins are connected by nearest neighbor coupling, allowing information to freely move

around the ring. This sort of transfer is currently available in a spin network inside a diamond controlled by magnetic fields (Hai-Jing, et al., 2013).

Due to the difficulty of this problem, it is important to limit the number of variables to optimize for the given time; in this instance only the magnetic field strength is being optimized. Unlike other studies, where controls are switched over time (Schirmer & Langbein, 2014), the controls in this system are set for the entire duration of the simulation. The optimization algorithm then will vary the values representing the bias of the controls on each spin with the aim of achieving the best transfer probability.

Optimization

Optimization is a branch of mathematics concerned with finding optimal values for a given function or equation. In general terms, they attempt to minimize or maximize the return from a system by varying the inputs. In this application, focus will be on minimizers, which are defined by the following equation:

$$\min_{x \in \mathbb{R}} f(x)$$

There are many different types of efficient optimization methods available, each with their own benefits. In the following section, there is a discussion of the different types, their benefits and applications.

Evolutionary Algorithms

Evolutionary algorithms are based around a population of candidates randomly generated to begin. Overall, they perform well, as they do not make any assumption about the problem, and this allows for a broad search space.

Algorithm outline

- Generate a population of size *x* with random initial values.
- *(Repeat until a termination condition is met)*
    - Select best candidates based on their relative fitness
    - Pair off best candidates to be parents
    - Create children through mutation and crossover operators
    - Replace the least fit individuals with the new children

Evolutionary algorithms are suitable for machine learning applications, where the goal is to investigate a space with little pre-existing knowledge. They are particularly suited to engineering problems, where usually a structure could benefit from optimization. An example of this is with aerodynamics of a vehicle, where a genetic algorithm can make small changes to the shape to reduce the effect of wind resistance.

Swam Intelligence

Swarm algorithms are designed to mimic natural real world behaviors of animals or organisms. They are designed to create a collective behavior from a number of individuals, where the interaction creates intelligent decisions. The behavior of individuals in the system can be designed to imitate many different styles of swarm, such as, birds flying, ant colonies, bacterial growth, or fish schools.

Swam intelligence is useful when attempting to mimic or detect real world swarming behavior, such as crowd simulation in CGI for films. Other applications

include network investigation and routing problems such as air-traffic control or autonomous vehicles.

Simulated Annealing

Simulated annealing is an algorithm designed around annealing: the process of controlling the cooling of an object to reduce the number of defects in the final product. When this technique is applied to an optimization problem, it searches for an acceptable answer in a prescribed time. The process itself starts with a relatively large search space, in which large steps are taken in an attempt to find the optimum value. As the temperature beings to decrease, the step sizes begin to become restricted with the intention of improving already found optima. Simulated annealing guarantees a result in a prescribed time, however the result is not guaranteed to be optimal.

Algorithm outline

- Generate random initial solution
- Evaluate value of solution based on a cost function
- *(Repeat until temperature reaches 0)*
    - Generate random neighboring solution, where step size based on the current temperature
    - Compare and move to the better of the two solutions
    - Reduce temperature

Simulated annealing is often applied to a discrete search space, and can provide a relatively accurate approximation of the global optima. It is a good algorithm when the landscape is complex, as the algorithm does not get stuck in local optima in the early stages.

Gradient-based Optimization

Gradient-based optimization methods are applicable where either an approximation of the gradient or the gradient itself can be calculated. These methods are usually a good choice when the gradient is easy to calculate or already available.

First order

GRADIENT DESCENT

Gradient descent is focused on moving towards the minimum (or maximum) by taking steps towards the negative of the search space. The step in gradient descent is calculated by a line search method, which returns a step size of α that sufficiently reduces the objective function.

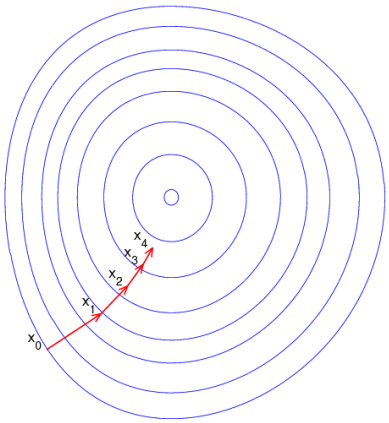


FIGURE 2 (ALEXANDROV, GRADIENT DESCENT.PNG, 2004)

Gradient descent has limitation when approaching minimums, or where the search space has narrow valleys that are close to a minimum. This is due to the simple design of calculating a descent direction.

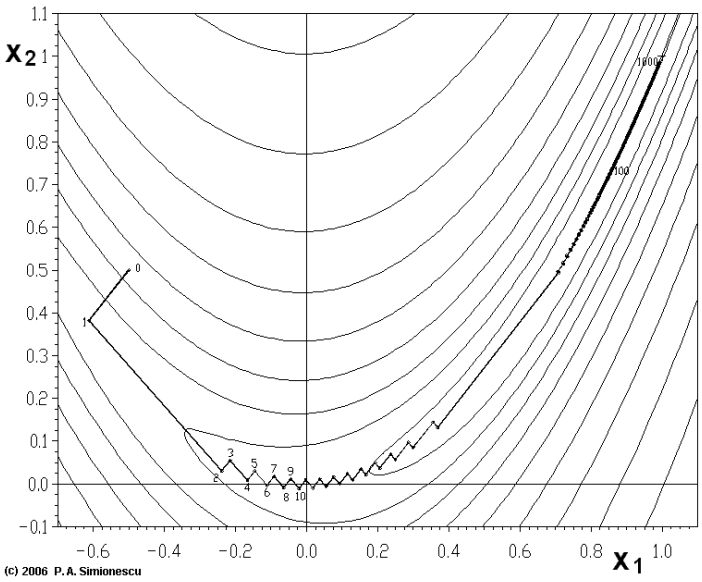


FIGURE 3 (SIMIONESCU, 2006)

Figure 2 represents the issue that gradient descent methods have when reaching a minimum value. The function above is the Rosenbrock function, where towards a minimum the function is extremely difficult to solve, as the gradients are minute. In figure 3 above, the gradient descent method reaches over 1,000 iterations when aiming to find the minimum which is at $f(1,1,\ldots,n)$, where n is the number of dimentions.

CONJUGATE GRADIENT

The conjugate gradient method is applied to linear equations that produce symmetric positive definite matrixes. It will find the exact answer in n steps, where n is the number of unknowns. The conjugate gradient method differs from gradient descent in that there is no line search is used to calculate the step towards the minimum to take.

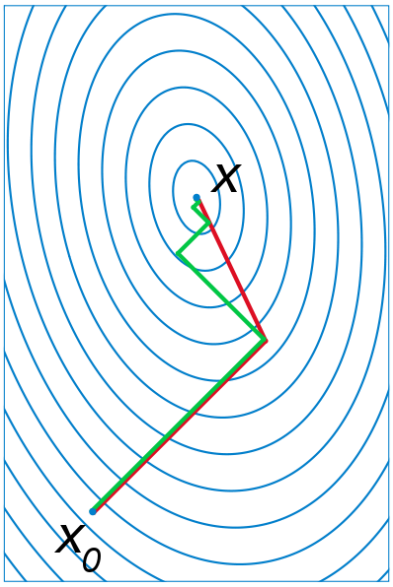


FIGURE 4 (ALEXANDROV, 2007)

Figure 4 displays the performance difference when finding the optima of a function, the red line represents a path taken by a conjugate gradient method, and the green line represents the path of a gradient descent method. The conjugate gradient method is more expensive to calculate than gradient descent but it is much more efficient when converging towards the minimum.

Quasi-Newton methods

When trying to find global optima, Quasi-Newton methods, aim to find a stationary point where the gradient is equal to 0, which suggests a global optimum. These are similar to Newton's method, however they are used when the full Jaccobian or Hessian matrix is too expensive to compute, or not available. Newton's method iteratively computes values of *x*, based on the following formula:

$$x_{n+1} = x_n - [J_g(x_n)]^{-1} g(x_n)$$

Any method that subtitles the exact Jaccobian matrix in Newton's formula is a quasi-Newton method.

Broyden-Fletcher-Goldfarb-Shanno (BFGS)

BFGS is a popular quasi-Newton method that calculates an approximation of the Hessian matrix for gradient calculations. It is an iterative optimization algorithm for solving unconstrained non-linear problems. BFGS calculates and stores a dense $n * n$ approximation of the Hessian matrix, which leads to a large memory requirement for large problems.

Algorithm outline

1. Calculate search direction p_k by solving: $B_k p_k = -\nabla f(x_k)$
2. Line search to get initial value of α_k
    a. Update $x_{k+1} = x_k + \alpha_k p_k$
3. $s_k = \alpha_k p_k$
4. $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

5. $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$

B_0 can be initialized as $B_0 = I$, so the first step is essentially gradient descent.

Limited-memory-BFGS (L-BFGS)

L-BFGS is a popular alternative to BFGS when the problem size is large and available memory is a constraint on the search space. Instead of calculating the Hessian matrix, L-BFGS maintains a history of the last *m* updates that are used in place of the Hessian matrix.

Approach

Tools

For this project, it was important to select appropriate tools, given the relatively small amount of time needed to complete the project. The following tools have been chosen to allow the development of a system that will meet the aims of this project.

Software

Matlab

Matlab was chosen for this project due to the rapid prototyping and large range of built in complex functions that are necessary for this project given the timeframe. Matlab natively handles imaginary numbers, where as C and C++ have liberates to handle them, however it is a much longer development process.

Matlab also has the mex interface for Fortran, C and C++, which allow the compilation of functions to be used natively with Matlab. This interface allows the development of the optimization algorithm in C and CUDA outside of Matlab, whilst compromising no functionality.

libLBFGS

libLBFGS is a c port of the L-BFGS algorithm, which was originally written by Jorge Nocedal in Fortran (Naoaki, 2014). This library was chosen for quick development of the program, as the focus here is on improvements to the current algorithm, not the development of a new algorithm. This library comes with plenty of functionality as standard that is not necessary, and it will need to be removed to simplify the source code.

C

C has been chosen due to the interface with the Matlab mex compiler, it also allows the use of CUDA for the parallelization of the line search. C also requires explicit memory control, which will be necessary for this application when dealing with large matrix that will need to be cleared from memory quickly. C was also chosen due to past experience with the language.

CUDA

CUDA was chosen for the parallelization of this project, as it is built around the single-instruction-multiple-data (SIMD) paradigm. CUDA kernels have many lightweight threads that each do small operations on data, in the case of the line-search this could be function evaluations. CUDA is a highly scalable language, however it is relatively underdeveloped and can be tricky to get kernels working effectively for complex problems.

GitHub

GitHub was chosen as a version tracker for this project, it also doubles as a project logbook for milestones. This tool will aid in the final report writing, along with bug and progress tracking.

Hardware

For this project, it was necessary to build a computer that could run the optimizations for long periods. There was careful consideration of the specifications of

the machine, to be aimed at solving this particular problem. The following is the final spesifications:

- Xeon X5560 Processor 4 Core 2.8GHz 8MB Cache
- 16GB DDR3 10600R Ram
- NVIDIA GTX 760 2GB

The NVidia GTX 760 was chosen as it shares the same Kepler architecture as the Tesla scientific computing cards (K10,20,40,80) . It was chosen over the GTX 750 due to the much higher memory bandwidth and size. Registered DDR memory was chosen with the intention of more being added if necessary to ensure the system remained stable as more memory was added. The Xeon processor was chosen as it is designed to run for long periods of time at a high load.

Optimization selection for this project

L-BFGS has been chosen as the optimization algorithm for this project, firstly as the gradient is available in this computation, which allows the calculation of a descent direction; which implies a global optimum. Secondly, it has been proven to be efficient at solving quantum control problems when compared to other methods (S. Machnes, 2011).

L-BFGS also has potential to be parallelized, despite it being naturally an iterative process. The line search portion of the algorithm is serial as standard, however, there is potential to speed up performance. This has been shown to bring an improvement in the past, where a parallel L-BFGS-B algorithm was implemented on GPUs (Fei, Rong, Wang, & Wang, 2014).

Finally, due to the low memory requirement of the L-BFGS method, it is favored over the standard BFGS algorithm. This is due to the complexity of the problem, where storing the approximation to the Hessian matrix maybe too large to fit into system memory once the problem size increases.

Pre-existing work

There has been much research into information transfer in spin networks, however the majority of previous research has been the finding optimal controls on different network topologies (Jonckheere, Langbein, & Schirmer, 2014) (Christandl, Datta, & Andrew J. Landahl, 2004) (Schirmer & Langbein, 2014) (Cui & Mintert, 2014). There does not appear to have been any research into parallelization of these techniques when applied to a quantum control problem.

Line-search Methods

To understand how a partly parallel L-BFGS algorithm could be implemented, it is first important to understand the role of the line search component of the algorithm. L-BFGS can use any line search algorithm, as it does not require an exact minimum to be found in a given direction.

In general, line search algorithms aim to find a suitable step size α that minimizes the objective function shown below. In this equation, p represents the direction provided by the optimization algorithm for the variable x. Line search methods are focused on searching a one dimensional search space from 0 to α, where alpha is a non-negative real number.

$$\phi(\alpha) = f(\alpha p_k + x_k)$$

Line search methods are used in optimization algorithms to provide a step size towards the global optimum. These methods are provided with initial values x and a direction that is calculated by the optimization algorithm. The line search will return a step size α that minimizes or maximizes the variables x to optimize based on the input direction p. From this stage, the optimization algorithm will calculate a new direction to search and pass these values back to the line search. A line search will contain a set of termination conditions, these depend on the individual algorithm, but usually it is when a sufficient decrease is found. This process repeats until the optimization algorithm hits a termination condition.

Line search algorithms are often lightweight and take up the minimum amount of computing time, to find a rough estimate for the step size. This is because it is thought to be more efficient to spend more computing time in finding the best descent direction in a complex problem. However, in this application, due to the extremely complex landscape it may be beneficial to devote more time to an expensive line search algorithm.

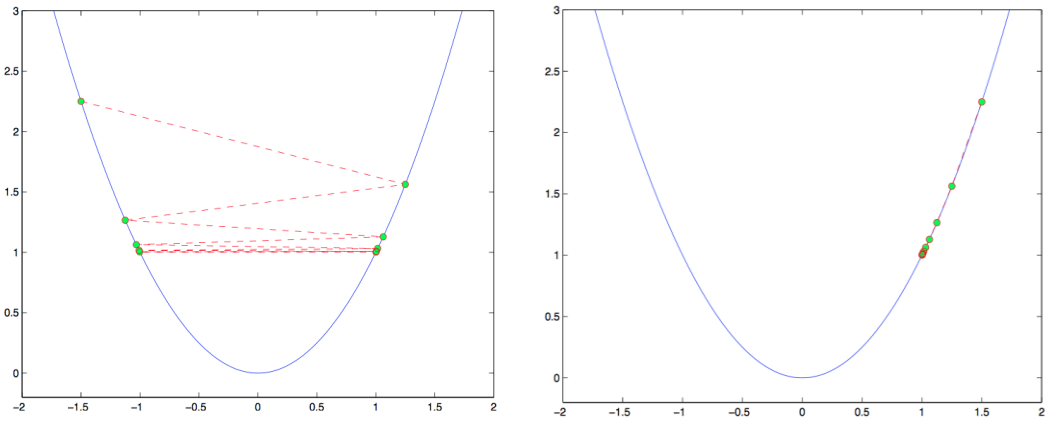


FIGURE 5 (HAUSER, 2007)

The figure 5 displays the issue with poorly conditioned line searches where the minimum step size is too large or too small. In both instances, the minimum is not reached as the line search is restricted by step sizes. As displayed above, it is important to take into consideration line search parameters and termination conditions.

Direct Line-search

Direct line search methods aim to compute the exact minimum of the line. The basic idea is that the minimum is found between a set of the brackets. These algorithms will recursively add and remove new brackets around the minimum until the minimum found is within tolerance.

Backtracking Line-search

The backtracking line search method is based around the Armijo–Goldstein condition, where a relatively large initial step is taken, and is iteratively reduced by stepping forwards and backwards until a minimum is found that sufficiently small. The Armijo–Goldstein condition for an adequate reduction for the given step size in relation to the objective function.

$$f(x + \alpha p) \leq f(x) + \alpha c m$$

If the above condition is met, then the line search will return the step size α. In the above equation p represents the search direction, m is the local slope of the equation and c is the control parameter for the algorithm with a value of $c \in (0,1)$.

Parallelization

There are two key areas that may provide a large improvement in performance, the line search component of the L-BFGS algorithm and the quantum target function. This section will discuss the benefits to parallelization of these sections.

Line search

Current line search methods are focused on providing a very rough estimate on the correct step-size given the direction that will optimize the variables. The key concept is that an optimization algorithm should spend substantially more time computing the correct direction, rather than the step-size. Because of these factors, line search algorithms are naturally serial and could be improved by providing a more accurate step-size in the same time through parallelization.

The line search in this paper is based around evaluating a line with regular intervals in parallel, providing a more accurate description of the landscape. With each evaluation loop, the line may be searched at up to 1000 regular intervals. The minimum of these values is then taken as the optimum step size. The aim here is to create a line search algorithm that will allow L-BFGS to reach a termination condition in less iterations, as it should improve the accuracy of the search.

Quantum target function

For calculation of probability given initial system controls is very expensive, this is due to the large number of matrix operations that are necessary to compute the final result. The issue with parallelizing the quantum target function is that calculating a system state requires the previous system state, so it is an inherently serial process. However, there are improvements that can be made in the way that each system state is calculated. This can come from careful selection of tools and methods, reducing the run time required.

In an ideal situation, where more time is available to develop the system, it may be beneficial to implement these matrix calculations on a GPU, where the architecture is suited to matrix arithmetic. However, for these methods to be efficient on the GPU, time taken for memory transfer from the CPU to GPU must be taken into consideration. So, if implemented there must be a minimum amount of data transferred between devices; such as single control variables rather than large $n * n$ matrices describing system states.

In this paper, the focus will be on creating the most efficient version of these methods with the tools provided by Matlab and the parallelization toolbox.

Algorithm Design

Algorithm design in this project was based around the best performance improvement that could be obtained in a short period of time. There are two distinct separate areas of the problem that can be improved; the optimization algorithm and improving the quantum target function.

Quantum target function

Due to time constraints and for simplicity, Matlab was chosen to model the quantum target function. This was due to the relatively lengthy time needed to develop a program that performed matrix functions with imaginary numbers in C. Due to the Matlab/Mex interface, it allows C to call Matlab functions and visa-versa. This interface allows for rapid development

The purpose of this algorithm is to provide a probability of transfer from A to B given the initial system controls. During the optimization the evaluation function is called thousands of times; when using the parallel line search technique. By improving this function slightly it can lead to a large reduction in runtime. The aim here was to reduce the number of evaluations performed by the Matlab code to reduce the runtime as much as possible.

Algorithm outline

EVALUATE

```
function [ fx, g ] = evaluate(time, x, params){

    fx = fx_eval(time, x, params);

    g = evaluate_gradient(time, x, params);

}
```

EVALUATE_GRADIENT

```
function [ g ] = evaluate_gradient(time, x, params){

    h = 0.000001;

    fx = fx_eval(time, x, params);

    for i = 1 : numel(x){

      xtemp = x;

      xtemp(i) = x(i) + (h/2);

      forward_diff = fx_eval(time, xtemp, params);

      xtemp(i) = x(i) – (h/2);

      g(i) = (forward_diff - fx_eval(time, xtemp, params))/h;

    }

}
```

FX_EVAL

```
function [ fx ] = fx_eval (time, x, params){
        t = round(time(1));
        delta = time(2);

        input(params(1)) = 1;
        target(params(2)) = 1;
        nSpins = params(3);

        for n = 1 : time_steps{
                H_O{n} = H0;
                for ii = 1 : nSpins{
                        H_O{n} = H_O{n} + (x (ii) * H{ii});
                }
        }

        //Calculation of U matrix
        for n = 1 : time_steps {
                U {n} =  expm(-1i*1*H{n}*delta);
        }

        //Calculate the final propagator
        total = U{1};
        for n = 2 : numel(U){
                total =  (U{n}*total);
        }

        probability = target * total * input;

        //Calculate the infidelity of the system to minimize
        fx = 1-abs(probability)^2;
}
```

Algorithm decisions

GRADIENT CALCULATION

Due to the high cost of calculating an exact gradient, in this evaluation function the gradient is calculated by finite difference. In particular it is calculated by the symmetric difference, which attempts to draw a secant line that intersects $f(x)$ that can be used to represent the gradient

$$\frac{f\left(x+\frac{h}{2}\right)-f(x-\frac{h}{2})}{h}$$

Due to the complex landscape of the quantum target function, there had to be careful consideration of the size of *h*. If *h* is too large then the gradient will be inaccurate, and if *h* is too small the gradient will be close to 0; suggesting an optima. For testing, an algorithm was setup to choose a value of *h* so that the secant line intersects $f(x)$. The algorithm recursively reduced the value of *h* until it was within tolerance. The optimal value of *h* for this application is: 0.000001.

Algorithm outline

```
function [ g ] = evaluate_gradient(time, x, params){
    ....
    if h > 0.00000001{
        for i = 1 : numel(x){
            if abs(fx - g(i)) > h{
                h = h/10;
                g = evaluate_gradient(time, x, h);
                return
            }
        }
    }
}
```

VECTORISATION OF MATRIX CALCULATIONS

Matlab provides tools with optimal performance for matrix and vector operations, called vectorisation tools. They supposedly allow improved performance when performing the same operations on different data sets. During design, it was important to take these into consideration, as the target function has many matrix calculations. However, when testing it was found that performance was significantly worse.

The setup here was calculating the system propagator U for a ring of size 13, where the target was transfer from spin 1 to spin 7 and all control amplitudes were set to 1. The test was run 100 times and taken an average time from all runs.

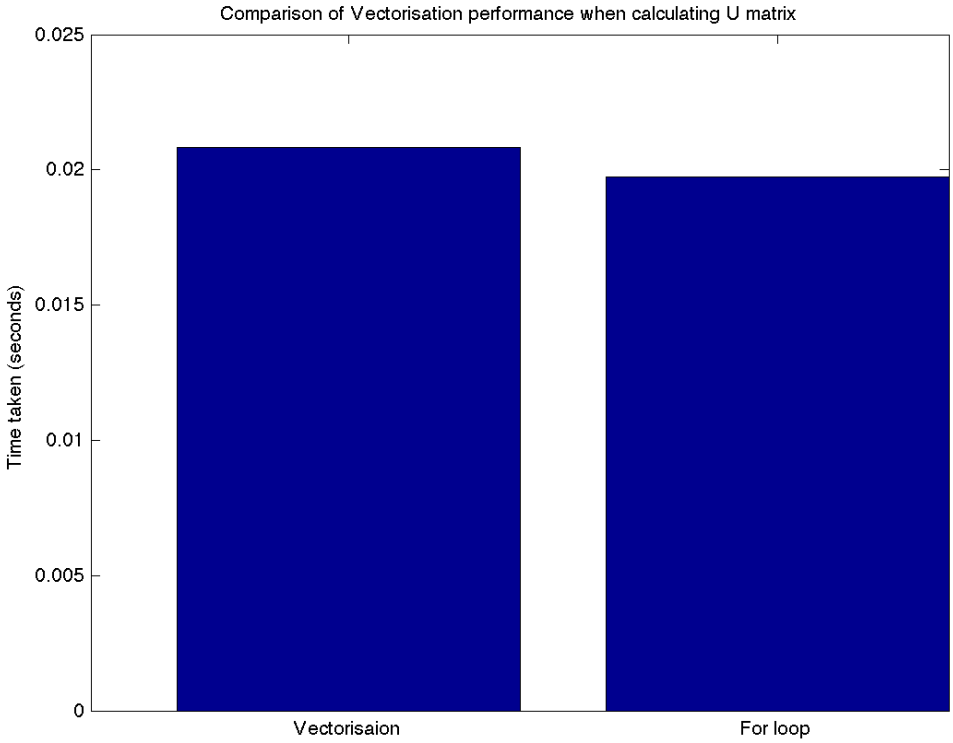


FIGURE 6

Figure 6 represents the difference in timing between the two methods. The results showed that the vectorisaiton approach is 5.46% slower than a standard for loop. Similar performance was found throughout the program, and for this reason Matlabs vectorisation tools were mostly avoided.

Parallel Line-search

One area that can be improved for parallelization in this project is the line search component of the L-BFGS optimization algorithm. Normally, this process is an iterative lightweight process that attempts to provide a rough step to optimize the values given. However, in this application, the line search algorithm was based around the single-instruction-multiple-data (SIMD) style of programming that is utilized by NVidias CUDA. This style of implementation means that this line search can conduct many evaluations in a given search direction at one time. By sampling a 2D search direction many times in one function call it allows for a better understanding of the landscape, and can improve the probability that optima are not missed.

Inspiration for this line search came from a paper (Fei, Rong, Wang, & Wang, 2014) that is based on parallelizing the L-BFGS-B (L-BFGS with box constraints). Where in the paper they used reduction on the GPU to find the minimum values for the step size α.

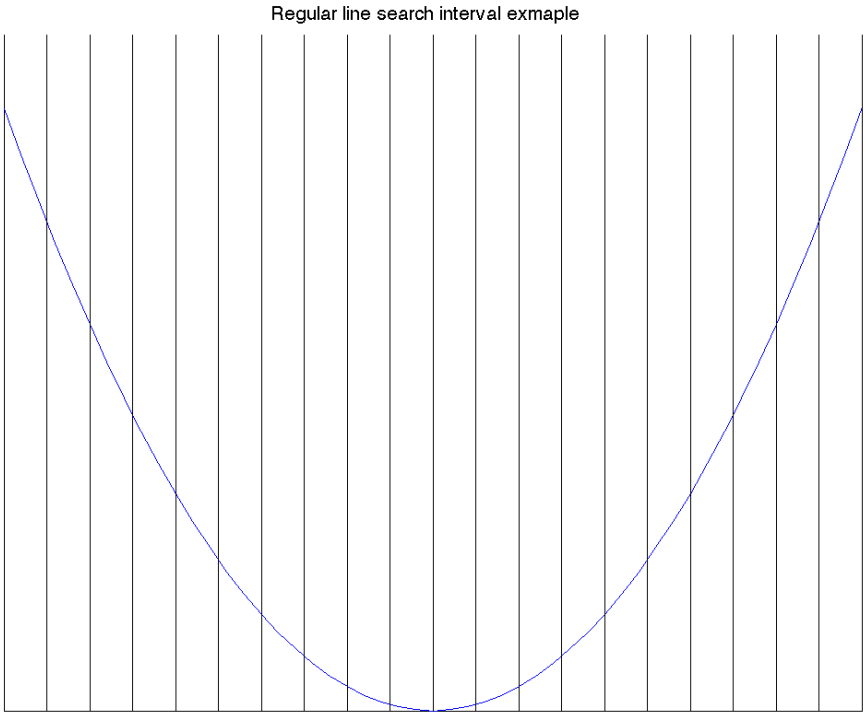


FIGURE 7 AN EXAMPLE OF A REGULAR INTERVAL SEARCH

In figure 7 the blue line represents the unknown line that is being searched, the black lines represent the regular interval sampling conducted by the line search.

With the understanding that the landscape of the quantum control problem is very complex with many local extremes, the aim here is to create an accurate line search, which would lead to an improvement in finding minimums. In theory, by providing the optimization algorithm with a more accurate line search, it should reduce the number of iterations to find a minimum, which further reduces the runtime.

The basic outline of the program is to split the search area into regular intervals, and find the minimum of all points. The algorithm will then recurse, checking to the left and the right of the current minimum. This continues until a termination condition is reached.

There is, however an issue with using CUDA based line search, where the function evaluations have to be conducted on the GPU; where in this case they are conducted through Matlab on the CPU. Because of this, I have presented a proof of concept with test functions later on, and developed a CPU parallel line search. The CPU parallel line search utilizes Matlabs parallelization toolbox, which allows the creation of a local parallel pool, allowing multiple threads to run simultaneously. In this case, the evaulations are all run in parallel as they would if they were in a CUDA kernel.

Algorithm outline

```
cudaLinesearch(int n, double x, double fx, double direction, double step){
    for i = 0 : 5{
        min_fx = +inf; min_thread = -1;
        local_step = current_step +((step/number_threads) * thread_id);
        for i = 0 : n
            x_local[i] = x[i] + (step * direction);
        for i = 0 : n
            fx_local += fx_eval(x);
        fx_values[thread_id] = fx_local;
        __syncthreads();
        for i = 0 : number_threads{
            if(fx_values[i] < min_fx){
                min_fx = fx_values[i];
                min_thread = i;
            }
        }
        __syncthreads();
        if(thread_id == min_thread){
            gradient = eval_gradient(x_local);
      if(thread_id > 0){
        current_step = thread_step - stepsize;
      }else{
        current_step = thread_step;
      }
            current_step = local_step;
            x = x_local;
            fx = fx_local;
        }
    __syncthreads();
    stepsize = (stepsize / num_threads) * 2;
```

```
        }
        return(fx, x, gradient);
}
```

Line search termination conditions

Maximum number of iterations

Step size is smaller than 1-e20

Optimizing spin network controls

Due to time constraints in this project, implementing a CUDA version of this problem is not feasible. This is due to the large setup time required to develop the quantum target function in C. For the algorithm to run solely on the GPU, the quantum target function must be calculated on the GPU, where currently it is modeled in Matlab on the CPU.

Because of these constraints, there has been a different approach to providing a proof of concept. The performance expected in terms of wall time should be similar, if not worse when utilizing the parallel line search method. However, in contrast the number of iterations performed should be less, suggesting that in the future, the algorithm could benefit from a more efficient parallel implementation.

To implement a parallel version of this line search technique, this implementation utilizes Matlabs parallel toolbox, where many local threads can be run simultaneously. This application loosely mimics the CUDA algorithm, where more than one line search evaluation can be conducted at one time. Although, it is important to note that this implementation used 8 CPU threads, rather than thousands of GPU threads.

When designing the test fucntions for the optimisaiton algorithm, they were utilized CUDA kernels for the parallel line search. The largest setback was implementing the CUDA compiler with the Matlab mex compiler, which allows CUDA files to be compiled and called from Matlab. This issue was due to an incompatibility with mex, CUDA, Windows 8 and Visual Studio versions. Installing Visual Studio (VS) 2010 & 2012, and then creating a custom install script to remove dependencies on VS 2008 rectified the issue. This issue was particularly frustrating, as there was no indication at the time of what was failing.

For the quantum target function, it was important to decide between a Matlab GPU or CPU implementation of the parallel line search. The issue is that only a small subset of commands are available to be run on GPUs via Matlab (MathWorks, 2015). Here, there is no suitable matrix exponential function that is required by the quantum target function. This is due to a specialized log2 function, which is not available to be run on GPUs through Matlab at the current time. Despite this, an attempt was made to run a GPU version of the implementation. Figure 8 represents the runtime achieved with the two different parallelization methods.

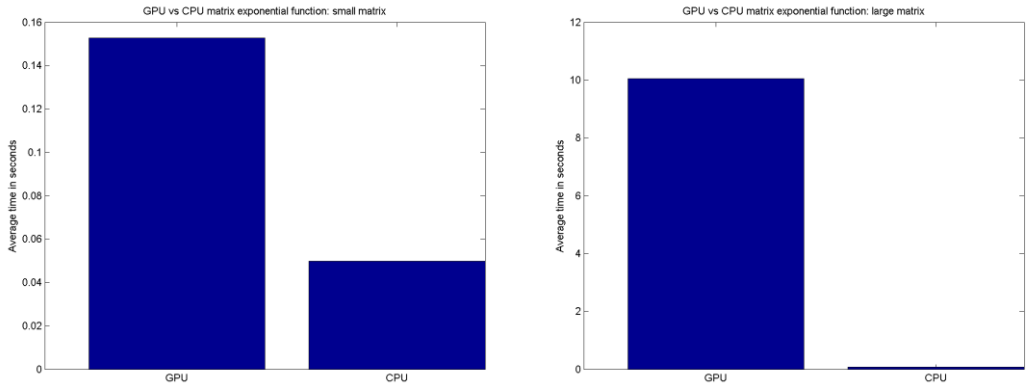


FIGURE 8

The timing results were collected for two separate matrix sizes: $10 * 10$ and $100 * 100$. Each matrix was filled with a random numbers between 0 & 1 in each matrix location. From there, each matrix was evaluated 10 times inside a '*parfor*' loop, which initializes Matlabs parallel execution of parallel loops, either on the CPU or GPU. The aim here was to test to see if the Matlab parallel implementation on the GPU would be able to run more threads simultaneously, however, the results suggest that it does not.

This GPU implementation is a hybrid, where the log2 function is performed on the CPU, and the value is passed back to the GPU. It is clear to see from the timings that this implementation is not efficient, and for this reason it was not used in the project.

Controlling spin networks

Introduction

WHAT IS THE GOAL?

The control of spin networks is a current research area where practical applications could lead to long distance data exchange on quantum spin networks. The aim of quantum control is to maximize the probability of successful data exchange. However due to the natural unpredictability of quantum mechanics, this is a complex task.

This application focuses on a closed network where only one bit of information is in the network at one given time. This system is designed to optimize initial controls, where the data will be collected from the system at a pre-determined time. The controls in this system are arbitrary numbers that represent the strength of an electromagnetic field that is individually placed on each spin. The electromagnetic fields alter the spin of each network node by aligning the spin with the magnetic field.

In this particular implementation, the controls are set for the duration of the system run time, and then a probability of transfer calculated at the prescribed time. For example: *Given time t and the target to get information from spin 1 to spin 4, what controls would maximize the probability?* This means that for every different configuration of the system controls, the evolution of the system needs to be re-calculated which is the largest computational cost.

System design

The system is modeled around a perfect spin chain, where it is completely isolated from the environment and no data is lost. The representation of the system is close to a bilinear control problem, which is based on the following formula (Pardalos & Yatsenko, 2008).

$$\dot{x}(t) = \left(A_0 + \sum_{i=1}^{m} u_i(t)A_i\right)x$$

Where x is a state vector, A is a constant $p * p$ matrix, and $u(t)$ is a restricted measurable control. However, this equation state needs to be modified to fit a quantum control problem.

$$|\dot{\psi}(t)> = -i\left(H_d + \sum_{j=1}^{m} u_j\,(t)H_j\right)|\psi(t)>$$

The equation above represents a bilinear control system in terms of the controlled Schrodinger equations. Here, the closed quantum system is described by the drift Hamiltonian H_d, H_j represents the internal manipulations of the system (control Hamiltonians), where each control can be manipulated by its control amplitude u_j. Control amplitudes here are time independent and are constant. To model the imaginary part of quantum mechanics, i represents$\sqrt{-1}$.

MEASURING PROBABILITY

At each stage in time it is possible, given an input column vector (ψ_k) and a target state (p_e) in the form of a row vector to calculate the maximum probability. The probability is a degree of possible success, when transferring data from the input to the target spin. The probability is calculated in the following way.

$$p_{lk}^{(t)} = |<\psi_k|U_{(t)}|p_e>|^2 = |\psi_e^{(t)}|^2$$

This probability measurement is the basis of the optimization, where the target is to obtain the minimum value. However, as this equation outputs the fidelity of the transfer, it needs to be converted to an infidelity, as to work with the minimization algorithm. This is done in the following fashion:

$$\min_x |1 - p_{lk}^{(t)}|^2$$

Construction of the drift Hamiltonian

The system Hamiltonian H_d is a $n * n$ matrix, where n is the number of spins in the chain. This matrix represents the topology of the network, where the ring is constructed from XX nearest-neighbor coupling, which is derived from the Pauli matrices:

$$\sigma_1 = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma_2 = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\sigma_3 = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

This coupling is discussed further in '*Information Transfer Fidelity in Networks of Spins*' (Jonckheere, Langbein, & Schirmer, 2014). It is not discussed any further in this paper as it is out of scope. For simplicity the system is based around a single excitation subspace, were only one bit of information will be added in the system at one time. The following matrix represents the constructed drift Hamiltonian, where any location with a '1' represents a possible state described by the wave function.

$$H_d(N) = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & 0 & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

System controls

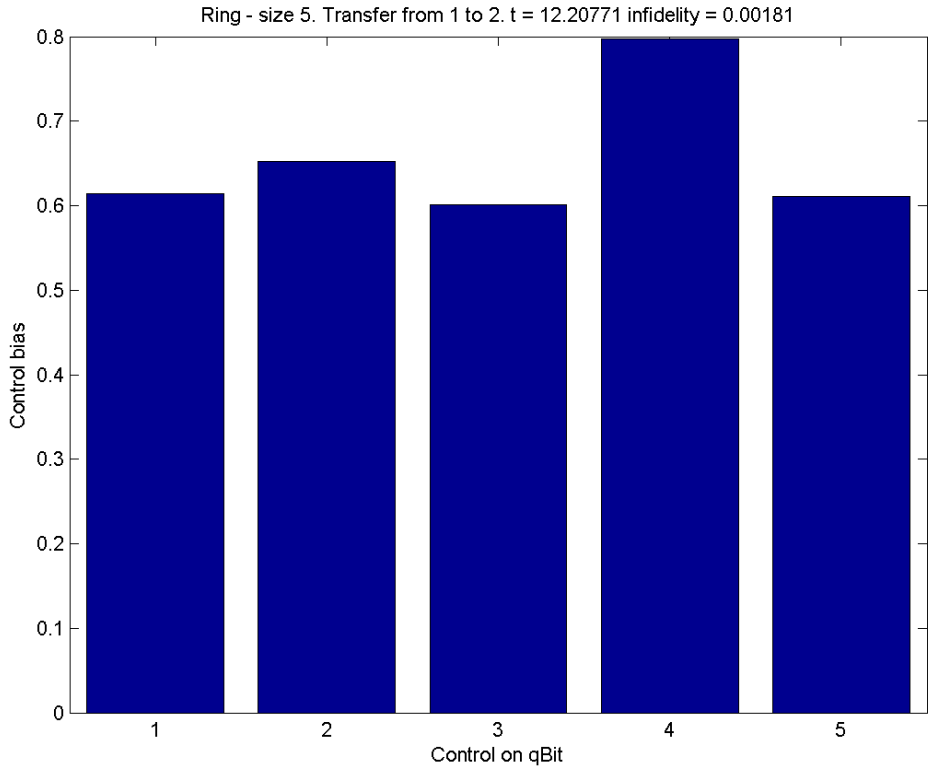
For the system controls, there is additional control Hamiltonians required to represent the placement of the electromagnetic field in the system. These electromagnetic fields are placed to have a direct effect on one spin in the ring only. There are n matrices; each matrix is $n * n$, where n is the number of spins in the system. Each matrix has a 1 in the $H[j, j]$ position, where j represents the number of the matrix, the rest of matrix is 0s, there are n matrices in total. The examples below are for matrices of any given size.

$$H_1(N) = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & 0 & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

$$H_n(N) = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & 0 & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

INITIAL CONTROLS

From early research, it is possible to maximize transfer fidelity by providing initial controls to the system that place a strong control amplitude on the middle node between the transfers in a ring of odd numbers. This initial control bias helps steer the optimization process towards finding a high fidelity faster.



These controls work as it essentially turns the topology of the ring into a chain, where nodes are linked in the following fashion (figure 9 & 10). The solid lines represent the connections between the network nodes, however data naturally flows along the dotted lines. Figure 10 displays the transfer properties in standard spin chains, as described by (E. Jonckheere, 2014).

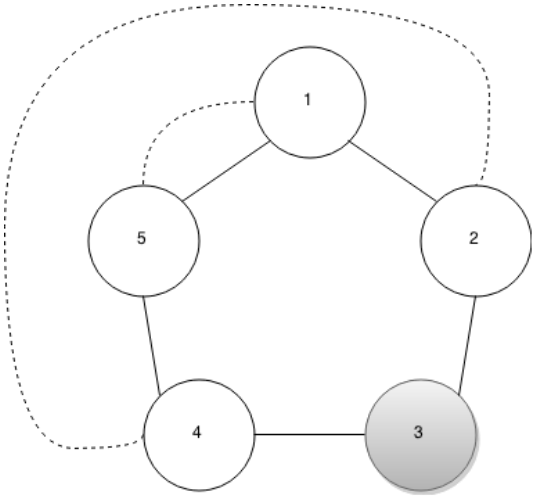


FIGURE 9 CONTROLS CAN CREATE CHAIN LIKE BEHAVIOUR

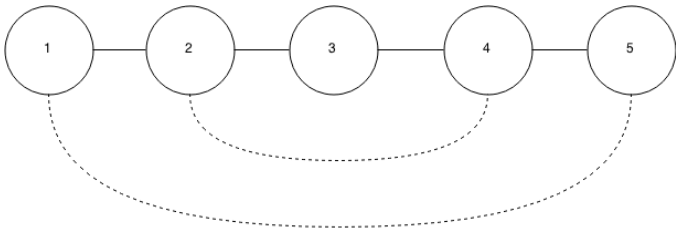


FIGURE 10 NORMAL CHAIN TRANSFER BEHAVIOUR

Due to these properties of spin chains, it is desirable to attempt to steer the optimization so the ring mimics this behavior. The initial system controls will be set to the following:

- Middle transfer spin is assigned a random value from 5 to 15; e.g. spin 3 on a transfer from spin 1 to 5.
- All other spins are given a control value from 0 to 1

Manipulations of the system

Calculating the natural evolution of the system is an iterative process that relies on the previous state. Each state is separated by a time difference Δ that is fixed for all iterations. In this instance Δ has the value of 10, which is an arbitrary time unit. The equation below represents the previously mentioned Schrödinger equation modeled as a control problem; from this equation it is possible to calculate the value of the wave function ψ for every time t.

$$|\dot{\psi}(t)> = -i\left(H_d + \sum_{j=1}^{m} u_j(t)H_j\right)|\psi(t)>$$

State Transfer

State transfer is calculated as a function of time, given the initial values. This is concurrent with the Schrödinger equation, where the evolution of a quantum system can be calculated from the initial values. Here it is necessary to calculate the system propergator 'U'. This is calculated from the system Hamiltonian. In the following equations, all of these conditions apply:

- i represents the imaginary unit $\sqrt{-1}$,
- $\hbar$ represents the reduced Planck constant which is equal to $\hbar = \frac{h}{2\pi}$. Where in instance, to simplify the problem, $\hbar$ is equal to 1.
- Every matrix is $n * n$ in size, where n is the number of nodes in the network

$$H(t) = H_d + \sum_{i=1}^{n} H_i f_i$$

The equation above represents the calculation of the system Hamiltonian for every time t, the Hamiltonian represents the total energy of the system. It contains a full possible set of outcomes when a measurement is made of the system. In the above equation, H_d is the system drift Hamiltonian, H_i refers to each control matrix and f_i refers to the control amplitude for the given control matrix. The system Hamiltonian represents the state of the system in the form of a matrix.

Once the calculation of the Hamiltonian has been done, it is possible to calculate the value of U, the system propagator that describes the transfer of the system between states. Calculating the value of U applies the system Hamiltonian to this specific quantum target function, this is the point when the quantum mechanical behavior is introduced.

$$U^n = expm(-i\hbar H^n \Delta t)$$

In the above equation, the value for U is calculated for every system time step (n), the values have been swapped, as t is used for explaining Δt; which represents the time difference between each time step.

$$U = \sum_{i=1}^{n} U^n + U^{n-1} + \cdots + U^0$$

Once the final value of U has been calculated, it is possible to calculate the final value of the wave function ψ by multiplying the initial system state with the system propagator.

$$|\psi(t)> = U|\psi(0)>$$

Optimal Targets

For this application, the aim is to maximize transfer probability given an input and an output node in a network. However, due to the inherent instability and unpredictability of quantum mechanics this makes the task much more complex than it is in traditional computing.

In this project there are two aims for a transfer; calculate the best transfer probability possible and calculate the shortest time where a transfer has a probability of 99.98% or more.

Method

TESTING

In order to test the effectiveness of the implementation, there is a set of optimization functions with known minimums. The sphere and Rosenbrock function have been http://www.cardiff.ac.uk/insrv/libraries/trevithick/index.html#opening for testing as these represent both a simple and complex function to minimize.

System setup

- Number of dimensions from 10 to 1000 in increments of 10 to provide a large range of results
- Initial values are random values between -5 and 5

These test functions will display if the algorithm reaches the known minimums, and will provide a good benchmark of the effectiveness of each algorithm.

BACKTRACKING LINE SEARCH

The sequential line search will only be conducted on one ring size for one transfer simply to compare the performance of both line searches. This is due to time constrains, as the sequential line search takes considerably longer to perform 2500 searches (100 random times, 25 optimizations per time). The results from this transfer will be used to compare performance between the two line search algorithms to display their behavior when applied to the quantum control problem.

System setup

- Ring of size 5
- Only one qubit of data in the system at one time
- Transfer from spin 1 to spin 2
- Middle control amplitude is set to 5 – 15
- All other control amplitudes are set from 0 – 1
- Target time a random value from 1 – 100
- Delta set to 10

Backtracking line search termination conditions

- Armijo-Goldstein condition is reached; return step size α
- Step size is less than 1e-20; return error, exit L-BFGS
- Step size is greater than 1e20; return error, exit L-BFGS
- Maximum number of line search iterations conducted; return error, exit L-BFGS

PARALLEL LINE SEARCH

Due to the complexity of the problem, it is important to collect many results of a variety of initial conditions. The following are the conditions for result collection when using the parallel line search.

System setup

- Rings of size 5, 9, 13
- Only one qubit of data in the system at one time
- Transfer from 1 to ring size in increments of 1, except 13 where transfers were incremented in 3
- Middle control amplitude is set to 5 – 15
- All other control amplitudes are set from 0 – 1
- Target time a random value from 1 – 100
- Delta set to 10

The initial bias controls are derived from earlier research as to how the ring performs when equal bias is set on the spins. These biases are taken as an optimal set of controls that have been used to steer the optimization algorithm.

Termination conditions for the optimization scheme also had to be carefully chosen to reduce system runtime, to allow a large range of collection of results in the time given. The following represent the L-BFGS settings.

L-BFGS termination conditions

- Line search returns a step size of 0
- Maximum number of iterations reached - 50
- Gradient is sufficiently small: $\frac{gradient\ norm}{1} < 1e-5$

Parallel line search termination conditions

- Step size is smaller than 1e-20; returns step size α

This method will provide result sufficient for

Results and Evaluation

Test functions

When testing optimization algorithms, there are sets of test functions available that are designed to test many aspects of optimization algorithms, including: velocity of convergence, precision, robustness and general performance. Two functions have been selected, the sphere function and the Rosenbrock function.

Sphere function

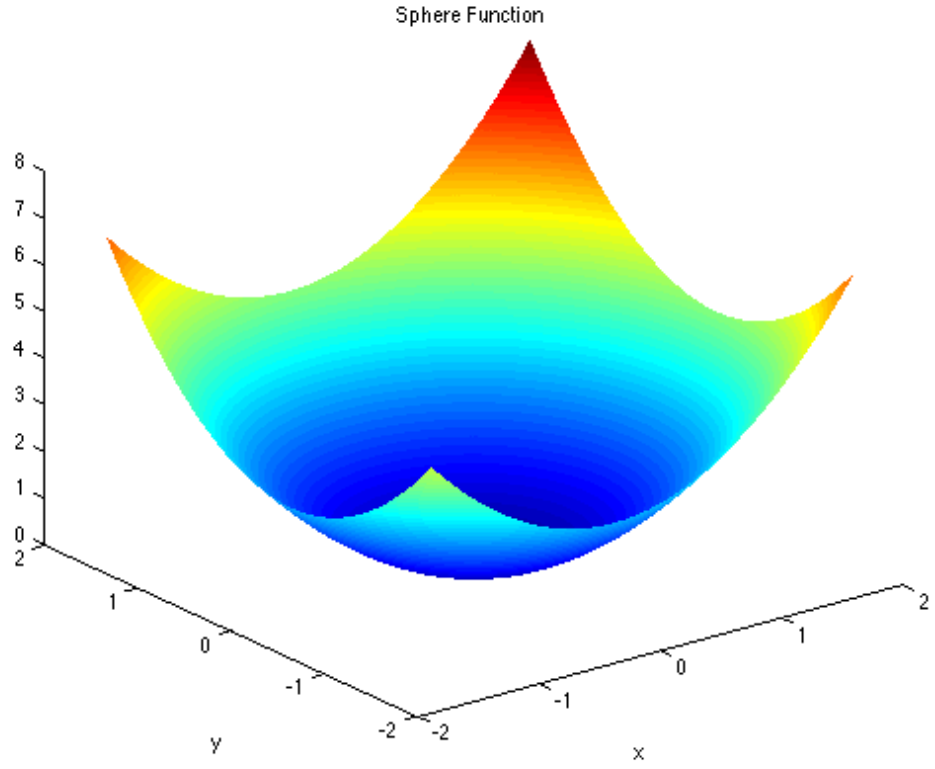


FIGURE 11

$$f(x) = \sum_{i=1}^{n} x_i^2$$

The sphere function is a simplistic test function; it was chosen to test the combined accuracy of the line search and direction calculation. The global minimum is found when all values of *x* are equal to 0. Due to the landscape of the problem, the minimum should be easy to reach with a correctly conditioned optimization algorithm. An accurate line search algorithm will allow the minimum to be found in very few steps.

Rosenbrock Function

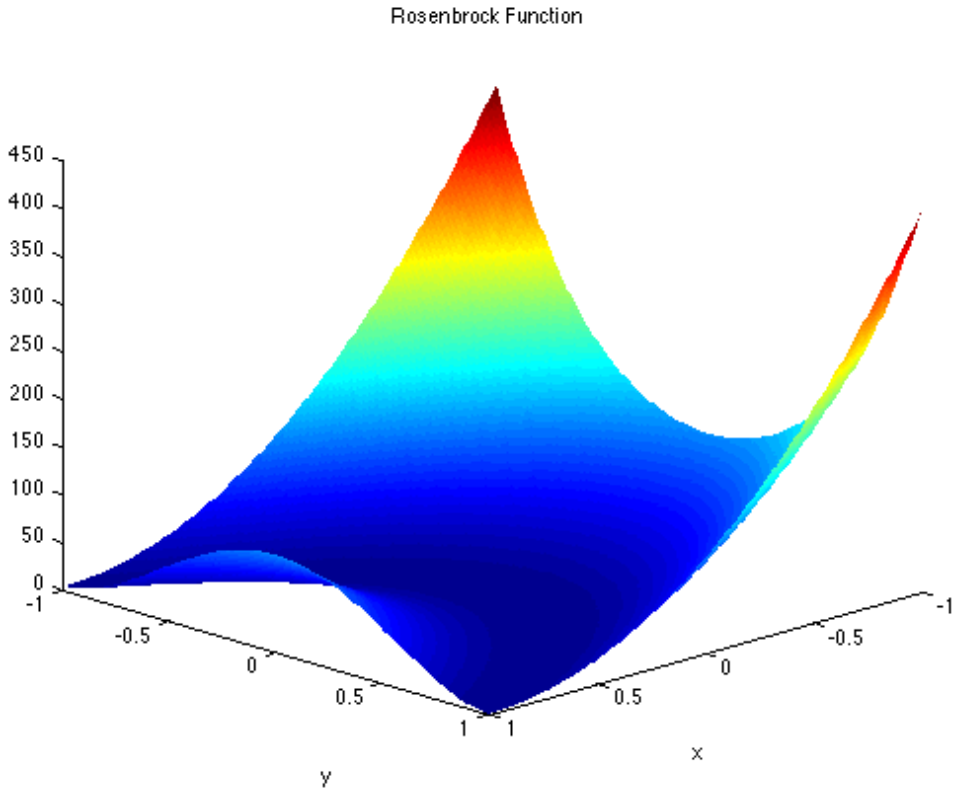


FIGURE 12

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

The Rosenbrock function (figure 12) was chosen due to the relative difficulty of finding the minimum of a problem. The function was first described in The Computer Journal (Rosenbrock, 1960). The function allows the optimization algorithm to find a value close to the minimum relatively quickly; however once close to the global minimal the gradient differences are minute, testing the accuracy of the optimization algorithm. The lowest parts of the valley are virtually flat; this is designed to test the true robustness and accuracy of an optimization algorithm.

Line search performance on test functions

Sphere function

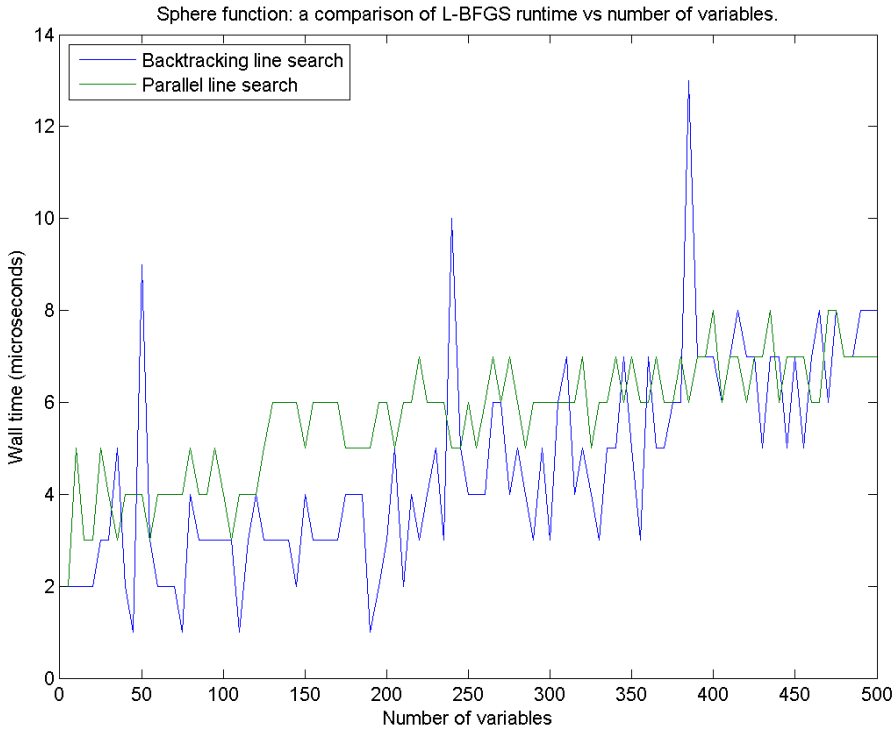


FIGURE 13

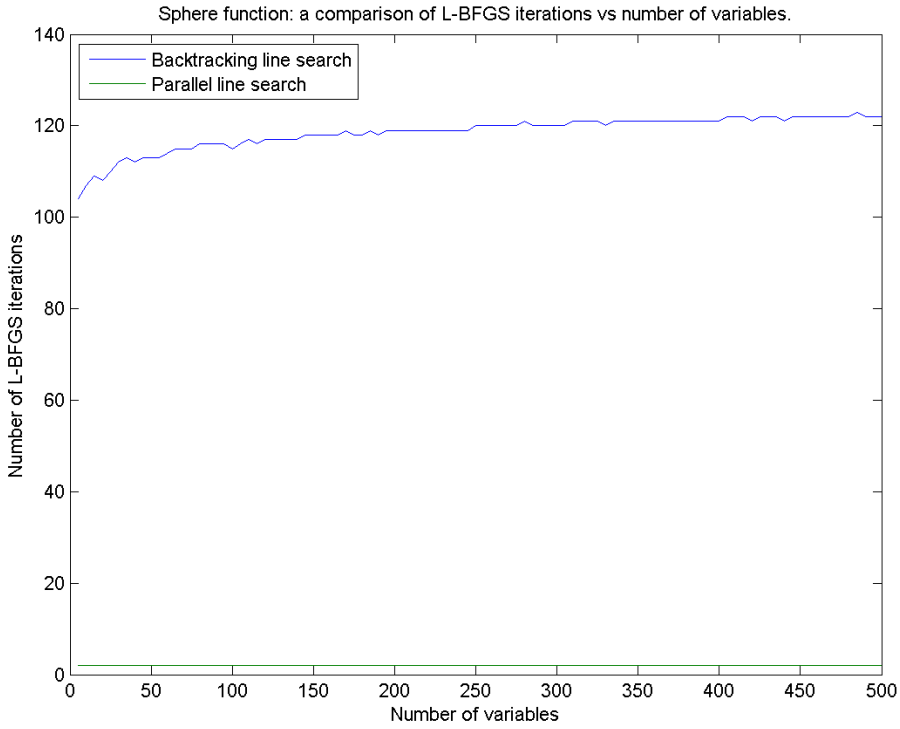


FIGURE 14

Rosenbrock Function

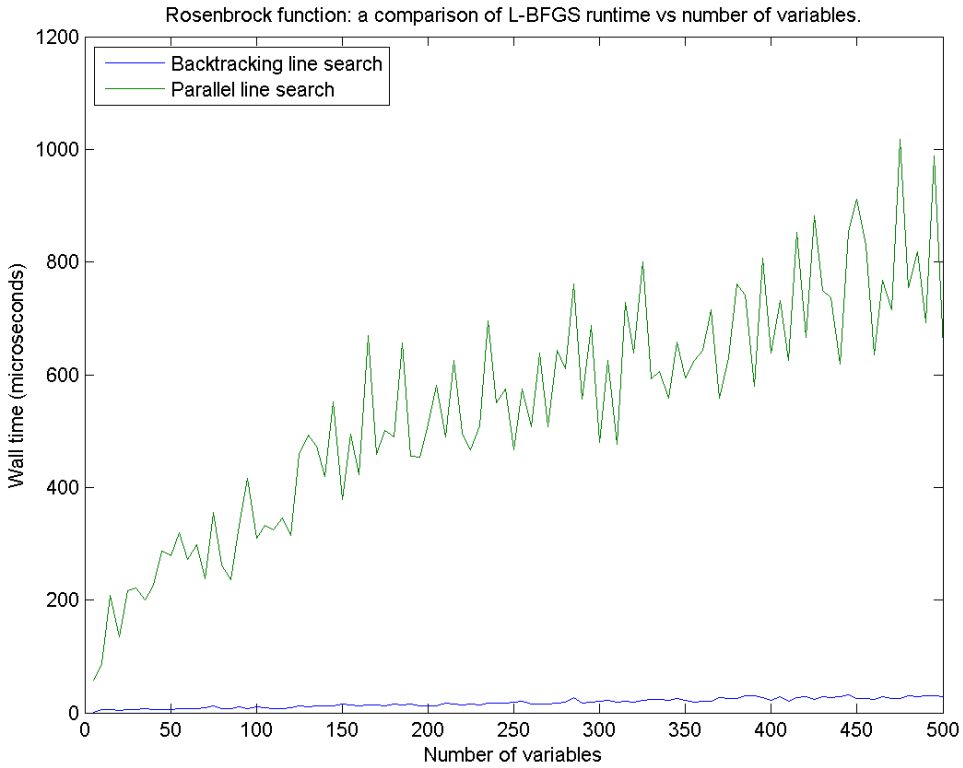


FIGURE 15

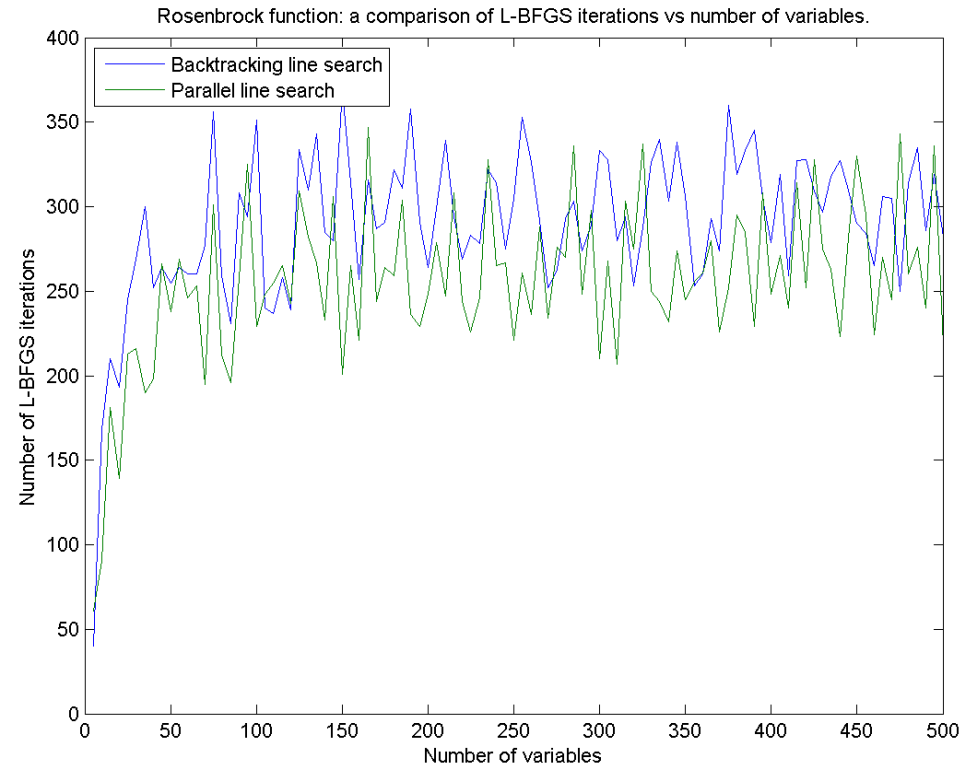


FIGURE 16

Evaluation

From the two test functions it is possible to derive some understanding of how each line search method works given two different environments. There are a few clear performance differences between the parallel and backtracking line search.

Sphere function performance

As previously discussed, the sphere function is a simplistic function, where a correct calculation of direction should lead to the minimum in very few steps, independent of the number of variables. This can be seen clearly when comparing the number of iterations performed by the L-BFGS algorithm, where at each iteration of the algorithm a line search is conducted. If a line search is more accurate, in theory it should mean that less L-BFGS iterations are conducted.

RUNTIME

When comparing run-times, it is clear that the lightweight backtracking line search performs more efficiently when given a smaller number of initial values, however it does not scale well when extra dimension are added. In comparison, the backtracking line search begins to preform better when more dimensions are added. There also appears to be a clear difference in the stability of the algorithms, where backtracking line search has a few spikes in wall time, where the parallel line search does not.

NUMBER OF ITERATIONS

The parallel line search significantly outperforms the backtracking line search here, where it is consistently solving the sphere function in two steps. When applying this line search to the quantum control problem this should significantly reduce runtimes, where function evaluations are extremely expensive. This contrast in behavior would explain the large increase in L-BFGS runtime when using a backtracking line search.

Rosenbrock function performance

The Rosenbrock function is a challenge for many optimization algorithms, as towards the minimum the gradient is minute. This function tests the ability to aid the optimization algorithm to find the minimum quickly, where a poor line search will lead to significantly more optimization iterations.

RUNTIME

For this problem, the parallel line search does not outperform the backtracking line search, and on the surface seems to be unsuitable. The lightweight backtracking line search helps L-BFGS to find the minimum in significantly less time. This graph displays the effect that the evaluation function has on the line search, where it increases the time taken for the line search dramatically.

NUMBER OF ITERATIONS

When comparing the number of L-BFGS iterations, the performance is fairly close, however the parallel line search is marginally better. This difference in the number of iterations performed on a hard problem has potential to be improved to form a faster, parallel implementation.

Conclusion

The aim when developing this line search algorithm is to reduce the number of function evaluations by improving the accuracy. This line search appears to be doing this for complex algorithms where the gradient calculations are difficult and performing

very well when the function is simple. Overall, this line search algorithm has been proved to be more successful than the basic backtracking line search in achieving these goals.

Quantum control

Algorithm Performance

The following results represent a comparison between the final parallel line search and the standard backtracking line search. The results display a transfer from spin 1 to spin 2 in a ring of 5 spins with a random time *t* from 0 to 100. 100 random times were tested, where each time was also tested 25 times.

All results, unless stated otherwise do not include failed runs (where 1 or less steps were taken).

BEST TRANSFER PROBABILITY

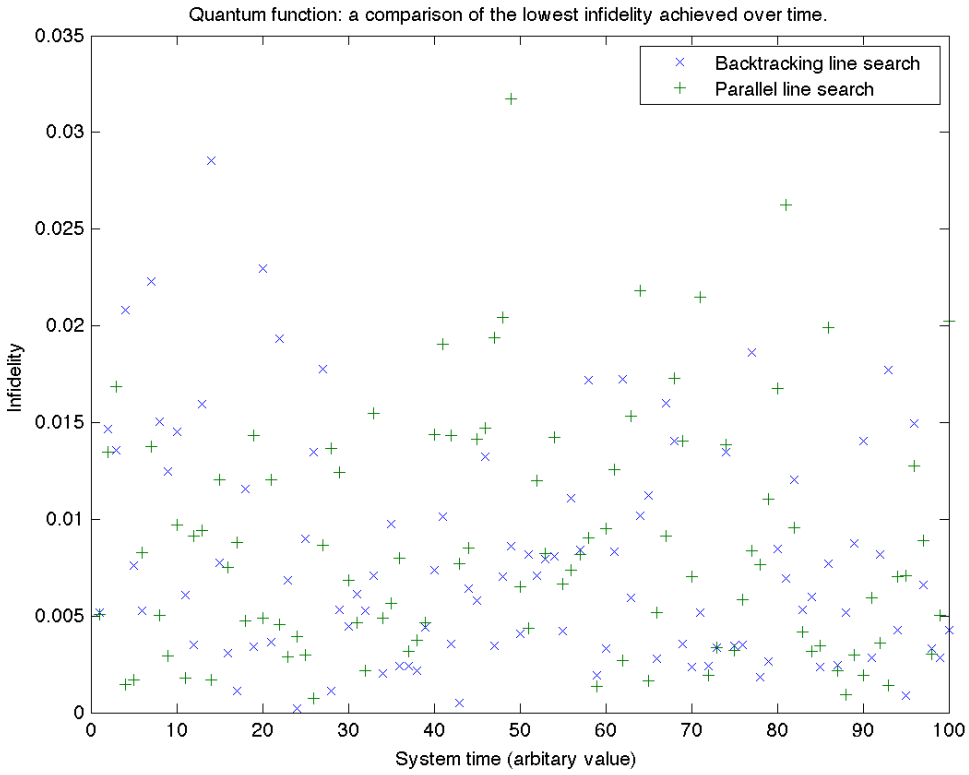


FIGURE 17

Figure 17 represents the best infidelity found given the time *t;* it is the best result of the 25 runs. These results essentially describe how successful the system is, where a lower infidelity is better. The majority of results from both line search techniques fall within similar ranges, however these results are not directly comparable, as the times are not exactly the same. This plot suggests that algorithm performance is similar, as the minimums found are similar. The majority of results shown in this plot would ensure a 99.985% success rate of information transfer between spins, which is a good result for all times.

The range in the results displays the difficulty of the problem, where no optimization ever reaches 0, but where many reach very close. It is also important to note that time does not appear to play a significant role in influencing results, where the optimization algorithm was able to reach an acceptable minimum at the majority of times.

DISTRIBUTION OF BEST TRANSFER PROBABILITY

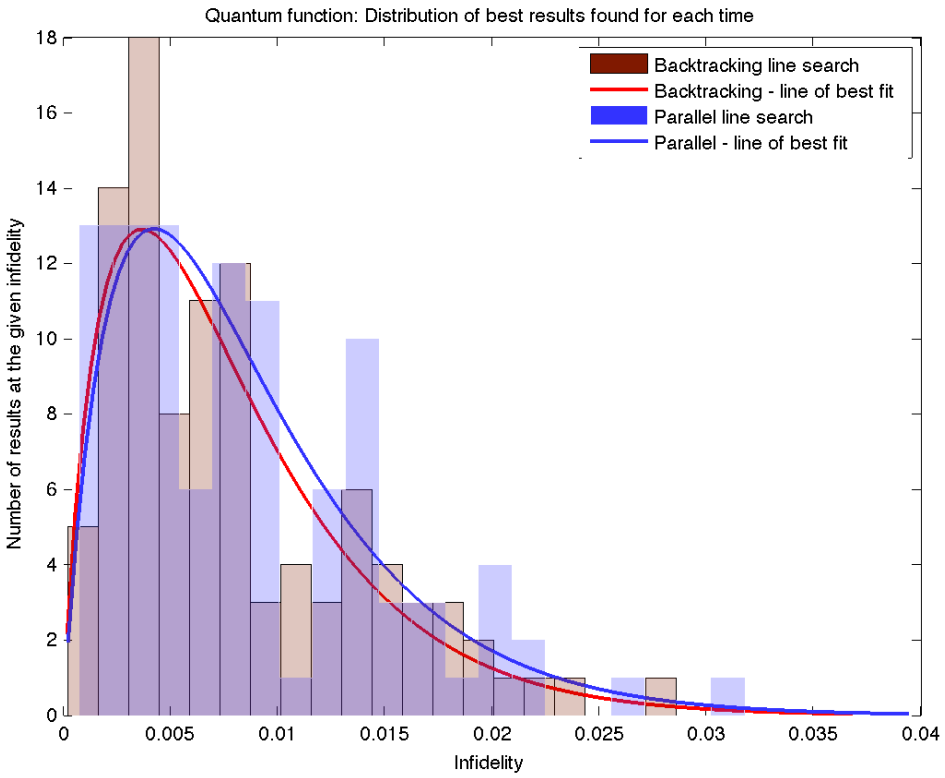


FIGURE 18

To describe the behavior of the system more clearly, figure 18 represents the distribution of the best results. The plot is a histogram of results with a line of best fit for each line search; both plots are laid over each other to allow a direct comparison of results. This plot is intended to show the accuracy and precision of the L-BFGS algorithm given the different line search algorithms.

From the plot, it is to see that the standard backtracking line search provides results where on average the infidelity reached is lower with a narrower range. In contrast, the parallel line search has a larger range of results, suggesting a less precise outcome and a higher infidelity achieved denoting a less accurate algorithm.

This suggests that the backtracking line search is more suited to solving the problem, as it is outperforming the parallel line search when finding the minimum. This may be due to the exit conditions of the parallel line search, where instead of looking for a sufficient decrease of the objective function relative to its slope, it is looking for an exact reduction.

AVERAGE FIDELITY REDUCTION

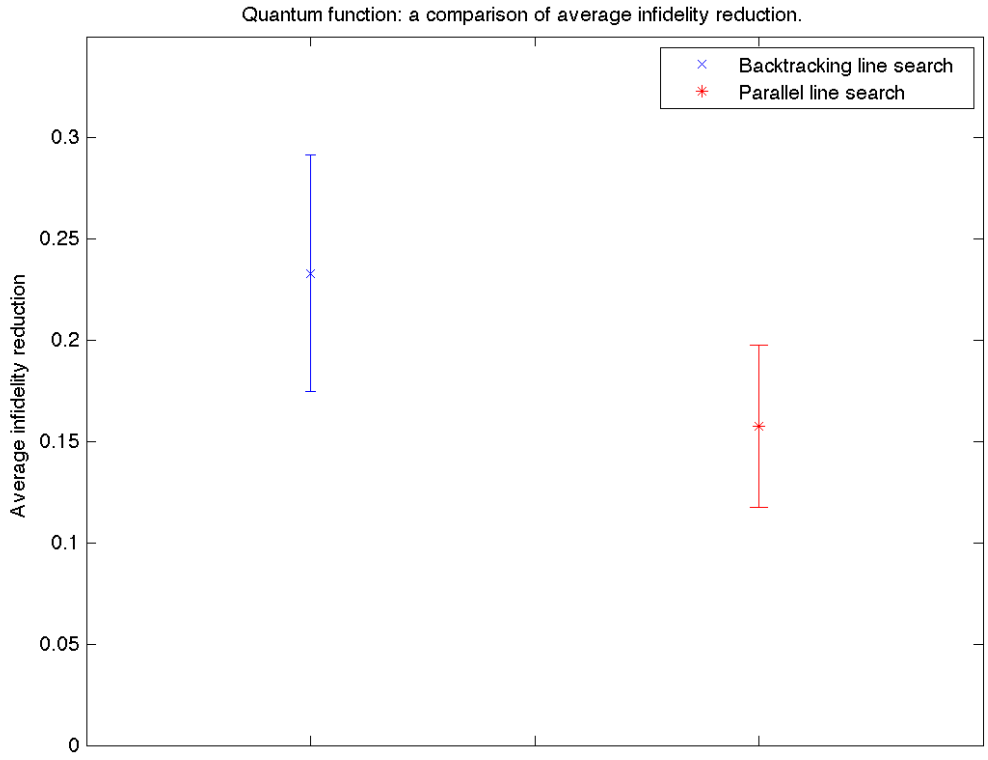


FIGURE 19

Figure 19 displays the average infidelity reduction of every L-BFGS run, not just the best runs. The aim of this graph is to display the efficiency and consistency of the L-BFGS algorithm, with the two line search functions. This graph clearly displays how helpful each line search algorithm is at aiding the optimization for finding minimums.

It is clear to see, that the backtracking line search has a much better average performance, where it constantly reduces the infidelity more than the parallel line search. Where as in comparison, on average the parallel line search performs much worse at aiding L-BFGS to find a minimum.

When considering the standard deviation, it is interesting to note that the parallel line search never outperforms the best performance from backtracking line search. Whereas the backtracking line search appears to outperform the parallel line search for the majority of the time.

These results suggest how much more suited the backtracking line search is to solving the problem, by consistently aiding the L-BFGS algorithm enough to reduce the infidelity more than if the parallel line search method is used. However, this maybe due to algorithm design, where the backtracking line search is looking for a sufficient decrease rather than an exact minimum. It may be, that by finding an exact minimum on the line it leads the optimization algorithm into local minima, where it is difficult to find an accurate search direction to the global minima.

There is also something to be said about the robustness of each algorithm, the results suggest that the backtracking line search will be able to find an infidelity close to a minimum in more cases than the parallel line search. For example, if the starting infidelity is around 0.25 and performance is based on the average infidelity reduction; the backtracking line search should be able to aid L-BFGS to reduce the infidelity to around 0.02. However, if the parallel line search is used, it may only reduce it to around 0.09. Although both results imply a high transfer probability (99.98% and 99.91% respectively), it would explain the results found in figure 19.

RUNTIME COMPARISON

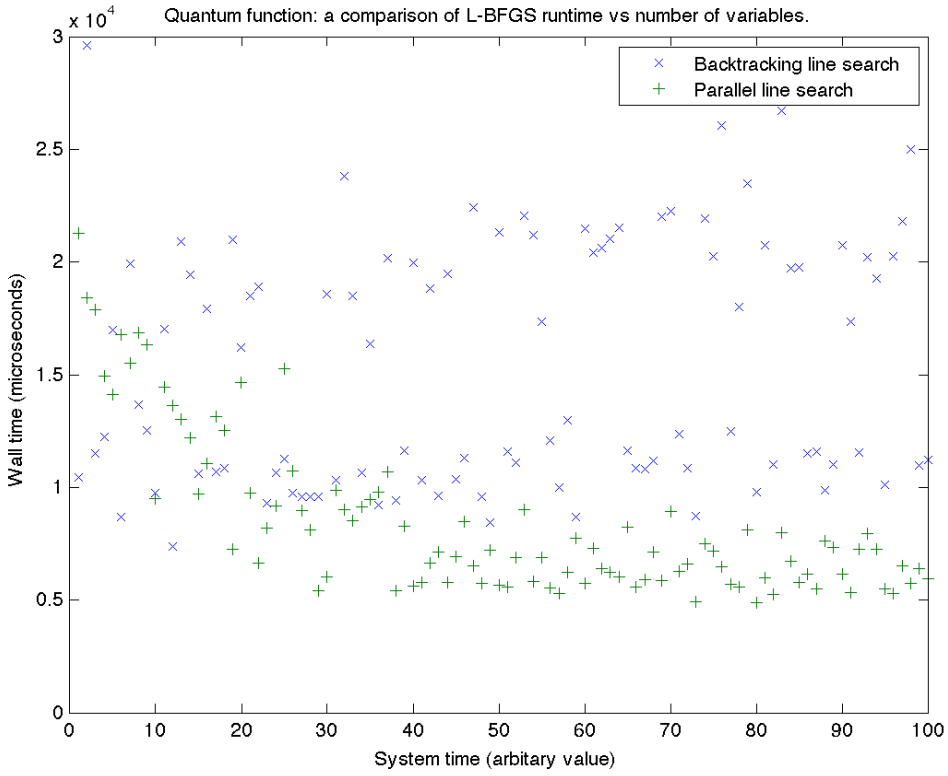


FIGURE 20

Figure 20 is a plot of the wall time for each optimization to run given the value of the system input time *t*. From this plot, it is possible to deduce a few distinct patterns in how the two algorithms are impacting the performance of the optimization. It is important to note that for early system time values, the search problem is extremely hard, explaining the scattered results.

Firstly, the parallel line search performance improves with the number of timeslots to calculate for. This is counterintuitive, as the evaluation function takes more time with more time steps, this may also suggest that the algorithm is finding a minimum in less iterations. Once the number of time steps (system time) reaches around 35, the parallel line search appears to have a linear performance with a narrow spread, suggesting a consistent behavior that will scale well with the number of variables.

In contrast, the performance of the backtracking line search is varied, and is spread across two distinct areas; one where the average time appears to be $1 * 10^4$ microseconds, and the other where the performance appears to be around $2 * 10^4$microseconds. This may be due to the number of line search iterations increasing, suggesting that the backtracking line search is low when the step size is not easy to find. It is also worth noting that once the number of system time slots reaches around 35 the performance is consistently worse than the parallel line search.

Finally, it is also worth noting the spread of the timing results for both of the line search algorithms, when the number of system time slots is below 35. The range of results could indicate the difficulty to find a minimum in the early stages, suggesting the search space has more local minima and maxima.

Overall this graph suggests that the parallel line search yields an improved performance in terms of overall optimization time, however this may be due to machine specific architecture. This is encouraging, as it also suggests that a more efficient parallel implementation could lead to a better increase in performance, as the current implementation can be improved.

COMPARISONS OF NUMBER OF ITERATIONS

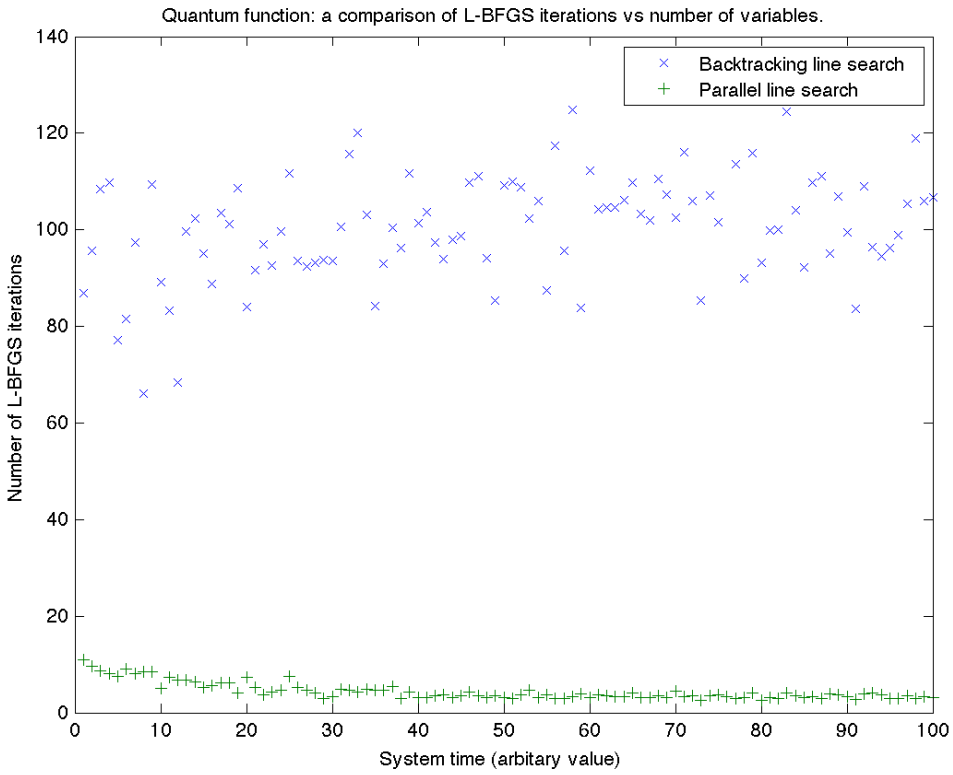


FIGURE 21

Figure 21 above represents the average number of L-BFGS iterations required to find a minimum or an exit condition, for all runs that did not fail. This graph is an analogue to the timing runs, where it is possible to distinctly measure algorithm performance independently from a computer's physical performance.

This plot clearly shows the two different behaviors of the line search algorithms, where the backtracking line search takes a many iteration lightweight approach, and the

parallel line takes few well-calculated steps. From these results, it is possible to conclude that the performance of the parallel line search is much more consistent than the backtracking line search, where it is consistently aiding the optimization algorithm to reach a termination condition in under 10 iterations. In contrast, the performance of the backtracking line search has a wider spread, and the average number of iterations increases over time.

With the results from figure 20 and 21, they both suggest that with improvements to the parallel algorithm runtime, the speedup could be substantial. As the number of total iterations is low, however the algorithm is not entirely efficient in the current implementation leading to a sub-standard runtime. There would be an obtainable speedup from a pure C/CUDA implementation of the quantum target and line search functions.

COMPARISON OF THE NUMBER OF FAILED RUNS

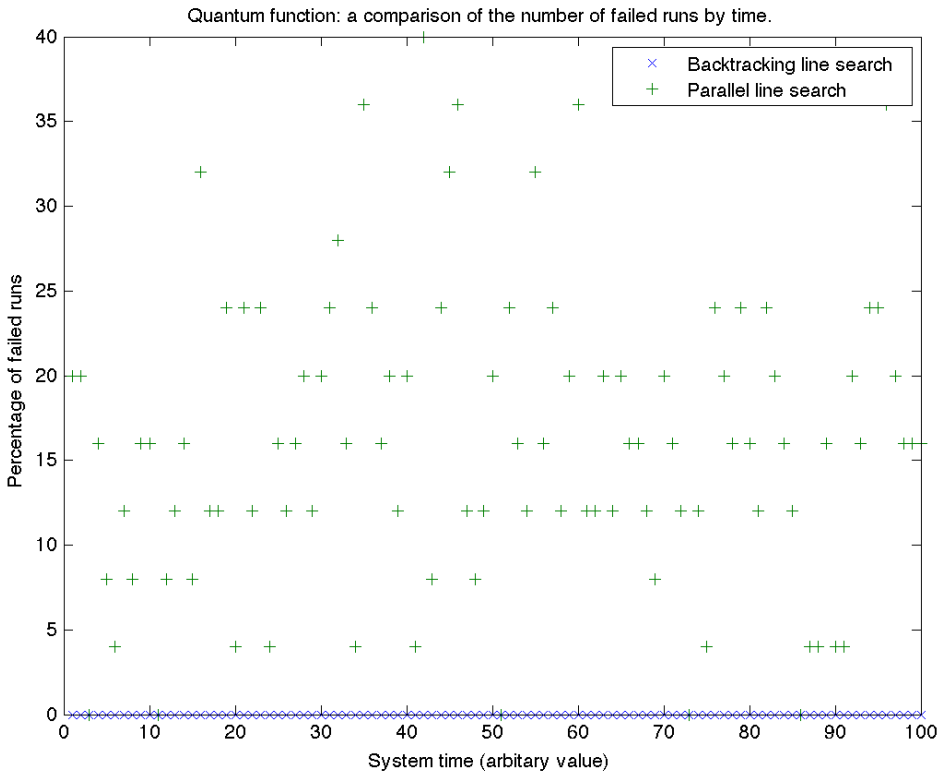


FIGURE 22

Figure 22 above represents the number of failed L-BFGS runs for all results, at a given system time. A failed run is where the system took one step or less. This definition of a failed run has come from the L-BFGS algorithm, where the first step is essentially a gradient descent. Once past the first step, the L-BFGS calculates a much more accurate direction from the inverse Hessian matrix. Typically, the first step for the system is often very small (> 1e-5), and this is an issue for a line search algorithm that is not efficient at searching through small spaces.

This plot shows the inefficiency of the parallel line search algorithm, where aiming to find a minimum in the search direction can be detrimental to program performance. In this instance the backtracking line search outperforms the parallel line

search by at least providing a step size. Due to the characteristics of the L-BFGS algorithm, it does not need an exact line search method, and it appears that the searching for an exact minimum can be detrimental to program performance.

These results suggest that employing the Armijo-Goldstein condition, rather than attempting to directly minimize, could improve the parallel line search algorithm. In this modification of the algorithm, the line search would sample in a similar way, but would find the condition with the greatest decrease in relation to the objective function. The sampling could be configured to segment the line into equal portions, and conduct a backtracking line search in each segment, then perform a reduction action that would return the step size α that best reduces the objective function in relation to the Armijo-Goldstein condition.

Controls produced

The following figure (23) is one example of a control produced for a ring of size 9, where the target was to transfer data from spin 1 to spin 9. The parallel line search reached an infidelity of 0.00034, suggesting a transfer probability of 99.99966%. In this occasion, the results for controls produced do not follow the symmetry implied by the initial controls.

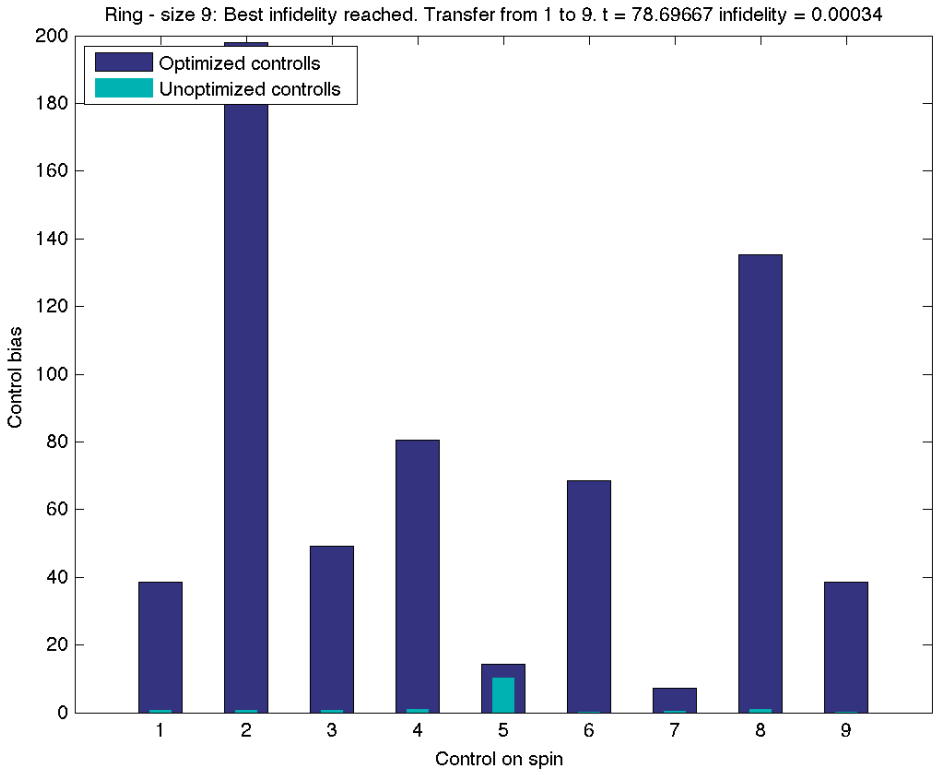


FIGURE 23

The infidelity reached in this example is an exceptional result, where around 1 bit would be lost per 300,000 bits transferred. This result is comparable to the packet loss of UDP packets across a network, where for a 8ms transfer the probability of loss is 0.23% (Bolot, 1993). A 8ms network transfer is comparable for the time taken to ping google.com from the Cardiff University network (appendix, figure 31). Considering the

uncertainty of quantum control, when compared to definite behavior of classical computing techniques this is an excellent result.

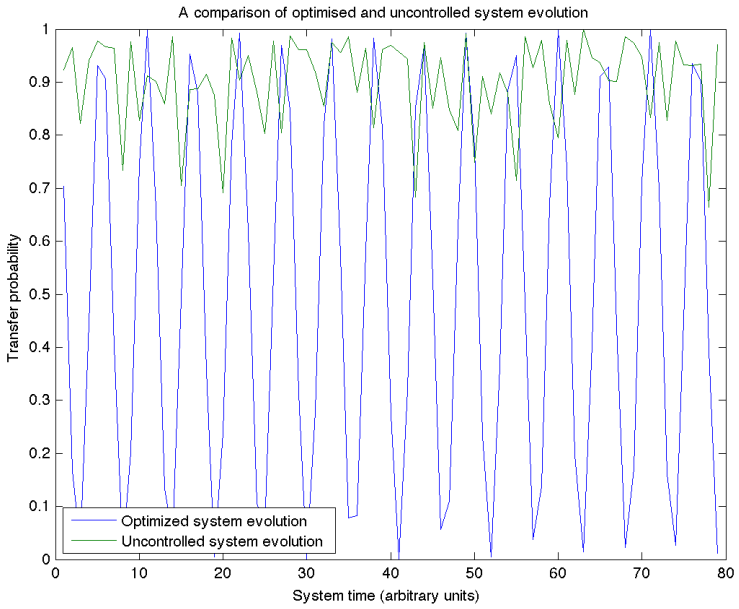


FIGURE 24 OPTIMIZED VS NATURAL SYSTEM EVOLUTION

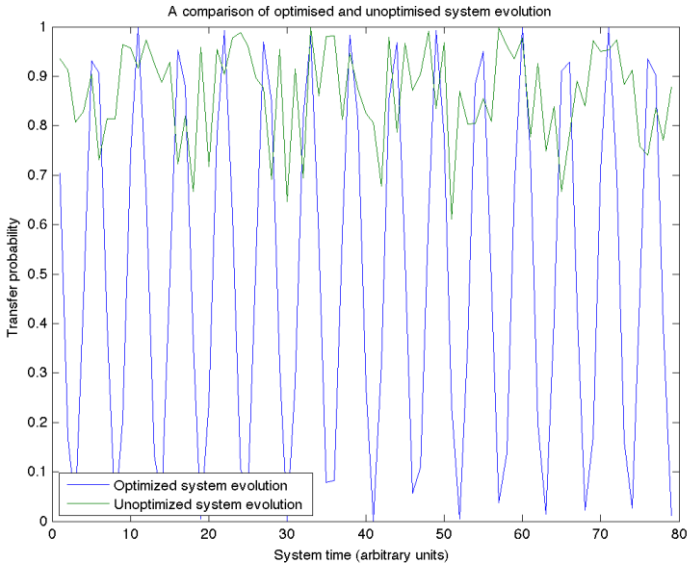


FIGURE 25 OPTIMIZED VS UNOPTIMISED SYSTEM EVOLUTION

To understand the effect that controls have on the system, figure 24 displays the transfer probability for the same system as before for an uncontrolled system compared to the natural evolution of the system given no controls. From this it is clear to see what effect the controls have on the system, where the natural evolution of the system is extremely sporadic, and how the best transfer probability achieved is around 35% where t is around 40. A well-optimized system provides a large improvement on the final transfer probability.

Figure 25 compares the probability of the un-optimized controls with their controlled counterparts. Although the random initial controls are designed to steer the optimization to find a global minimum, their effect is only marginally better than the natural evolution of the system. From here, it is clear to see how much an improvement the optimization can bring to the system.

Types of controls produced

From the results, there appear to be three classes of controls produced; one where the ring follows the initial system controls and behaves like a ring. Another, where the optimization appears to segment the ring into separate parts, and finally a set of controls where there is no clear indication of a pattern.

SYMMETRIC CONTROLLS

As previously discussed in initial research, it can be favorable to attempt to steer the optimization to a chain topology as transfer in chains is relatively known. This type of control appears in the smaller ring sizes more frequently as the optimal result.

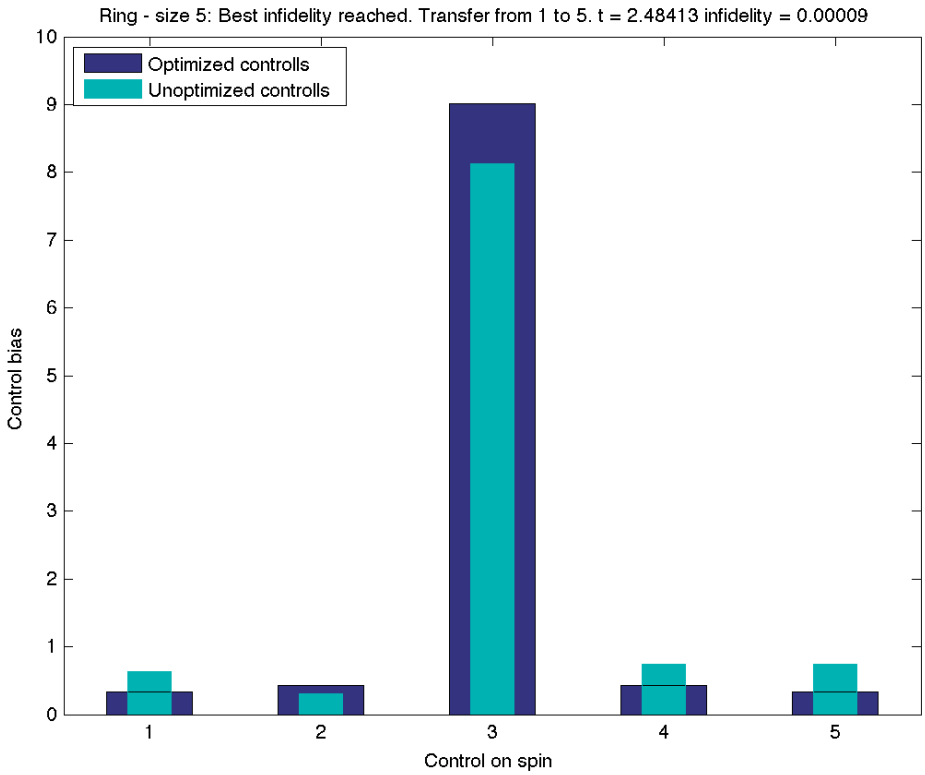


FIGURE 26

Figure 26 displays the type of controls achieved, where the optimization has made relatively little changes to the initial random controls. For a ring size of 5, this type of control holds the optimal transfer for every transfer (1 to 2, 1 to 3…).

RING SEGMENTATION

The second distinct type of control produced is less frequent than the symmetric chain type controls, and appears more frequently on larger ring sizes. This type of

control appears to segment the ring into distinct parts, where it seems to be isolating the qubit to only transfer through a much smaller sub-set of the network.

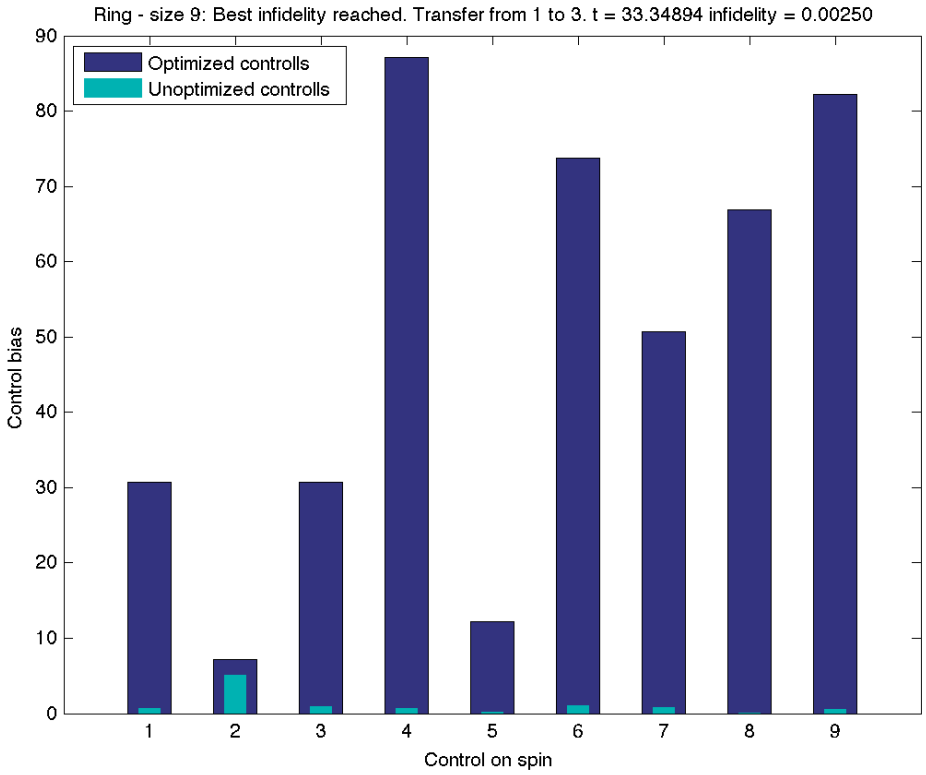


FIGURE 27

Figure 27 represents these types of controls, where in this instance the target is to transfer from spin 1 to spin 3. It appears that the optimization is isolating a subset of the network; by placing strong controls to create boundaries. In this instance, it seems to segment a network from, where spins 1,2 and 3 are isolated and spins 5,6,7, and 8 are isolated. However, as the initial system state is where the information is contained in spin 1, the probability of successful transfer to spin 3 is increased, as it should remain relatively contained in the segmented network. Figure 28 displays the theoretical links that are made with this control, denoted by dashed lines, solid lines represent the links and the gradient on spin 4 and 9 represent the strong controls.

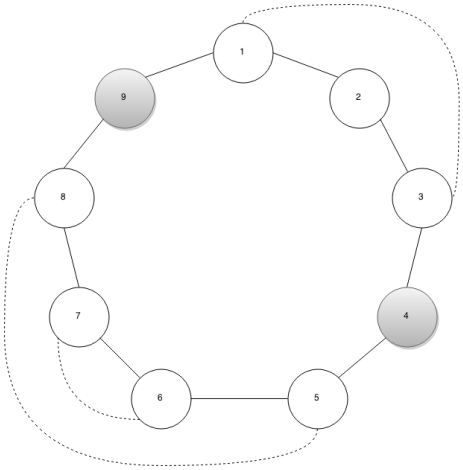


FIGURE 28

This diagram depicts a clearer picture of the possible behavior of the network given the controls. These assumptions are based on the behavior of spin chains. From here, it is clear to see how this would be advantageous to use this style of control for a transfer from spin 1 to spin 3.

UNKNOWN CONTROLLS

As the network size is increased, there are different types of controls that are produced, where there is no clear pattern. Figure 29 displays two types of optimal controls for a ring of size 13. Both of the solutions have a relatively high control bias on the initial and target spin, which would naturally suggest that the transfer is less likely (if compared to controls on smaller ring).

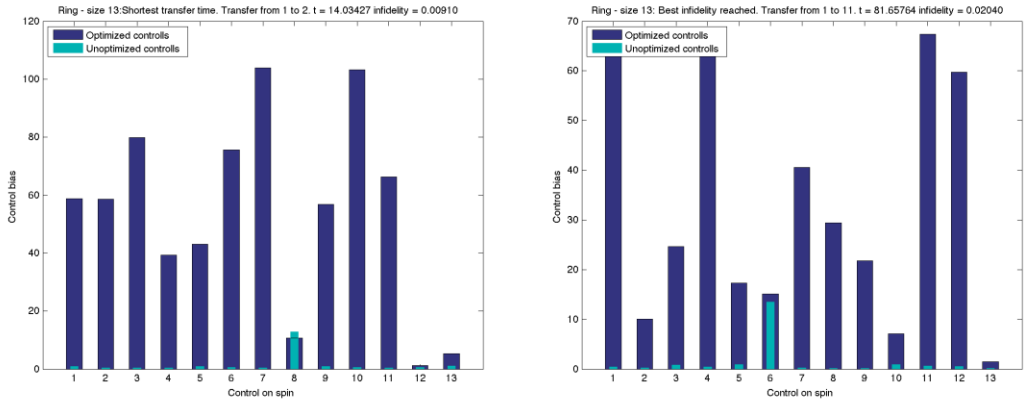


FIGURE 29

Evaluation

The range of controls possible represent the difficulty of the landscape that is being optimized, and how little is known about how these networks behave. Quantum spin chains are an incredibly difficult problem to optimize, due to the unpredictability of behavior inside the network. There is possibly a large range of quantum network controls that need to be categorized, which are optimal for each size of ring or transfer. In this instance, only odd size rings have been used for transfers in the attempt to steer optimization to symmetric ring like behavior. However even this know property of quantum networks provides sub-standard results compared to these unknown controls that are being produced.

Evaluation

From the results produced in terms of algorithm performance and results produced it is clear to see that this problem is extremely difficult, with very unpredictable behavior. Parallelization of current techniques has lead to a speed up when producing controls, however, due to the parallel algorithm being under-developed, it lead to a worse performance in terms of average transfer probability achieved.

The method employed in this study was designed around gathering a large range of results with an acceptable level of accuracy. The narrow band of conditions used to collect results provided ample results for a basic level of analysis, however these results will be hard to apply to a larger group. This is because there are many other network topologies that need to be researched, rather than just rings of odd size. For example, the controls produced for a ring of size five will not necessarily work on a ring with even number. Overall, the method for this project was successful at producing descriptive test and main system results.

The two line search techniques take two different approaches to computing a suitable step size, where currently the backtracking line search yields a better performance in terms of accuracy and precision. However the parallel line search performs significantly better in terms of system wall time and number of iterations. A future approach could be designed to combine components of each line search; search for a suitable decrease in parallel, rather than an exact minimization.

In this paper, there has been minimal discussion of parallelizing the quantum target function; due to the implementation given the time constraints. This process could potentially benefit from being implemented on GPUs, where matrix calculations suit the architecture of the cards. There were efforts made to develop a C implementation of the target function, however it was dropped due to time constraints.

When comparing the controls procured, it is simple to see that due to the unpredictability of quantum spin networks it is hard determine what controls are most efficient. Considering that this application was designed to steer the optimization algorithm to one type of possible control, there are results that contradict this logic, suggesting that there are many different classes of controls.

The method in this project was restrictive to a certain subset of results; this is due to the long wall time of calculating an entire set of results for a ring. On the computer designed for this project, with it running 24/7 it took just under 20 days to collect the results. If more time was available, more ring sizes would be tested, a larger range of times would be tested and the number of evaluations per time would be increased. This would allow for a more in-depth study of algorithm performance and the behavior of spin networks.

When comparing sequential and parallel results, only a ring of size 5 was used. This is extremely restrictive, as looking at the results there appears to be one type of control that is dominant. If there were more time available it would be beneficial to compare performance across several ring sizes, and to see if the line search influenced the types of controls produced on different network sizes.

Conclusion

The aim of this project was to develop a GPU parallel implementation of the L-BFGS algorithm, which would be applied to a quantum control problem. During the course of this project, it quickly became apparent that it would not be possible to achieve this target in the time given. However, a different approach was taken to display a proof of concept; test functions that displayed CUDA functionality, and a CPU parallel L-BFGS algorithm applied to a quantum target function.

Overall, the system produced displays promising results when applied to the quantum control problem by reducing wall time and number of system iterations. This suggests that the methods explored in this paper could be applied to a more efficient parallel implementation, which would lead to larger increases in performance.

The method explored in this paper was designed around saving time, where only a small range of network topologies, transfers and sizes were collected. Despite a timely development of the parallel algorithm, it took over three weeks to collect all results, which is a large portion of the total time for the project. For this reason, it will be incredibly hard to abstract this study to other network topologies, sizes or transfers.

The results produced displayed some interesting characteristics of spin chains, where the output controls did not indicate clear behavioral patterns. The results produced are varied, despite the initial controls attempting to steer the optimization algorithm to a symmetric-chain type control. The results indicate the complex behavior of spin networks, where it may be beneficial to research a classification of the different types of controls that suit different network topologies.

The development of the system went smoothly overall; the largest issue in the project that consumed time was the setup of Matlab and integration with CUDA and Windows 8. However, this was expected and was integrated into the initial plan, meaning each section of the report was delivered on time.

In conclusion, this project has provided a good proof of concept, where a parallel implementation of the L-BFGS algorithm brought benefits in terms of runtime and number of iterations. In future work, this paper can be the basis of a faster and more accurate algorithm.

Future improvements

A CUDA PARALLEL IMPLEMENTATION

There are several areas where this work can be improved, where the largest improvement could be made through a different parallel implantation in GPGPU programming techniques. This form of parallelization is particularly suited to matrix arithmetic and may yield faster results. This is due to having many (often thousands) lightweight threads that will perform individual operations, and then perform a reduction to collate results.

In this implementation, the quantum target function may see large performance increases in terms of wall time due to the large number of matrix operations. Although this implantation would no longer be based on code written for Matlab, it would still be possible to compile and run the program from Matlab. This would allow for continuation of the use of Matlab's graphing and analysis functions.

MACHINE LEARNING OF INITIAL PARAMEMTERS

In this paper, estimation of initial controls was based on random number generation. It would be interesting to collect data on networks where the initial controls where the same, the output for each network size, time and transfer would become the basis for a learning data set. From there, using a machine learning technique it would be possible to estimate initial system bias parameters based on previous controls. The aim here would be to create a system that could limit the search space for the optimization algorithm by providing an accurate initial guess at system parameters. For example: *Transfer qubit of information from spin 1 to spin 9 in a network of 10 spins, what time and controls would maximize transfer probability?*

A MORE DETAILED INVESTIGATION OF TIME

In this application, the time search is very coarse, where the distance between points in time is relatively large. It would be beneficial in the future to decrease the distance between step sizes in time, and increase the number of time slots. For example, 500 time steps with a delta size of 1, rather than 50 time slots with a delta size of 10. This would provide a much more detailed, step-by-step evolution of the system, it would allow for a more detailed understanding of how the system evolves, and what times are optimal for different transfers. However, due to the direct relation between the number of time steps and the time taken for the quantum target function this would greatly increase the wall time of the system. In the final version of the system evaluation in this project, there are the following number of operations in relation to the number of spins (s) and timeslots (t).

$$(3 * t) + t^s + s$$

Considering that the line search function runs this target function at least 2,500 times for each iteration until an exit condition is met, the number of timeslots evaluated should be kept to an absolute minimum; unless wall time is not an issue.

AN IMPROVED LINE SEARCH METHOD

Currently, by exactly minimizing to achieve the step size α may not be the best solution, as L-BFGS does not need an exact step. It would be beneficial to firstly restrict the search space, by not allowing the line search algorithm to recuse to find the exact minimum. In this version it would sample the line at regular intervals once. From there, instead of fining the minimum of the values, it would be better to use a condition for

finding the best value in relation to either the Armijo-Goldstein condition or the Wolfe conditions.

This change in the line search behavior would greatly decrease the line-search wall time, but decrease the accuracy currently offered. However, this approach combines the benefits of a normal backtracking line search with these conditions, which has been proven to be more accurate in this paper, with the speed of the parallel line search. If anything, these conditions combined with a parallel approach would provide a more accurate result than a normal backtracking line search.

A DIFFERENT LINE SEARCH APPROACH

Simulated annealing is a useful method when attempting to find optima, where it takes large step sizes around the search space that eventually become smaller and greedier around optimal values. The aim of simulated annealing is to provide a result in the time given, in this application; it would produce a step size α along the line. The step size α could also be inline with the Armjio-Goldstein or Wolfe conditions as previously mentioned. This line search technique may have benefits over the parallel line search, as it would provide a definite answer in a given time, however it may not nessasarly be the optimum.

Extensions to this work

DIFFERENT NETWORK TOPOLOGIES

In this paper there is only an exploration of rings that have an odd number of spins, this research has a very narrow scope that is hard to generalize. It would be more appropriate to explore a larger range of network topologies; chains, rings, stars, hypercube, fully connected, bus, mesh networks or any combination of any of these. By exploring the different network topologies, it would provide a better understanding of how network topology affects the transfer of qubits.

NO RESET BETWEEN TRANSFERS

Currently, the network is simulated from a neutral starting position for every transfer. It may be possible to introduce a second transfer after the first in a system without resetting back to a neutral position. The major issue here is that when a qubits state is measured it affects the state of the system. It may be possible to shut down the qubits used for previous transfers with strong controls and continue to use the rest of the system for the future transfers.

ADD TIME AS AN OPTIMISAION PARAMETER

Currently the system is prescribed a time when the data is going to be measured from the system. It would be interesting to explore time as another optimization parameter, as there may be times where a greater transfer probability can be achieved. However, when comparing the minimum infidelity in figure 17 over time, there appears to be a uniform distribution, suggesting that there are no optimal times for transfer if optimal controls are found.

Another option would be, given a set of controls that are not optimized, find the best time for the transfer. However, there appears to be no real benefit to this scheme, as there are few real world applications where this would be the optimal choice.

CLASSIFICATION OF CONTROLS

In this paper, there are two different types of controls described that produced some of the optimal results (symmetric chain and partitioning of rings). However, as the ring size increased, there were some complex controls that were observed of which I could not explain the behavior inside the network. I believe that there is a large set of quantum spin network controls that should be classified, not just for rings, but also for all network topologies. The classification should come from a scheme where a large set of network sizes is checked, transfer from and too all nodes is checked and where initial controls have a uniform distribution; such as all random numbers in the range 0 to 1. These controls would be the benchmark for describing controls in spin networks and allow a broader understanding of how these controls manipulate the state of the system.

EXPAND AND COMBINE NETWORK TOPLOGIES

As the study of this paper is around the application of a 'quantum-router', the natural evolution of the system would be to extend the topology to contain other network components. Figure 30 below displays a network where chains have been added to certain ring spins, in a hardware application the end nodes may represent different computer components; hard drives, processors or memory units. Or on a much larger scale, a small network that contains identities where the target is to get data from Alice to Bob.

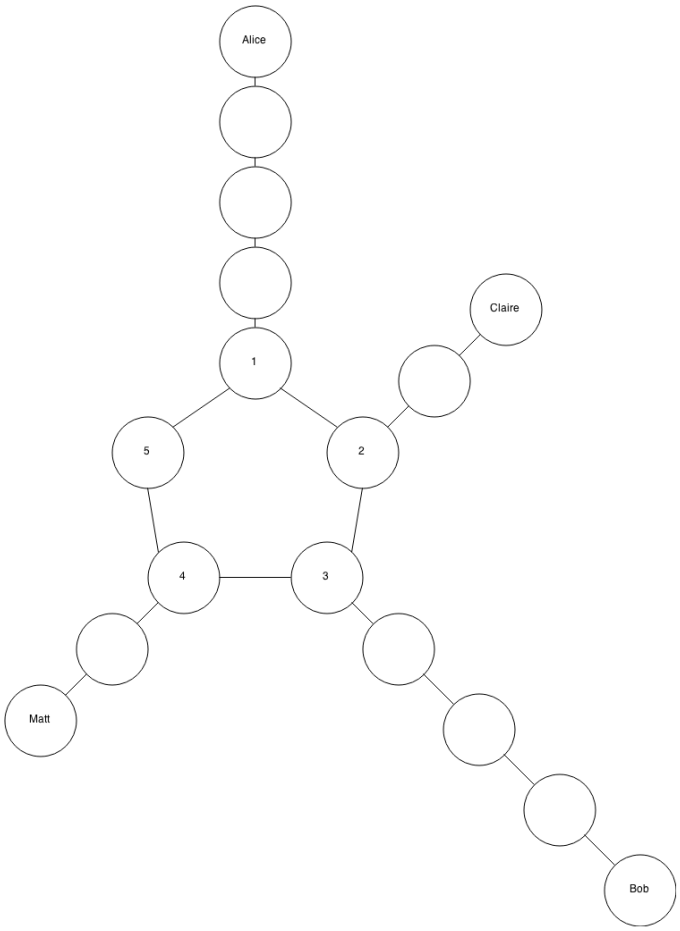


FIGURE 30

Reflection on learning

In reflection, this project has taught me a great deal of applying a scientific method to a project to achieve a successful outcome. In the past, I have had experience on working on projects with a specific aim; such as developing a piece of software with the functionality 'x'. However, this dissertation was based on a much more exploratory approach, where planning program development relies heavily on current process and discovery. This structure is completely different to any that I have done in the past, where the outcome was unknown. It was essential to constantly review and reflect on current progress to steer the project towards a modified outcome; CPU parallel rather than GPU implementation.

When compiling results from a scientific study, displaying them in the correct manner is essential. This project has produced an extremely large data set of results, where the individual values produced have little value. Whereas combining results into descriptive graphs can display a large amount about algorithm behavior that is not readily apparent when viewing the raw data.

The project development process has taught me how important it is to keep up-to-date with current research, where recent papers can aid in the development of your own research. Throughout the project development I managed to follow the plan highlighted in the initial project report (see appendix, figure 32). This allowed me to complete the project with time to allow for results collection, which took a substantially long time.

Personal time management in this project was extremely important, as the development and competition on time relied on my commitment. This was not too much of an issue throughout the project, as the subject area is interesting to me. By having a supervisor meeting every week helped develop ideas and provide solutions for current problems. Through the use of GitHub in this project, it allowed me to keep an accurate track of what I had achieved in the previous weeks, it has also aided in the writing of this final report by allowing me to reflect on key milestones in the project.

In terms of amount learnt, this project has taught me a great deal about scientific research methods, quantum mechanics, control theory and optimization. This project was chosen due to an interest in quantum computing. However the largest limiting factor was how little past experience I had with mathematics of this level; I had to spend a month learning A-Level math before trying to understand how these systems were formed. Despite this, I have finished this dissertation with a much deeper understanding of mathematics and physics.

Overall, I have learnt a great deal on how to apply a scientific method to achieve accurate and descriptive results; in an application like this, it is vital to provide data to back up a statement. Presenting scientific data in a meaningful way is crucial to allow readers to effectively understand exactly what is being presented.

Table of abbreviations

- **Qubit**: A quantum bit of information, the analogue of a digital bit of information.
- **BFGS** : The Broyden–Fletcher–Goldfarb–Shanno algorithm, a second order quasi-Newtonian optimization method.
- **L-BFGS**: The Limited-memory-BFGS algorithm, a variant of BFGS where large matrices are not stored.
- **Spin**: (in reference to quantum spin), the angular momentum of a molecule's spin around its own axis. There is also an orbital form of spin, such as when a molecule orbits a nucleus.

References

Åkerman, J. (2014, April 22). Toward a Universal Memory . *Science, 308*, 508-510 .

Alexandrov, O. (2004, November 19). *Gradient descent.png.* Retrieved March 12, 2015, from Wikimedia: http://commons.wikimedia.org/wiki/File:Gradient_descent.png

Alexandrov, O. (2007, June 20). *Conjugate gradient illustration.svg.* Retrieved March 14, 2015, from Wikimedia: http://commons.wikimedia.org/wiki/File:Conjugate_gradient_illustration.svg

Baylis, W. E., Hushilt, J., & Wei, J. (1992). Why i? *American Journal of Physics, 60*(9), 788–797.

Bolot, J.-C. (1993). End-to-end packet delay and loss behavior in the internet. *SIGCOMM '93 Conference proceedings on Communications architectures, protocols and applications, 23*(4), 289-298.

Christandl, M., Datta, N., & Andrew J. Landahl, A. E. (2004, May 4). Perfect state transfer in quantum spin networks. *Physical Review B, 92*, 187902.

Cui, J., & Mintert, F. (2014, July 4). Long distance entanglement in disordered spin chains . Cardiff.

Dirac, P. A. (1939, July). A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society, 35*(03), 416-418.

E. Jonckheere, F. C. (2014, July). Quantum networks: Anti-core of spin chains. *Quantum Information Processing, 13*, 1607-1637. Retrieved from Cornell University Library.

Eichmann, U., Bergquist, J. C., Bollinger, J. J., J. Gilligan, M., Itano, W. M., & Wineland, D. J. (1993). Young's Interference Experiment with Light Scattered from Two Atoms. *Physical Review, 70*(16).

Fei, Y., Rong, G., Wang, B., & Wang, W. (2014, May). Parallel L-BFGS-B algorithm on GPU. *Computers & Graphics, 40*, 1-9.

Hai-Jing, W., Chang S., S., Claudia E., A., Scott J., S., Dmitry, B., Alexander, P., & Vikram, S. B. (2013, June 05). Sensitive magnetic control of ensemble nuclear spin hyperpolarization in diamond. *Nature Communications, 4*.

Hauser, D. R. (2007, January 01). *Line Search Methods for Unconstrained Optimisation.* Retrieved March 26, 2015, from University of Oxford Mathematical Institute: ttps://people.maths.ox.ac.uk/hauser/hauser_lecture2.pdf

Jonckheere, E., Langbein, F., & Schirmer, S. (2014, Aug 16). *Information Transfer Fidelity in Networks of Spins.* Retrieved January 12, 2015, from Cornell University Library: http://arxiv.org/abs/1408.3765

MathWorks. (2015, January 05). *Run Built-In Functions on a GPU*. Retrieved March 12, 2015, from MathWorks: http://uk.mathworks.com/help/distcomp/run-built-in-functions-on-a-gpu.html

Naoaki, O. (2014, 12 5). libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). Sendai, Sendai, Japan.

Pardalos, P. M., & Yatsenko, V. (2008). *Optimization and Control of Bilinear Systems.* New York: Springer.

Rosenbrock, H. H. (1960). An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal, 3*(3), 175-184.

S. Machnes, U. S.-H. (2011, August). Comparing, Optimising and Benchmarking Quantum Control Algorithms in a Unifying Programming Framework. *Phys. Rev. A, 84*(2), 0-23.

Schirmer, S., & Langbein, F. (2014, Mar 2). Characterization and Control of Quantum Spin Chains and Rings. Cardiff.

Schrödinger, E. (1926, December 1). An Undulatory Theory of the Mechanics of Atoms and Molecules. *Physical Review, 28*(6), 1049–1070.

Simionescu, P. (2006, November 17). *Banana-SteepDesc.gif.* Retrieved March 12, 2015, from Wikimedia: http://commons.wikimedia.org/wiki/File:Banana-SteepDesc.gif

Vandersypen, L. M., Steffen, M., Breyta, G., Yannoni, C. S., Sherwood, M. H., & Chuang, I. L. (2001, December 20). Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature, 414*, 883-887.