A Survey of Optimization Algorithms to Minimize Quantum Gate Transformation Errors for Pulse Sequences Using Single Flux Quantum Pulses

By: Antoine Moats, University of Southern California
Mentor: Professor Rebing Wu, Tsinghua University

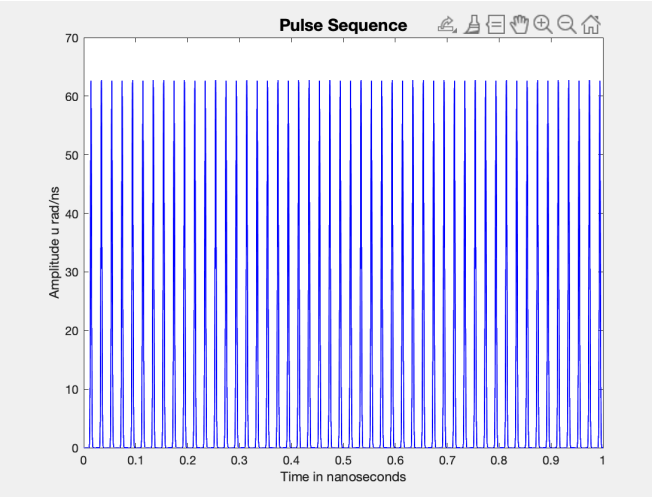
The purpose of this paper is to analyze the ability of different MATLAB search algorithms to find single flux quantum (SFQ) pulse sequences that minimize gate transformation errors which are defined throughout the literature[1,2]. Moreover, these algorithms were tested across different scenarios such as increasing the length of the pulse sequence, grouping the pulses together, and reducing the pulse area. The algorithms I compared were the built-in MATLAB optimization algorithms: single and multi-objective genetic algorithm (GA), surrogateopt (SO), particle swarm optimization (PSO), and simulated annealing (SA). In most of the simulations 4 eigenstates were simulated as this was the case in many papers in the literature on SFQ pulses[1,2].

Binary Pulse Sequences v.s. Segmented Pulse Sequence

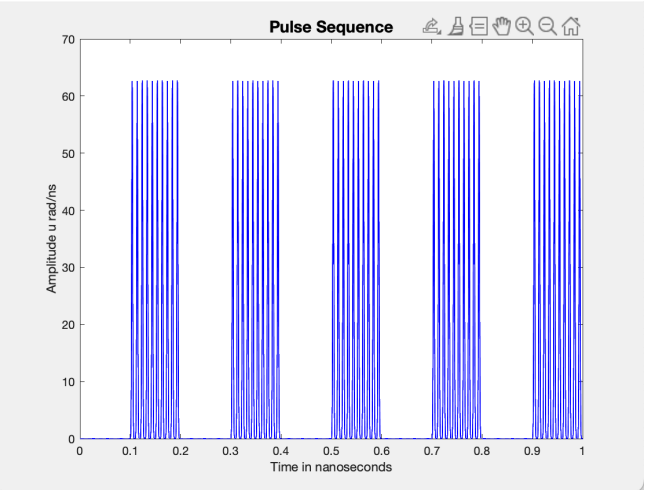
In order to encode the fact that the pulse sequence can only include values corresponding to On/Off, integer constraints need to be placed. In addition, the index of variables in MATLAB can not be less than 1. Therefore the free evolution (U_0) and applied pulse (U_1) operators are stored in a cell array in MATLAB as U{1} and U{2}. This limits the integers in the pulse sequence to be either 1 or 2. This was done for the GA, SO, PSA, and SA algorithms.

One potential issue that arises with binary sequences is the frequency of pulses matches the frequency of the excitation to higher order eigenstates causing leakage to occur. Leakage throughout the pulse sequence increases the likelihood that the gate error will be large. Therefore, in order to reduce the chance of leakage, grouping the pulses together may cause the gate error to be smaller. This can be done by changing the integer constraints from 1 to 2 to be from - L to L. Thus, the pulse sequence is made up of N segments where each segment is an integer between -L and L (where N and L are user inputs). A segment with negative length C (greater than or equal to L) corresponds to a segment of C pulses being Off and a segment with positive length C (less than or equal to L) corresponds to C pulses being On. An example illustrating the difference is shown below:

Alternating Binary Sequence of 100 Pulses



Alternating Pulse Sequence of 10 segments of 10 Pulses each



Binary Pulse Performance:

Algorithm	Error	Number of Eigenstates	Pulse Area	Size of Pulse Sequence
Genetic Algorithm	~10^{-2}- 10^{-4}	4	pi/20, pi/300	1000-4000
Surrogate Optimization	~10^{-1}	4	pi/20, pi/300	1000-4000
Simulated Annealing	~10^{-1}	4	pi/20, pi/300	1000-4000
Particle Swarm	~10^{-1}-10^{-3}	4	pi/20, pi/300	1000-4000

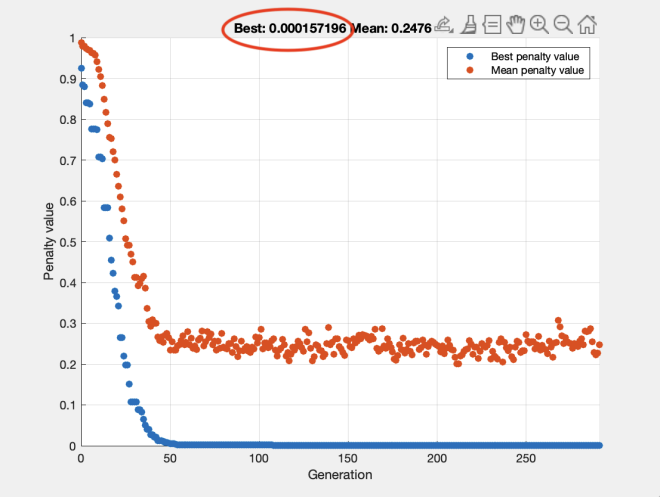
The best overall results were from the genetic algorithm when the pulse area was pi/300 and the total pulse length was made up of 4000 time steps.

Segmented Pulse Performance:

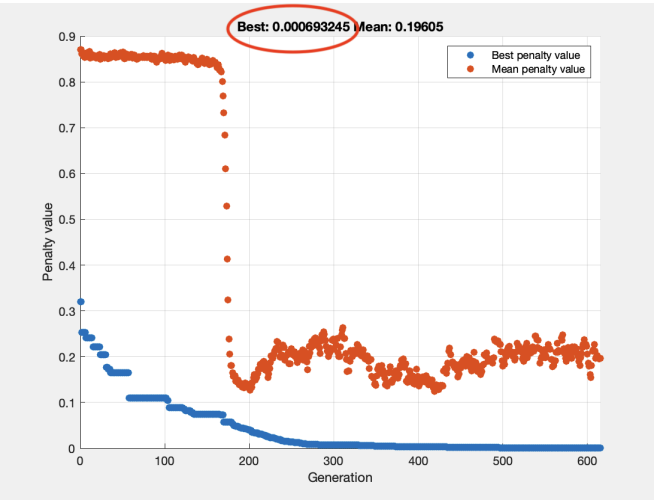
Algorithm	Error	Number of Eigenstates	Pulse Area	Number of Segments	Maximum Length of Segments
Genetic Algorithm	~10^{-2}- 10^{-4}	4	pi/20, pi/300	30-100	100
Surrogate Optimization	~10^{-2}	4	pi/20, pi/300	30-100	100
Simulated Annealing	~10^{-1}	4	pi/20, pi/300	30-100	100
Particle Swarm	~10^{-2}	4	pi/20, pi/300	30-100	100

The best overall results were seen from the genetic algorithm when the pulse area was pi/20 and the number of segments was 30 and the maximum length of the segments was 100. Overall the final gate error was comparable for all of the algorithms and was not significantly impacted by grouping the pulses into segments. An example of excellent results from the genetic algorithm run for the binary case and the segmented case is shown below:

Binary case of 4000 time steps and pulse width of pi/300



30 segments of maximum length 100 and pulse width of pi/20



Averaging the Error Over Unknown Number of Eigenstates

In the literature, the number of eigenstates involved in the calculation of gate error is usually 3 or 4. The reasoning for this is that the higher order eigenstates do not play a factor in leakage. In reality, the precise number of eigenstates is unknown unless a measurement is made in which case the superposition collapses and the quantum state is a pure state. Because the precise number of eigenstates is unknown it is important to consider the case where a pulse sequence can be optimized for an arbitrary number of eigenstates. This was done by encoding a lower bound and an upper bound of possible eigenstates. The algorithms would calculate the value of the error when the number of eigenstates was limited to the lower bound and increment the number of eigenstates by one until the upper bound was reached. The algorithms would then average the error across all of the number of possible eigenstates and find the pulse sequence that minimized this average.

Algorithm	Average Error	Lower bound-Upper Bound of eigenstates	Pulse Area	Size of Pulse Sequence
Genetic Algorithm	~10^{-3}	2-4	pi/300	1000-4000
Surrogate Optimization	~10^{-1}	2-4	pi/20, pi/300	1000-4000
Particle Swarm	~10^{-1}-10^{-3}	2-4	pi/300	1000-4000

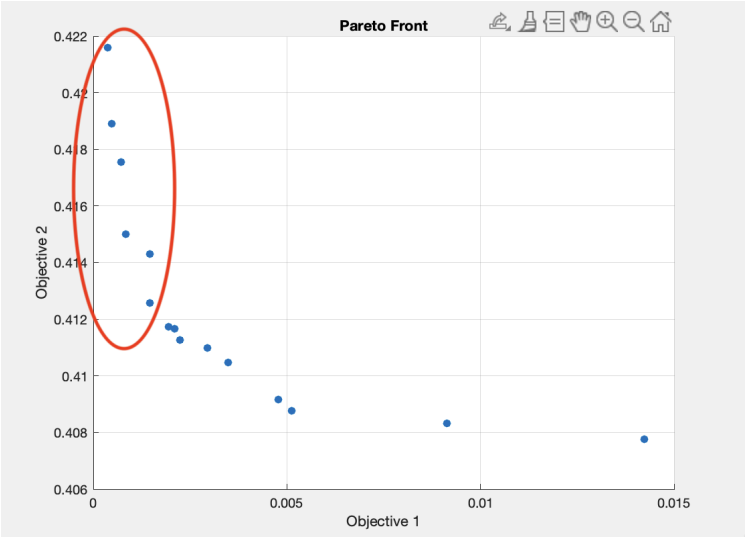
Due to the lack of performance of the simulated Annealing algorithm in MATLAB I did not include it in the above table. Addition, the number of eigenstates is capped at four because for higher upper bounds of the eigenstates the minimized errors became on the order of magnitude of 10^{-1} across all of the algorithms. This process illustrated that averaging the error for cases when the system was constrained to a range of eigenstates was not effective in finding pulse sequences that limited the error of a gate transformation. Moreover, a pulse sequence should be optimized when there is a known number of eigenstates. While knowing the exact number of potential leakage eigenstates of a system is impossible, as leakage to higher order eigenstates always occurs, it was an interesting find that contradicted some of the papers in the literature on SFQ pulses[1,2]. Several papers argued that a pulse sequence optimized for 3 or 4 eigenstates was sufficient for $n > 5$ eigenstates[1,2] . The argument centered around the fact that leakage to higher order eigenstates drops off in a nonlinear fashion as more eigenstates were added. While this intuitively made sense, my findings did not directly support this conclusion.

Single v.s. Multi-objective Sequences

Another idea was to calculate the average leakage probability over time and to find an algorithm that minimized the average leakage probability over time as well as the final gate error. Because there are functions being minimized, this is known as a multi-objective optimization algorithm. In MATLAB, the genetic algorithm is the only algorithm that can be utilized as a multi-objective algorithm. Using this new multi-objective function, the binary sequences were used to see if reduced values of final gate errors could be found.

Pareto Front of Multiobjective Genetic Algorithm for 4 eigenstates

The Pareto front illustrates values that minimize both objective functions in a way that any decrease in one objective would lead to an increase in the other. Here, the first objective function - the gate error - is of more significance than the second objective function - the average leakage probability for higher order eigenstates. Therefore the values on the left hand of the graph are the optimal solutions that are included in the table below.

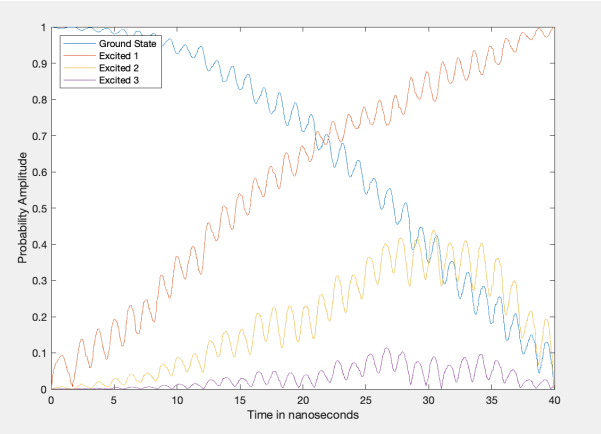


Algorithm	Final Gate Error	Number of Eigenstates	Pulse Area	Size of Pulse Sequence
Genetic Algorithm	~10^{-3}- 10^{-4}	4	pi/20	1000-4000
Multi-objective GA	~10^{-1}-10^{-2}	4	pi/20	1000-4000
Multi-objective GA	~10^{-2}- 10^{-4}	4	pi/300	1000-4000

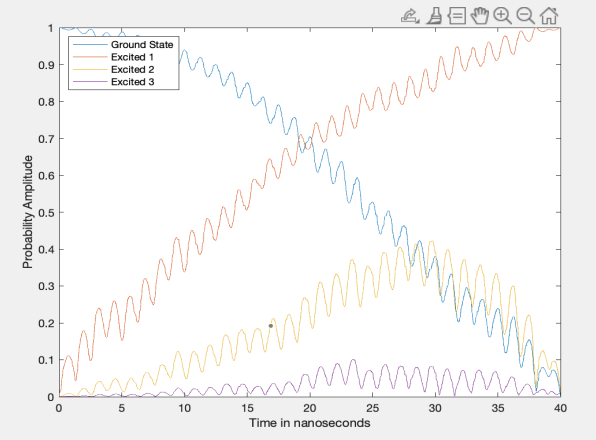
Discussion

The smallest gate errors were consistently found when the genetic algorithm was used. Across almost all of the parameters (varying pulse length, pulse area, number of eigenstates etc.) the genetic algorithm was the best. Moreover, the genetic algorithm provided values that were consistent with those found in the literature ~10^{-4} across each of these constraints as well. While low values were obtained for pi/20 and shorter pulse sequences, smaller pulse areas (pi/300 or pi/100) and longer pulse sequences are optimal for reducing leakage throughout the course of the pulse sequence. This can be seen from multi-objective genetic algorithm runs where the final gate error and the average probability leakage are minimized simultaneously and from single objective genetic algorithm runs. In the diagrams below, the probability for each state overtime is graphed, and even though the single objective GA does not explicitly minimize the probabilities of the second and third excited states, the average probabilities over time are minimized. This is not found in sequences with pi/20 pulse areas where the second and third eigenstates are “overexcited”, meaning leakage throughout the gate transformation is very high. Binary sequences are also more favorable to segmented pulse sequences because they consistently decrease the probabilities of the ground state and the first excited state. The segmented pulse sequences ended up sporadically changing the probabilities of the ground and excited states leading to leakage to higher order states. Reducing leakage to higher order states throughout the gate transformation is important because for high leakage sequences, any external disturbance would increase the likelihood that the pulse sequence did not effectively switch the probabilities of the two qubit states (the ground state and the first excited state).

GA binary single obj pi/300 and 4000 time steps



GA binary multi obj pi/300 and 4000 time steps



The two graphs are essentially the same; the average leakage probability is marginally reduced which can be deduced from the fact that the peaks of the 2nd and 3rd eigenstates are slightly smaller for the second pulse sequence.

Side Investigations

Bounded leakage proof:

In order to calculate the average leakage probability, an initial probability column vector must be inputted. This is due to the fact that for different starting probabilities, the leakage to higher order states is different. For example, a qubit that starts purely in the first excited state will have a different leakage to the second and third excited states than a qubit that starts purely in the ground state. Thus, I proved that the average probability leakage to higher order states is bounded by a starting probability of either the pure ground state or by a starting probability of the pure excited state. Once I did this, I was able to calculate which one was greater for a given pulse sequence in order to determine the highest average probability leakage for that pulse sequence. This allowed the comparison of the average probability leakage used in the second objective function for the multi-objective genetic algorithm to be simple as the initial probability is no longer a variable. Instead, each pulse sequence is determined to be optimal based on the pulse sequence that has the lowest value of the highest average probability leakage.

Genetic Algorithm settings:

In MATLAB, the Genetic Algorithm takes in several outputs allowing the user to "tune" the algorithm minimizing the errors. After trial and error and reading papers[1,3] that also utilized the genetic algorithm, the following settings produced the best results.

- 'CrossoverFcn' , crossoverFunction
- 'crossoverFraction', 0.9
- 'EliteCount', 25
- 'FitnessScalingFcn', 'fitscalingshiftlinear'
- 'PopulationSize', 1000
- "MaxGenerations", 1000

References

1. Optimal Qubit Control Using Single-Flux Quantum Pulses
   Per J. Liebermann and Frank K. Wilhelm
   https://arxiv.org/pdf/1512.05495.pdf

2. Binary Optimal Control Of Single-Flux-Quantum Pulse Sequences
   Ryan H. Vogta, N. Anders Petersson
   https://arxiv.org/pdf/2106.10329.pdf

3. Tuning Genetic Algorithm Parameters using Design of Experiments
   Mohsen Mosayebi and Manbir Sodhi
   http://www0.cs.ucl.ac.uk/staff/W.Langdon/gecco2020/companion_files/wksp176s2-file1.pdf