Characterizing Computational Techniques for Experimentally Feasible Non-Local Evolution of Two-Qubit Systems

Shankar Balasubramanian Didi Chang-Park Kyle Herndon Zane Rossi

TJHSST Modern Physics and Optics Lab

Abstract: This paper investigates computational methods and presents a computational framework for simulating non-local evolution of two-qubit systems. This evolution may then easily be combined with finite local operations to give control of near-arbitrary precision. Current research has focused on developing the theoretical background which we hope to computationally implement. We will characterize how useful operators (particularly the *CNOT* gate) are generated in this process, and they will be approximated and performed computationally in optimal time utilizing a adaptive-algorithm approach. We will also analyze the non-local piece of the system Hamiltonian, including determining experimental means of generating it. Particularly, we will be evaluating its action on the Poincaré-Bloch-Sphere.

Mutation Protocol

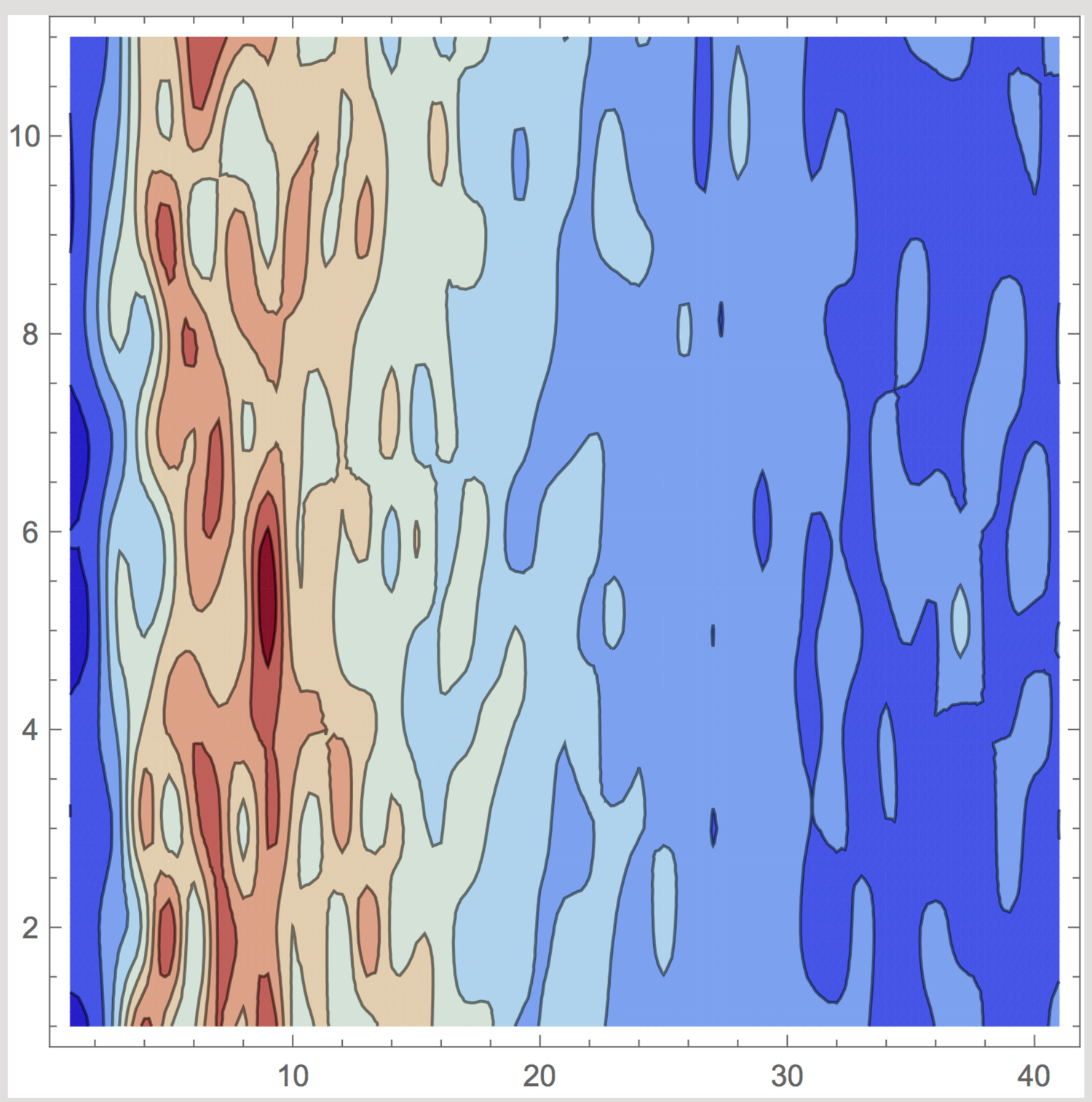

Figure 1: This plot shows the fidelity of the best organism in a ten-organism set after 100 generations with respect to mutation chance and mutation magnitude. The vertical axis, when the value is multiplied by 0.1, gives the chance in any generation for one of the angles defining an organism to be mutated. The horizontal axis, when multiplied by 0.025, gives the mutation magnitude (this ranges between 0 and 1 radian(s)). The plot shows clear correlation between mutation magnitude and organism success.

Mutations are designated on the basis of chance and magnitude. We set a particular threshold value, which we are free to change. This threshold value (α) determines whether or not a particular organism's parameters will be mutated. The parameters, as discussed above, are the β' rotation coefficients which are specified in the exponential form of the local operations. The other changeable parameter, μ, specifies the degree to which the angles will be mutated. These mutations encompass an envelope of random values between $-\mu$ and μ (a uniform distribution) which will be added to the current β' values to update them. Figure 1 depicts the efficacy of the simulation (particularly, the fidelity), for different values of α and μ. Lighter colors represent simulations resulting in higher fidelities. The values of α and μ do not change across the entire simulation; however, the efficiency of our program can be increased by modifying these values per each generation

The Space as a Whole

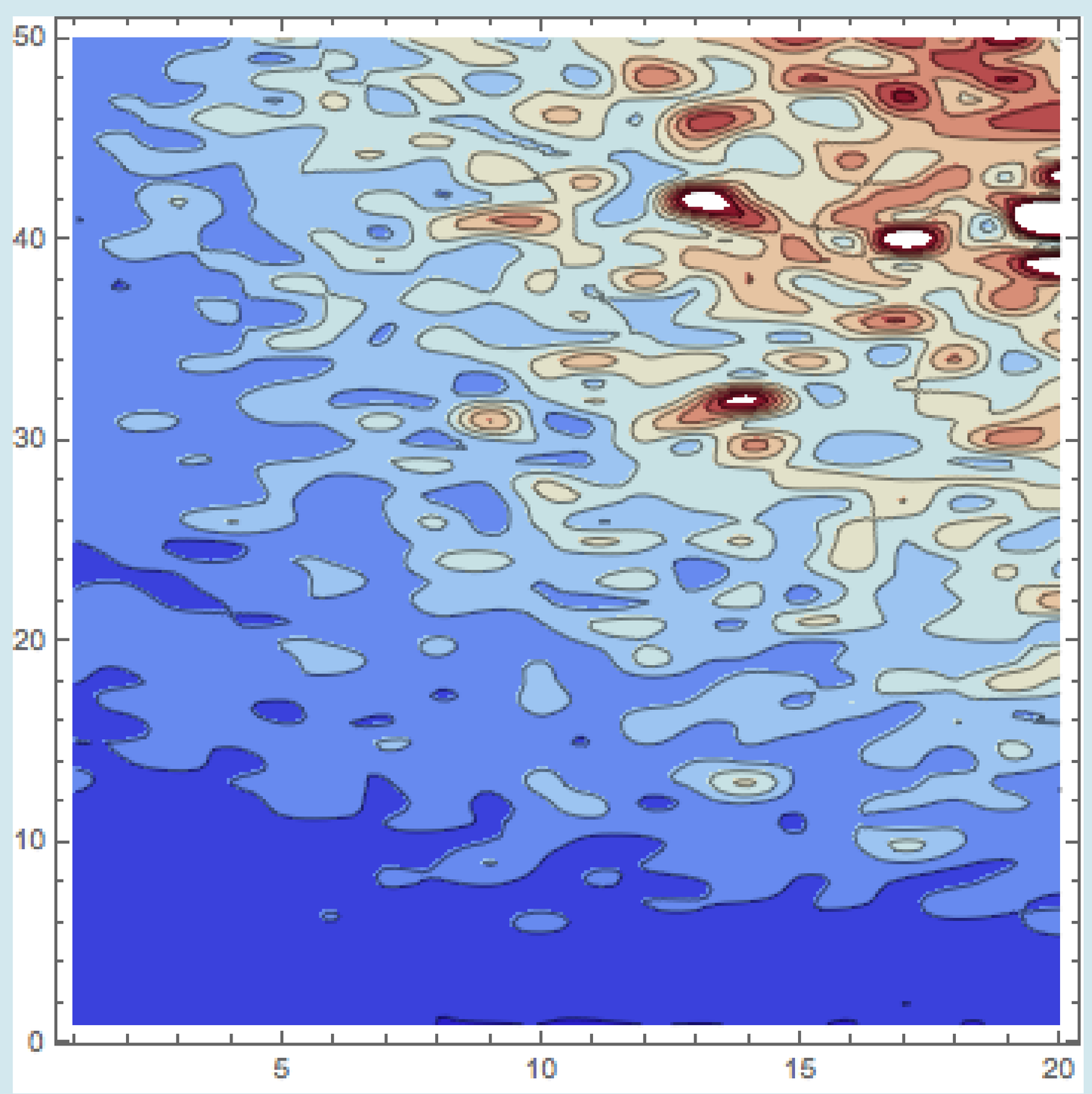

Figure 2: Caption. Caption Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption.. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Ca Ca Caption.ption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption.ption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption.Caption. Caption. Caption. Caption. Caption. Caption. Caption.

Fig

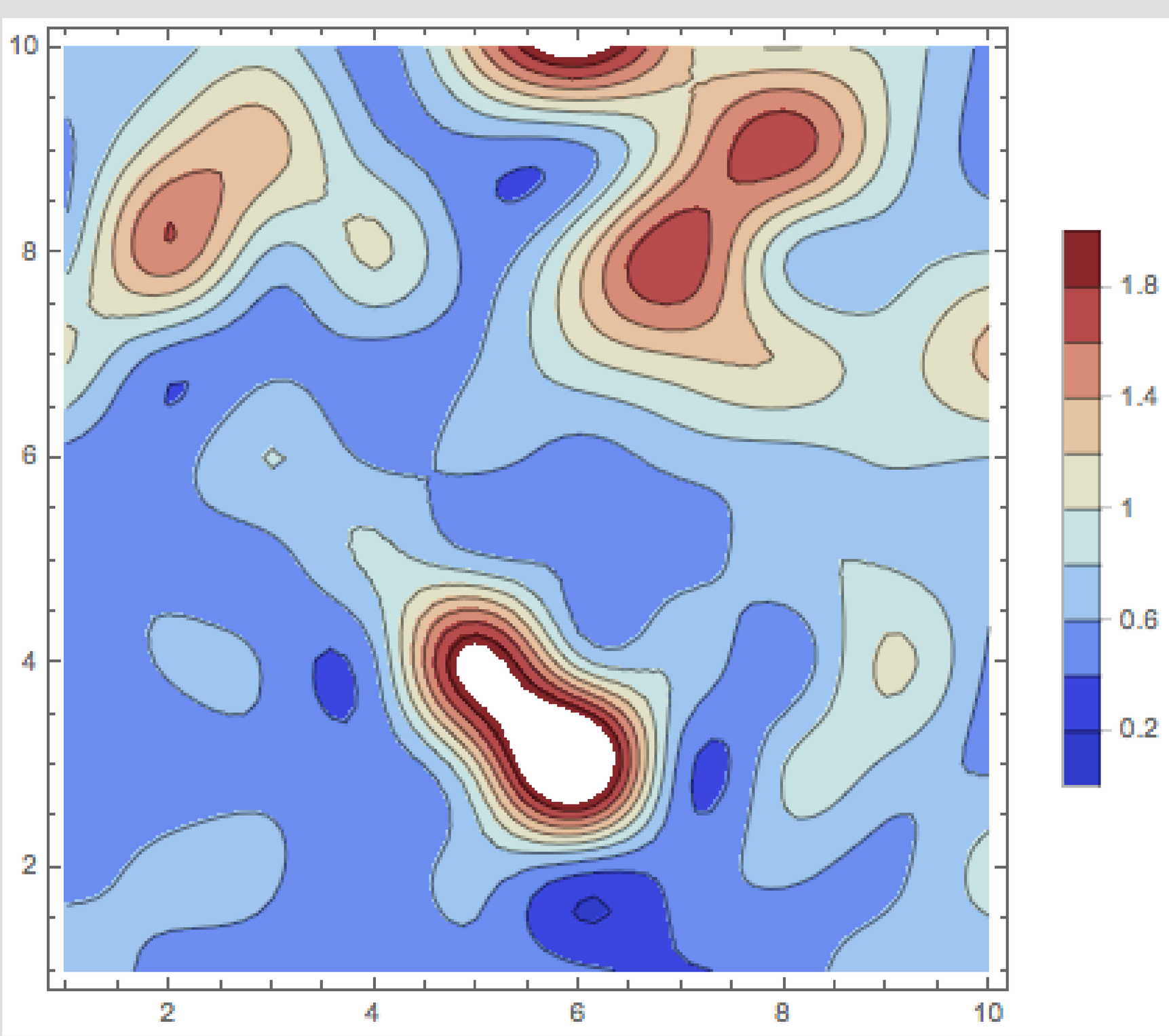

Figure 3: Caption. Caption Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Ca Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Cap Caption.tion. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption. Caption.ption.. Caption.

Variation in Population

A series of methods is used to determine fidelity in a populations of a set size, generation number, and gene length. We can then produce a series of partially-evaluated functions that can be mapped over a list of integers. When the results of this mapping are plotted, we gain knowledge of a slice of the search space. Knowledge of these slices, analogous to partial derivatives, can help determine which variables produce the best solutions with the smallest computational cost.

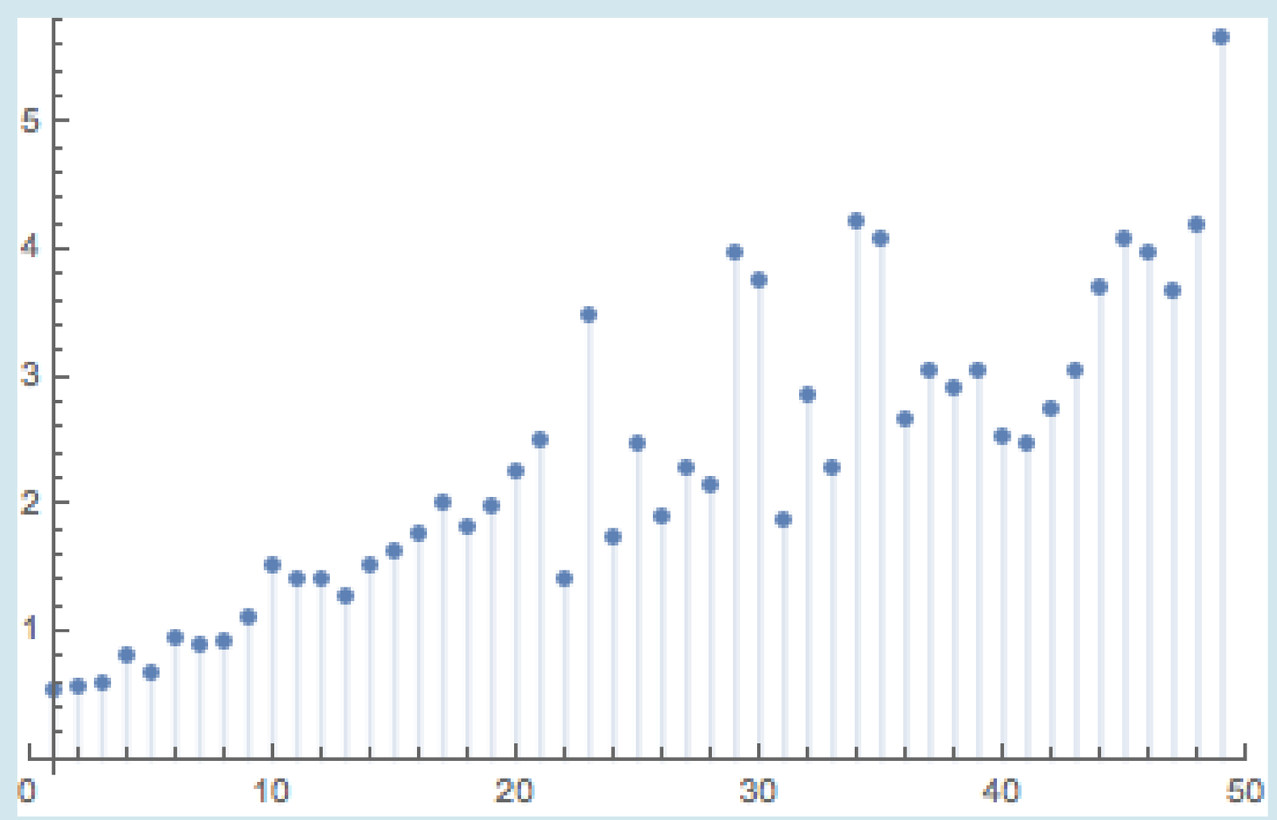

Figure 4: A plot of the average of 20 runs in which the population is along the horizontal axis, and fidelity ($\mathcal{F}$), is along the vertical axis. During these runs, the organism population was brought through 20 generations, with each organism having two genes. The mutation chance (α) and mutation magnitude (μ) are drawn from the maximum of Figure 1 (0.5 and 0.225 respectively).

Here we see a plot of constant generation number and gene length, where population size is being altered. Over a large average, the upward trend is clearly visible, as would make sense, and closely follows an exponential trend. We can conjecture that, given a saturation of the space with organism, we may eventually approach a point of diminishing returns, but that is not apparent in Figure 4. These factors indicate a volatile variable, where we are sacrificing memory for some time and search space-covering.

Variation in Generation Number

The same process is used as described above, save now the slice through the search space is made in a way orthogonal to the original slice. On average, due to the structure of the space, the behavior along one slice is related to that at adjacent slices. We see a similar but not identical behavior.

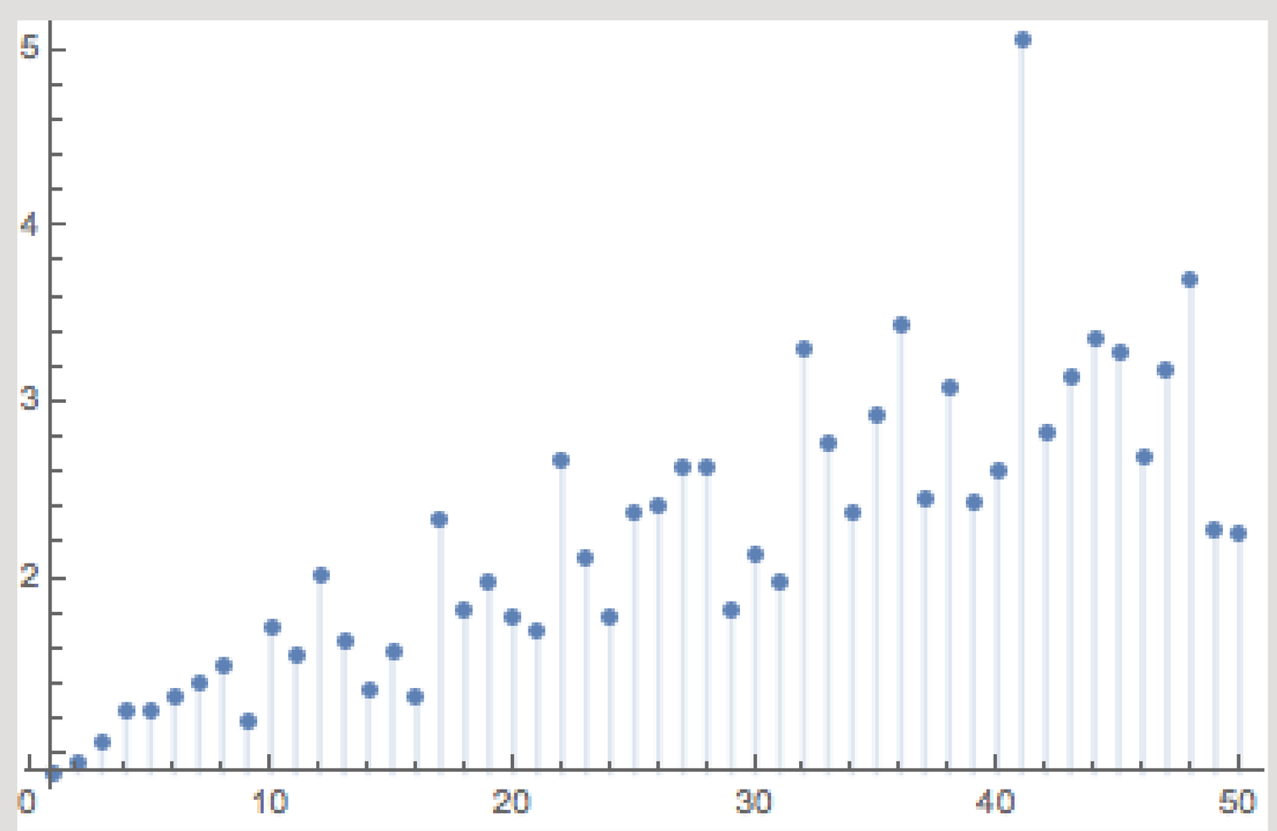

Figure 5: A plot of the average of 20 runs in which the generation number is along the horizontal axis, and fidelity is along the vertical. During this run, the number of generations is taken from 1 to 50, while the gene number remains at 2, and the population is constant at 20. (α and μ are preserved from Figure 4)

There is still a general upward trend when the data are aggregated. However, the growth with respect to generation number appears more logarithmic in character. This view is highlighted further in the paths along Figure 2. These factors indicate that generation number is not as volatile a variable.
There are a number of explanations for this, but the most intuitive is that that size of the space is simply too large to be worked through by mutation alone. When there are few organisms, they stay within a close neighborhood of their original location (when breeding is not introduced, as it was not in these plots.) However, when the initial system is widely populated, each organism can work with the others to `patch' the space.

Random shit

Main Body Code

```
In[1]:= SIGX = {{0, 1}, {1, 0}};
        SIGY = {{0, -I}, {I, 0}};
        SIGZ = {{1, 0}, {0, -1}};

        TimeIndependentEvolution[operator_, t_] := MatrixExp[-I * operator * t]
        TIE[operator_, t_] := TimeIndependentEvolution[operator, t]

In[6]:= mixedEvolution[hamiltonian_, dt_, A_, B_] :=
         TIE[hamiltonian, dt].KroneckerProduct[A, B]
        eulerMixedEvolution[hamiltonians_List, times_List, listA_List, listB_List]
         (*This function does the above but with lists*) := {product = IdentityMatrix[4];
          For[i = 1, i ≤ Length[hamiltonians], i++,
           product = product.mixedEvolution[hamiltonians[[i]], times[[i]], listA[[i]],
              listB[[i]]]]; product}
        productEvolution[hamiltonian_, time_, listA_List, listB_List]
         (*This function does mixedEvolution but with lists for only A's and B's*) :=
         {product = IdentityMatrix[4];
           For[i = 1, i ≤ Length[listB], i++,
            product = product.mixedEvolution[hamiltonian, time, listA[[i]], listB[[i]]]];
           product}[[1]]

In[9]:= matrixGenerator[alpha_, beta_, gamma_, delta_]
         (*Creates a matrix corresponding to the coefficient parameters passed*) :=
         alpha * SIGX + beta * SIGY + gamma * SIGZ + delta * IdentityMatrix[2]

        matrixGenerator2[phi123_List]
         (*Creates a matrix corresponding to angles defined by a list *) :=
         matrixGenerator[Cos[phi123[[1]]], Sin[phi123[[1]]] * Cos[phi123[[2]]],
          Sin[phi123[[1]]] * Sin[phi123[[2]]] * Cos[phi123[[3]]],
          Sin[phi123[[1]]] * Sin[phi123[[2]]] * Sin[phi123[[3]]]]

        randomMatrixGenerator[] (*Creates a random unitary hermitian matrix*) :=
         {phi1 = RandomReal[{0, Pi}];
           phi2 = RandomReal[{0, 2 * Pi}];
           phi3 = RandomReal[{0, Pi}]; matrixGenerator[Cos[phi1], Sin[phi1] * Cos[phi2],
            Sin[phi1] * Sin[phi2] * Cos[phi3], Sin[phi1] * Sin[phi2] * Sin[phi3]]}[[1]]
```

Figure 6: A plot of anarchy.

Background

The ultimate goal of Control Theory is the ability to arbitrarily manipulate the state of an ensemble of qubits. In the context of quantum computation, this problem is reduced further: basic quantum gates allow for modular systems [2]. However, the generation of these gates is non-trivial, and the arbitrary Hamiltonian is, in general, not readily constructible. Often one will wish to achieve a specific unitary evolution operator or gate U corresponding to the solitary time evolution of a Hamiltonian H. When H may not be engineered easily, the evolution can be achieved in other ways [2] , often involving the use of local operations (LO), repeated measurements on the system [1], along with standard Hamiltonian-driven evolution drawing from some collection (possibly a one element collection) $H_1, H_2, \cdots, H_n$, all of which *can* be lucratively generated.
In the simplest case, a two particle system, the Hamiltonians involved may contain non-local terms, usually corresponding to particle-particle interactions. This paper will propose a framework for taking the general non-local Hamiltonian [1]

$$H = A \otimes \mathbb{1} + \mathbb{1} \otimes B + \sum_{ij} M_{ij}\, \sigma_i \otimes \sigma_j \tag{1}$$

and interspersing its non-local part's evolution with LO to achieve a desired U. It is only the non-local part of the Hamiltonian which must be studied, and through decomposition of this piece into its canonical form, and some insight into Lie Theory, we are able to formulate a computational approach [3]. Devising a scheme to generate a U from a collection of available operators and Hamiltonians has direct applicablity to a variety of problems, but cannot be done purely analytically on a massive scale, thus making a computational method necessary.

Adaptive Algorithm Theory

Adaptive algorithms attempt to mimic the process of evolution by the mechanisms of mutation and natural selection. A set of mathematical objects dubbed *organisms*, each containing some set of data, are subjected to tests against some metric depending on the organism's features **[■]** .
This is considered one *generation*, and the best organisms of each generation are bred together (by some deterministic algorithm) and mutated to compete in the next generation.
In preparation for performing a process like the one described, we take three major steps.

- To begin, the degrees of freedom of the system should be analyzed, as these will function as the dimensions of the search space. During this process, the degrees which are less pertinent to a solution's fidelity should be identified and their relative mutation amplitude and frequency altered accordingly.
- Next, the mutations assigned to each degree of freedom will be designed so as to best approach a (hopefully) global extrema in the search space. In general, more volatile mutations will be given to the more effective degrees of freedom, and all mutations will decrease in magnitude and frequency as the number of generations increases **[■]** .
- Finally, we need determine the method by which the best organisms of each generation are bred (combined in a way that may retain traits of both.). Note that there is no guarantee under any binary operation that the resulting object will have the same advantages assigned to the parent objects. In the case of well-defined and relatively smooth systems, this breeding will often resemble a sort of average, and the child will often perform similarly to the parents.

With these in mind, we can visualize the process as a sort of traversal of a complex multidimensional search space. This will give some heuristic for the arguments for convergence.

Figure 7: This is a simplification; the organism being tested is represented by the circle in blue, while the search-space (which has a defined metric represented by the grid) surround it. The paths of different color shown represent the non-deterministic paths possible as the organism mutates and is bred with other successful organisms. The actual space being searched is not two- but many-dimensional.

Arguments & Degrees of Freedom in the Test Organism

For the adaptive algorithm process described, each organism, denoted Θ, is given a set of arguments which define it and serve as its characteristics. For our purposes, these can be further divided into *constants* and *mutables*.
Among the constants, we have H, a set Hamiltonian that is easy to generate, and is used in every free evolution between LO. It does not change in each simulation, across all organisms, and across all generations. We also have N, which defines the number of LO-FE pairs for all organisms in a given simulation. None of these variables is subject to the mutation protocol that defines adaptive algorithms.

$$\underbrace{(A_1 \otimes B_1)e^{-iHt_1} \cdots (A_N \otimes B_N)e^{-iHt_N}}_{N} \tag{2}$$

The mutable data contains all of the sets. This includes the set of A_i, which are unitary operators on the first particle, the set of B_i, which are unitary operators on the second particle, and the set of t_i, which are the discrete durations for each FE, and have the constraint of summing to a particular value.
In any effective adaptive algorithm, the structure of the search space must be known. This allows both a functioning and efficient algorithm. In this scenario, the dimension of the space depends on the step number N in a linear fashion. We begin with a local operation, A, which is of the form

$$\beta \cdot \sigma = \begin{pmatrix} \beta_0 + \beta_3 & \beta_1 - i\beta_2 \\ \beta_1 + i\beta_2 & \beta_0 - \beta_3 \end{pmatrix} \tag{3}$$

This appears to have four degrees of freedom, one for each element of β. However, there is an implicit restraint given the unitarity of the LO, we may recognize this as an exponentiation of a linear combination of the pauli-matrices.

$$A^\dagger A = \mathbb{1} \implies A = \mathbb{1}\cos|\beta'| + (\beta' \cdot \sigma)\sin|\beta'| \tag{4}$$

And we recognize this form as being equivalent to the specification of a single point by a new vector β' on the three sphere. Thus A is also equivalent to

$$exp\{\beta' \cdot \sigma\} = exp\{\beta'_1\sigma_1 + \beta'_2\sigma_2 + \beta'_3\sigma_3\} \tag{5}$$

This gives six degrees of freedom for each pair of local operations. We also note that given variability in the discrete FE times, we would be given $N-1$ more degrees of freedom (the restriction is due to their sum being defined). All considered, we are searching a $6N$-dimensional space for every N-family.

Conclusion

References

[1] C.H. Bennett. Optimal simulation of two-qubit hamiltonians using general local operations. 2008.
[2] Michael J. Bremner. Practical scheme for quantum computation with any two-qubit entangling gate. *Physical Review Letters*, 89(24), 2002.
[3] Henry L. Haselgrove. Practicality of time-optimal two-qubit hamiltonian simulation. *Physical Review*, 2003.