Work Summary

Further result for fock state transfer and some obvservations from literature review. Also, I have taken a more detailed look at the possibility of changing the definitions of fidelity in QuTip.

Main Work

0. Derivative for eigenvectors and eigenvalues
1. QuTip implementation of propagator gradient for GRAPE
2. Modifications on the fidelity computation function
3. Results

Gradient for eigenvectors and eigenvalues

Let Hermitian matrix $\hat{A}$ have eigenvalues λ_i and corresponding eigenvectors $\vec{v_i}$ such that $\hat{A}\vec{v_i} = \lambda_i\vec{v_i}$. The derivative of the eigenvectors and eigenvalues can be expressed using the derivative of the matrix $\hat{A}'$ as shown in Equation 1 and 2. Here, it is assumed that $\vec{v_i}$ are orthonormal vectors, and thus $\vec{v_i}'$ are perpendicular to $\vec{v_i}$.

$$\begin{aligned}
\vec{v_i}^\dagger \left(\hat{A}\vec{v_i}\right)' &= \vec{v_i}^\dagger \left(\lambda_i\vec{v_i}\right)' \\
\vec{v_i}^\dagger \left(\hat{A}'\vec{v_i} + \hat{A}\vec{v_i'}\right) &= \vec{v_i}^\dagger \left(\lambda_i'\vec{v_i} + \lambda_i\vec{v_i}'\right) \\
\vec{v_i}^\dagger\hat{A}'\vec{v_i} + \lambda_i\vec{v_i}^\dagger\vec{v_i'} &= \lambda_i'\vec{v_i}^\dagger\vec{v_i} + \lambda_i\vec{v_i}^\dagger\vec{v_i}' \\
\lambda_i' &= \vec{v_i}^\dagger\hat{A}'\vec{v_i} \qquad (1)
\end{aligned}$$

Using the fact that $\vec{v_i}$ forms a complete basis.

$$\begin{aligned}
\vec{v_j}^\dagger \left(\hat{A}\vec{v_i}\right)' &= \vec{v_j}^\dagger \left(\lambda_i\vec{v_i}\right)' \\
\vec{v_j}^\dagger \left(\hat{A}'\vec{v_i} + \hat{A}\vec{v_i'}\right) &= \vec{v_j}^\dagger \left(\lambda_i'\vec{v_i} + \lambda_i\vec{v_i}'\right) \\
\vec{v_j}^\dagger\hat{A}'\vec{v_i} + \lambda_j\vec{v_j}^\dagger\vec{v_i'} &= \lambda_i'\vec{v_j}^\dagger\vec{v_i} + \lambda_i\vec{v_j}^\dagger\vec{v_i}' \\
\vec{v_j}^\dagger\hat{A}'\vec{v_i} &= \left(\lambda_i - \lambda_j\right)\vec{v_j}^\dagger\vec{v_i}' \\
\vec{v_i}' &= \sum_{j\neq i}\frac{\vec{v_j}^\dagger\hat{A}'\vec{v_i}}{\lambda_i - \lambda_j}\vec{v_j} \qquad (2)
\end{aligned}$$

1 QuTip implementation of propagator gradient for GRAPE

This is to express how QuTip implements propagator gradient for GRAPE. Following the conventions set by the report sent out last week. Assume the truncated Hilbert space to have dimension N_{cavity}, the total time slice number to be n, and the duration of each time slice is Δt. The total Hilbert space of the qubit and harmonic oscillator has dimension $N = N_{cavity} \times N_{qubit} = 2N_{cavity}$. The Hamiltonian can be expressed as

$$\hat{H}\left(t_i\right) = \hat{H}_{drift}\left(t_i\right) + \sum_j \epsilon_j\left(t_i\right)\hat{H}_j$$

, where ϵ_i is the control pulse amplitude subject to optimization. The pulse amplitude ϵ_i is constant in each time slice t_i to $t_i + \Delta t$.

Here discuss the case when the target is an unitary operator, but it can be easily generalized to include target state. Assume the target operator is $\hat{U}_{targ}$. Assume the propagator for time slice i be $\hat{A}_i$, where i range from 1 to n. The initial operator is $\hat{U}_0$, which is usually set to the identity operator. The resulting operator from the Hamiltonian is

$$\hat{U}_f = \hat{A}_n \cdots \hat{A}_2\hat{A}_1\hat{U}_0$$

. $\hat{A}_i$ is calculated through decomposition. Let the eigen-basis decomposition of the total Hamiltonian be

$$\hat{H}\left(t_i\right) = \hat{V}\hat{D}\hat{V}^\dagger, \text{ where } \hat{V} = \begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_N\end{bmatrix} \text{ and } \hat{D} = \begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_N\end{bmatrix}$$

. Then, the propagator $\hat{A}_i = e^{-i\Delta t\hat{H}(t_i)} = \hat{V}e^{-i\Delta tD}\hat{V}^\dagger$.

The default fidelity for unitary evolution is the Hilbert-Schmidt inner product

$$f_{HS} = \frac{Tr\left\{\hat{U}_{targ}^{-1}\hat{U}_f\right\}}{N} = |f_{HS}|e^{i\theta}$$

. When global phase is ignored, the fidelity is taken to be $f = |f_{HS}|$, which range from 0 to 1 when it's unitary evolution. The derivative can be calculated as expressed in Equation 3.

$$\begin{aligned}
\frac{\partial f}{\partial \epsilon_i} &= \frac{\partial\sqrt{f_{HS}^* f_{HS}}}{\partial \epsilon_i} \\
&= \frac{1}{2}\left(\sqrt{\frac{f_{HS}^*}{f_{HS}}}\frac{\partial f_{HS}}{\partial \epsilon_i} + \sqrt{\frac{f_{HS}}{f_{HS}^*}}\frac{\partial f_{HS}^*}{\partial \epsilon_i}\right) \\
&= \frac{1}{2}\left(e^{-i\theta}\frac{\partial f_{HS}}{\partial \epsilon_i} + e^{i\theta}\frac{\partial f_{HS}^*}{\partial \epsilon_i}\right) \\
&= \frac{1}{2}Re\left\{e^{-i\theta}\frac{\partial f_{HS}}{\partial \epsilon_i}\right\} \\
&= \frac{1}{2}Re\left\{\frac{e^{-i\theta}}{N}\frac{\partial Tr\left\{\hat{U}_{targ}^{-1}\hat{U}_f\right\}}{\partial \epsilon_i}\right\} \\
&= \frac{1}{2}Re\left\{\frac{e^{-i\theta}}{N}Tr\left\{\hat{U}_{targ}^{-1}\hat{A}_n \cdots \frac{\partial \hat{A}_i}{\partial \epsilon_i} \cdots \hat{A}_1\hat{U}_0\right\}\right\} \qquad (3)
\end{aligned}$$

The propagator gradient is then calculated as expressed in Equation 4.

$$
\begin{aligned}
\frac{\partial \hat{A}_i}{\partial \epsilon_i} &= \frac{\partial\left(\hat{V}e^{-i\Delta t D}\hat{V}^\dagger\right)}{\partial \epsilon_i} \\
&= \frac{\partial \hat{V}}{\partial \epsilon_i}e^{-i\Delta t D}\hat{V}^\dagger + \hat{V}\frac{\partial e^{-i\Delta t D}}{\partial \epsilon_i}\hat{V}^\dagger + \hat{V}e^{-i\Delta t D}\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i} \\
&= \hat{V}\left(\hat{V}^\dagger\frac{\partial \hat{V}}{\partial \epsilon_i}e^{-i\Delta t D} + \frac{\partial e^{-i\Delta t D}}{\partial \epsilon_i} + e^{-i\Delta t D}\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i}\hat{V}\right)\hat{V}^\dagger \\
&= \hat{V}\hat{B}\hat{V}^\dagger, \qquad (4) \\
\text{where } \hat{B} &= \hat{V}^\dagger\frac{\partial \hat{V}}{\partial \epsilon_i}e^{-i\Delta t D} + \frac{\partial e^{-i\Delta t D}}{\partial \epsilon_i} + e^{-i\Delta t D}\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i}\hat{V}
\end{aligned}
$$

Here, using Equation 2, observe that

$$
\left(\hat{V}^\dagger\frac{\partial \hat{V}}{\partial \epsilon_i}\right)_{l,m} = \vec{v_l}^\dagger\frac{\partial \vec{v_m}}{\partial \epsilon_i} = \left(1-\delta_{l,m}\right)\frac{\vec{v_l}^\dagger\frac{\partial \hat{H}}{\partial \epsilon_i}\vec{v_m}}{\lambda_m-\lambda_l} = \left(1-\delta_{l,m}\right)\frac{\vec{v_l}^\dagger\hat{H}_i\vec{v_m}}{\lambda_m-\lambda_l}
$$

amd similarly

$$
\left(\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i}\hat{V}\right)_{l,m} = \left(\frac{\partial \vec{v_l}}{\partial \epsilon_i}\right)^\dagger\vec{v_m} = \left(1-\delta_{l,m}\right)\frac{\vec{v_l}^\dagger\hat{H}_i\vec{v_m}}{\lambda_l-\lambda_m} = -\left(\hat{V}^\dagger\frac{\partial \hat{V}}{\partial \epsilon_i}\right)_{l,m}.
$$

Also, applying Equation 1 gives

$$
\left(\frac{\partial e^{-i\Delta t D}}{\partial \epsilon_i}\right)_{l,m} = \delta_{l,m}\left(-i\Delta t\right)e^{-i\Delta t\lambda_l}\frac{\partial \lambda_l}{\partial \epsilon_i} = \delta_{l,m}\left(-i\Delta t\right)e^{-i\Delta t\lambda_l}\left(\vec{v}_l^\dagger\hat{H}_i\vec{v}_l^\dagger\right)
$$

Thus, given that $\hat{D}$ is diagonal with entries λ_i, the entries of $\hat{B}$ can be expressed as below,

$$
\begin{aligned}
B_{l,m} &= \left(\hat{V}^\dagger\frac{\partial \hat{V}}{\partial \epsilon_i}\right)_{l,m}e^{-i\Delta t\lambda_m} + \left(\frac{\partial e^{-i\Delta t D}}{\partial \epsilon_i}\right)_{l,m} + e^{-i\Delta t\lambda_l}\left(\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i}\hat{V}\right)_{l,m} \\
&= \left(e^{-i\Delta t\lambda_l}-e^{-i\Delta t\lambda_m}\right)\left(\frac{\partial \hat{V}^\dagger}{\partial \epsilon_i}\hat{V}\right)_{l,m} + \delta_{l,m}\left(-i\Delta t\right)e^{-i\Delta t\lambda_l}\left(\vec{v}_l^\dagger\hat{H}_i\vec{v}_l^\dagger\right) \\
&= \left(1-\delta_{l,m}\right)\left(e^{-i\Delta t\lambda_l}-e^{-i\Delta t\lambda_m}\right)\frac{\vec{v_l}^\dagger\hat{H}_i\vec{v_m}}{\lambda_l-\lambda_m} + \delta_{l,m}\left(-i\Delta t\right)e^{-i\Delta t\lambda_l}\left(\vec{v}_l^\dagger\hat{H}_i\vec{v}_l^\dagger\right). \qquad (5)
\end{aligned}
$$

Expressing with the conventions set in QuTip, $B_{l,m} = Q_{l,m}W_{l,m}$, i.e. it's the element-wise product of W and Q, where $\hat{W} = \hat{V}^\dagger\hat{H}_i\hat{V}$ and $Q_{l,m} = \left(1-\delta_{l,m}\right)\frac{e^{-i\Delta t\lambda_l}-e^{-i\Delta t\lambda_m}}{\lambda_l-\lambda_m} + \delta_{l,m}\left(-i\Delta t\right)e^{-i\Delta t\lambda_l}$.

Combining Equations 3, 4, and 5 gives the gradient of fidelity with respect to each control ϵ_i in each time slice.

2 Modifications on the fidelity computation function

The fidelity is defined as shown above by the Hilbert-Schmidt inner product. The evolution unitary operator is a $N \times N$ dimension matrix. The target unitary can be expressed as

$$
\hat{U}_{targ} = e^{-i\hat{H}t}\otimes\hat{P}_{gg} + \hat{U}_{ee}\otimes\hat{P}_{ee} + \hat{U}_{ge}\otimes\hat{P}_{ge} + \hat{U}_{eg}\otimes\hat{P}_{eg}
$$

, where $\hat{P}_g = |g\rangle\langle g|$ and similarly for the other state projectors. Normally, $\hat{U}_{ge}$ and $\hat{U}_{eg}$ are set to zero. The fidelity calculated by the Hilbert-Schmidt inner product takes into account the "distance" of all entries of the evolution unitary operator to the target operator. However, in practice only a small portion of the matrix matters. The modification to the fidelity computation can come from two parts.

2.1 Possibile modification 1

The simulated dynamics should correspond to the system when the qubit starts and ends in the ground state. Therefore, $\hat{U}_{ee}$, $\hat{U}_{ge}$, and $\hat{U}_{eg}$ are all not in our concern. Therefore, implement the modifications corresponding to the reasons stated above. The first modification to the fidelity can be

$$f_1 = \frac{Tr\left\{\hat{U}^\dagger_{ground}\left(\hat{U}^{-1}_{targ}\hat{U}_f\right)\hat{U}_{ground}\right\}}{N_{cavity}}, \text{ where } \hat{U}_{ground} = \hat{I}_{cavity} \otimes |g\rangle$$

. Here, when target $\hat{U}_{ge}$ and $\hat{U}_{eg}$ are zero, it is equivalent to only setting the target unitary for the cavity $\hat{U}_{gg}$ in ground state to be $e^{-i\hat{H}t}$, i.e.

$$f_1 = \frac{Tr\left\{\left(e^{-i\hat{H}t}\right)^{-1}\hat{U}_{f,gg}\right\}}{N_{cavity}}.$$

2.2 Possibile modification 2

It is defined that

$$\hat{H} = \begin{bmatrix} \hat{H}_{hop} & \mathbf{0} \\ \mathbf{0} & \hat{H}' \end{bmatrix}$$

. Here, for 2 electron hopping on 4 sites, $\hat{H}_{hop}$ is a 6×6 matrix, and in a chosen basis,

$$\hat{H}_{hop} = -J_1 \begin{pmatrix} 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \end{pmatrix}.$$

When doing simulations, the initial and final states of the harmonic oscillator are expected to be within the Hilbert space of the first 6 energy levels such that it can be mapped to the electron dynamics. Then, the Hamiltonian $\hat{H}'$ is not important and doesn't impact the simulation fidelity. The resulting unitary is

$$e^{-i\hat{H}t} = \begin{bmatrix} e^{-i\hat{H}_{hop}t} & \mathbf{0} \\ \mathbf{0} & e^{-i\hat{H}'t} \end{bmatrix}$$

Taking into account the first modification, the second modification can be implemented through

$$f_2 = \frac{Tr\left\{\hat{I}_{6,g}\hat{U}^{-1}_{targ}\hat{I}_{6,g}\hat{U}_f\right\}}{6}, \text{ where } \hat{I}_{6,g} = \hat{I}_{6,cav} \otimes |g\rangle\langle g|$$

and here $\hat{I}_{6,cav} = \begin{bmatrix} \hat{I} & \\ & \mathbf{0} \end{bmatrix}$ with the 6×6 identity matrix $\hat{I}$.

This should be equivalent to

$$f_2 = \frac{Tr\left\{ \left(e^{-i\hat{H_{hop}t}} \right)^{-1} \hat{U}_{f,6\otimes gg} \right\}}{6},$$

where $\hat{U}_{f,6\otimes gg}$ represents the final evolution unitary's first 6×6 entries when it's mapped to the ground state projector operator $\hat{P}_{gg}$

2.3 Potential flaws

For the first modification f_1, the final optimized unitary operator can have non-zero $\hat{U}_{ge}$ and $\hat{U}_{eg}$, which are not taken into the fidelity formula. For example, assume the pulses are optimized for a time length of Δt. When the target evolution time is $2\Delta t$, then needs to apply the pulses twice. Only considering the evolution projected on $\hat{P}_{gg}$, then the target unitary for the cavity is $e^{-i\hat{H}(2\Delta t)}$. However, the evolution unitary driven by applying the pulse sequences twice is $(\hat{U}_{gg})^2 + \hat{U}_{ge}\hat{U}_{eg}$. As higher order term appears, this deviation will be larger as the number of times the pulse sequences are applied increases.

For the second modification, besides the flaw mentioned in the previous paragraph, the fidelity doesn't take into consideration of the $\mathbf{0}$ matrices in the upper left and lower right entries of the unitary $e^{-i\hat{H}t}$.

3 Results

This section discusses the conventions and the effect of some parameters to the fidelity. The results are shown for both cases of fidelity f_{HS} and f_1. In the plots, f_{HS} is called the full fidelity and f_1 is the ground state fidelity. f_2 hasn't been implemented yet. The results are mostly ran using Princeton's Adroit cluster. I haven't implemented parallel computing yet, so it is only around 2-4 times faster than running on my laptop.

3.1 Conventions

In this section, the results are obtained under some conventions as detailed below. The drift Hamiltonian is $\hat{H}_{int} = \chi \hat{a}^\dagger \hat{a} \otimes \hat{P}_{ee}$, where $\chi = 2\pi \times 2.2$MHz, i.e. having period $\approx 0.5\mu$s. The target Hamiltonian is set to

$$\hat{U}_{targ} = \hat{U}_{gg} \otimes \hat{P}_{gg} + \hat{U}_{ee} \otimes \hat{P}_{ee} = e^{-i\hat{H}t} \otimes \hat{P}_{gg} + e^{-i\chi\hat{a}^\dagger\hat{a}t} \otimes \hat{P}_{ee}$$

. When using the fidelity expression f_1, $\hat{U}_{ee}$ can be any unitary, but choosing it to be $e^{-i\chi\hat{a}^\dagger\hat{a}t}$ converges easier when using f_{HS} and hence easier to make comparisons.

The initial pulse shapes are all set to random with offset zero. The truncated Hilbert space of the cavity is set to dimension $N = 20$. Without specification, the pulse length that is being optimized is set to 2μs.

As we discussed last meeting, the magnitude of the hopping Hamiltonian has eigenvalues from $-2J$ to $2J$. Thus, let $4J \approx \chi/10$.

3.2 Effect of iterations

The number of iterations is the maximum limit set for the number of updates the pulse sequence will go through before stopping. When the algorithm stops because of iteration number limit reached, the time spent is proportional to the number of iterations as shown in Figure 1.

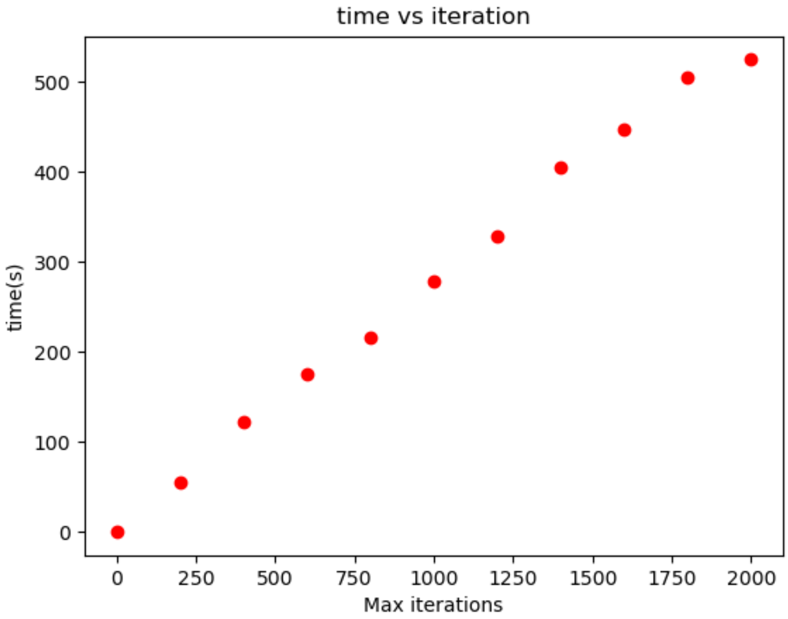

Figure 1: Run time vs iteration number

The fidelity error is plotted against the max iteration number in Figure 2 and 3. Both ground state fidelity and full fidelity are calculated and labelled in the graphs, but Figure 2 and 3 uses f_{HS} and f_1 in the algorithm's definition of fidelity.

The results are as expected. f_{HS} can be seen as 'including' f_1, so in Figure 2, where f_{HS} is being optimized, f_1 is decreasing as well. On the contrary, in Figure 3, f_1 is being optimized, while f_{HS} doesn't improve significantly. Worth noting that f_1 matters more than f_{HS} as explained before. With 2000 iterations, f_1 is 97.4% in Figure 2 and 99.5% in Figure 2, indicating a better performance in the modified fidelity expression.

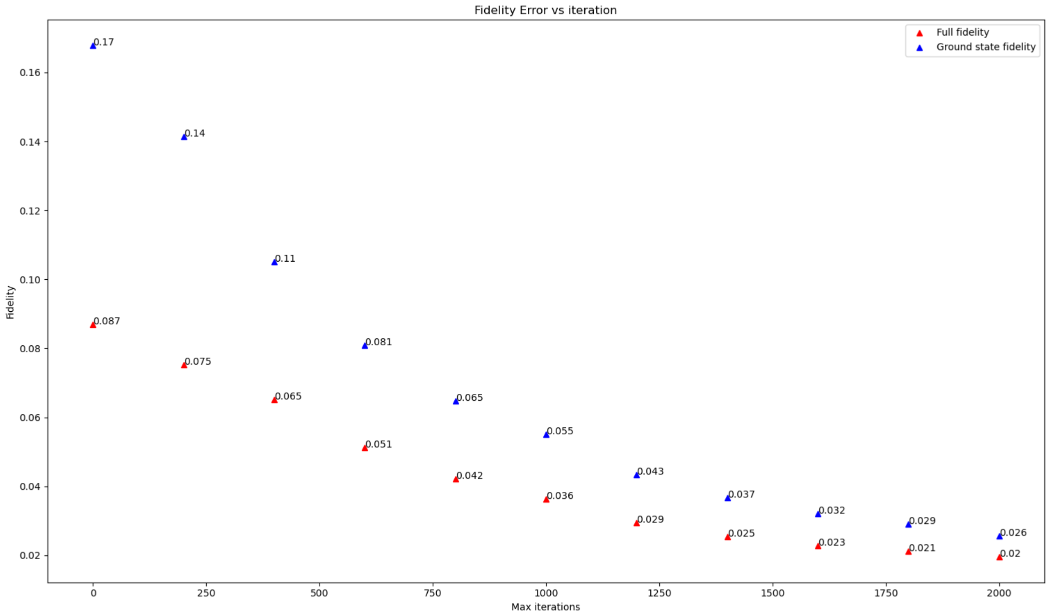

Figure 2: Fidelity error vs iteration number. Fidelity calculation in algorithm using f_{HS}.

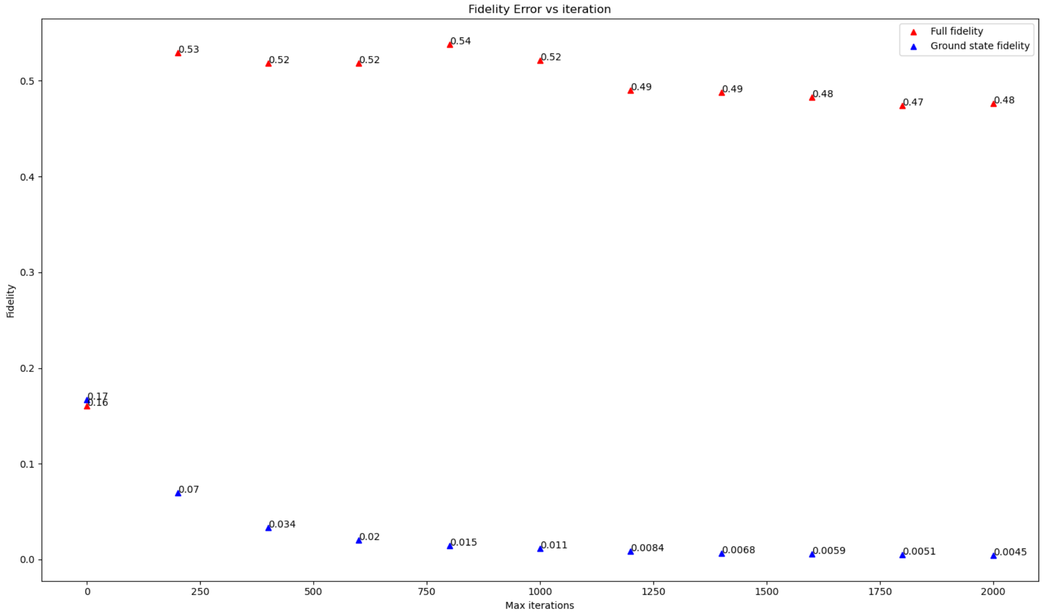

Figure 3: Fidelity error vs iteration number. Fidelity calculation in algorithm using f_1.

3.3 Effect of pulse scaling

Pulse scaling refers to the scaling of the random initial pulse. If the scaling is A, the pulse amplitude for each time slice is randomly chosen from $[-A, A]$. In Figure 4 and 5 I have plotted the fidelity error vs the initial pulse scaling when the algorithm using fidelity definition f_{HS} and f_1 respectively. My expectation was that there are peaks of fidelity at pulse scaling which is closer to the target unitary. However, neither of the plots show that behavior.

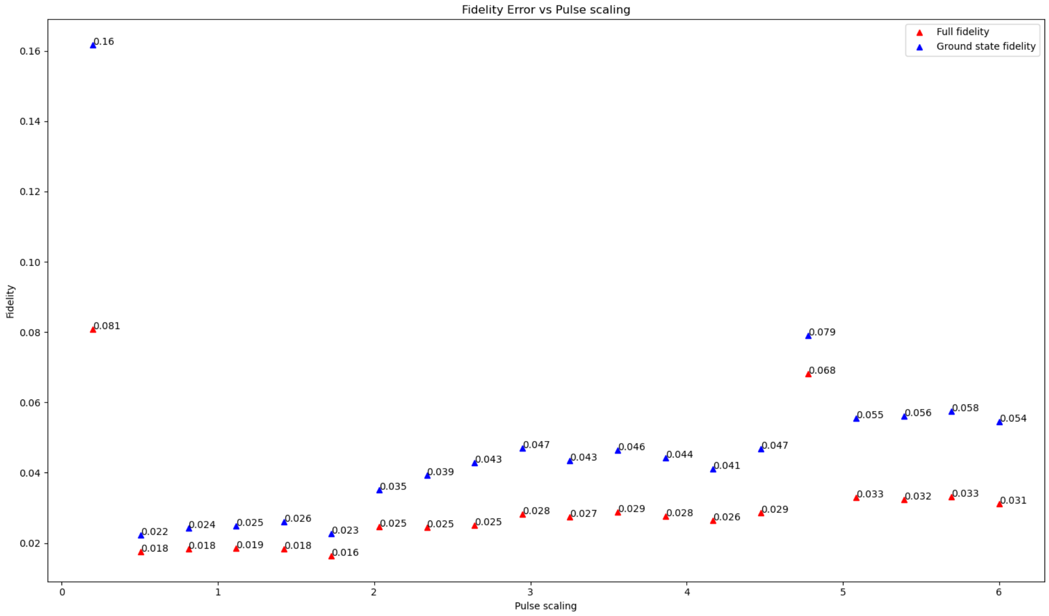

Figure 4: Fidelity error vs initial pulse scaling. Fidelity calculation in algorithm using f_{HS}.

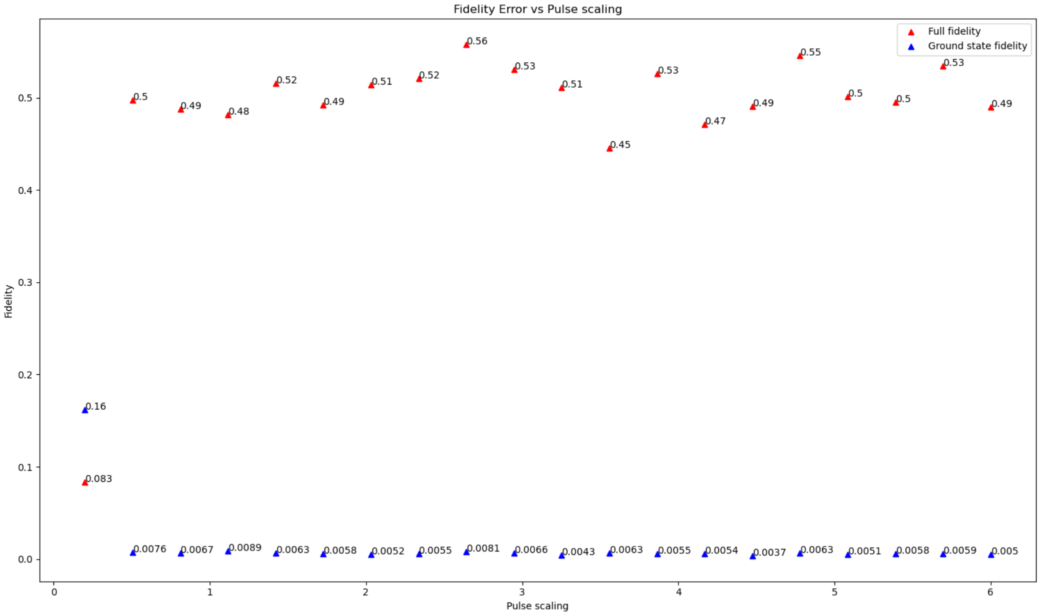

Figure 5: Fidelity error vs initial pulse scaling. Fidelity calculation in algorithm using f_1.

3.4 Effect of evolution time

Evolution time set to the algorithm is equivalent to the length of the pulse subject to optimization. Assume the target total evolution time to be $T = 10\mu$s such that the electron state changes effectively. There is the choice to set the pulses to length T/n, where n is any integer. The algorithm outputs the final optimized unitary driven by the pulse sequences U_f. Define the single run fidelity to be the fidelity comparing U_f and $U_{targ}(t = T/n)$, and define the evolution fidelity to be the one comparing $(U_f)^n$ and $U_{targ}(t = T)$. When the different pulse lengths are applied, the time duration of each time slice, where the pulse stays constant, is 10ns.

Figure 6, 7, 8, and 9 plots the single/evolution fidelity error vs the number of pulse sequence that needs to be applied to reach T. As usual, the scenarios are calculated for both cases when the GRAPE algorithm applys the fidelity expression f_{HS} and f_1.

In general, the results are within expectation. For single run fidelity, the distance between the target unitary and the initial unitary increases with pulse length, but the number of time slices (degrees of freedom) also increases, so it's hard to predict the pattern of single run fidelity with respect to the pulse length. For evolution fidelity, it is expected that the fewer pulse sequences need to be applied, the higher the fidelity is. This is because each pulse sequence is optimized to end at specific points such as ground state for the qubit, so applying multiple times the pulse sequence is like setting up more constraints. Moreover, as mentioned before, the non-zero values of $\hat{U}_{ge}$ and $\hat{U}_{eg}$ for a single pulse sequence induces deviations in the evolution operator.

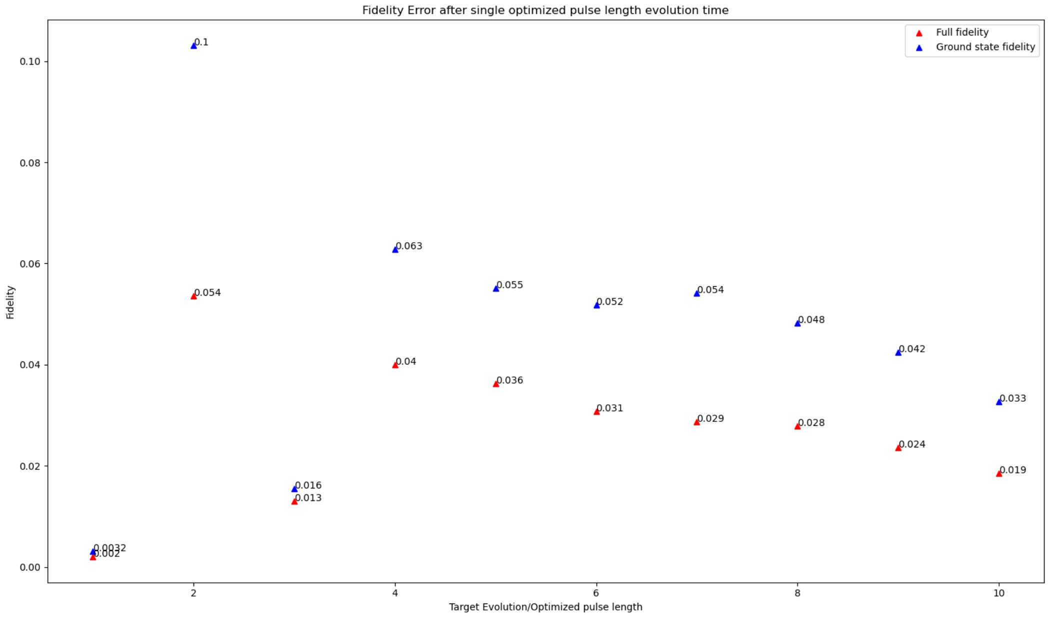

Figure 6: Single run fidelity error vs $n = T/t$. Fidelity calculation in algorithm using f_{HS}.

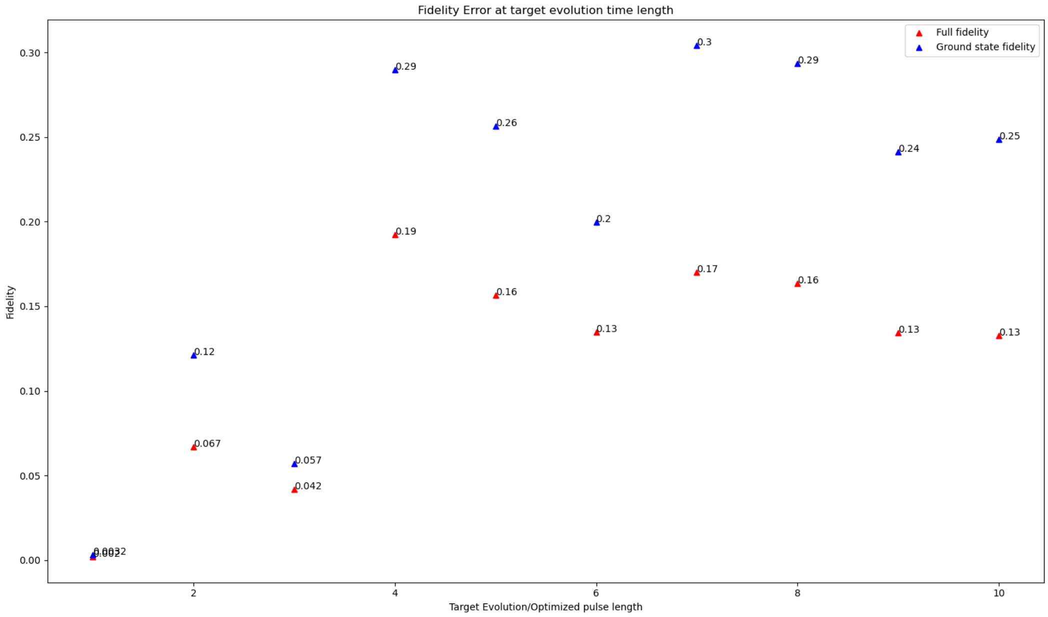

Figure 7: Evolution fidelity error vs $n = T/t$. Fidelity calculation in algorithm using f_{HS}.

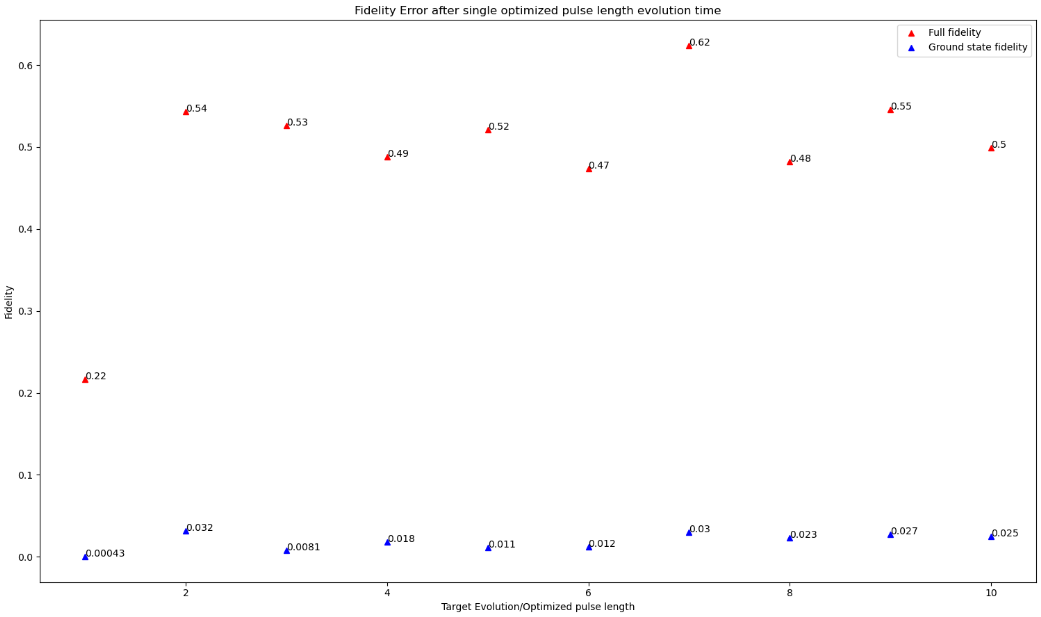

Figure 8: Single run fidelity error vs $n = T/t$. Fidelity calculation in algorithm using f_1.

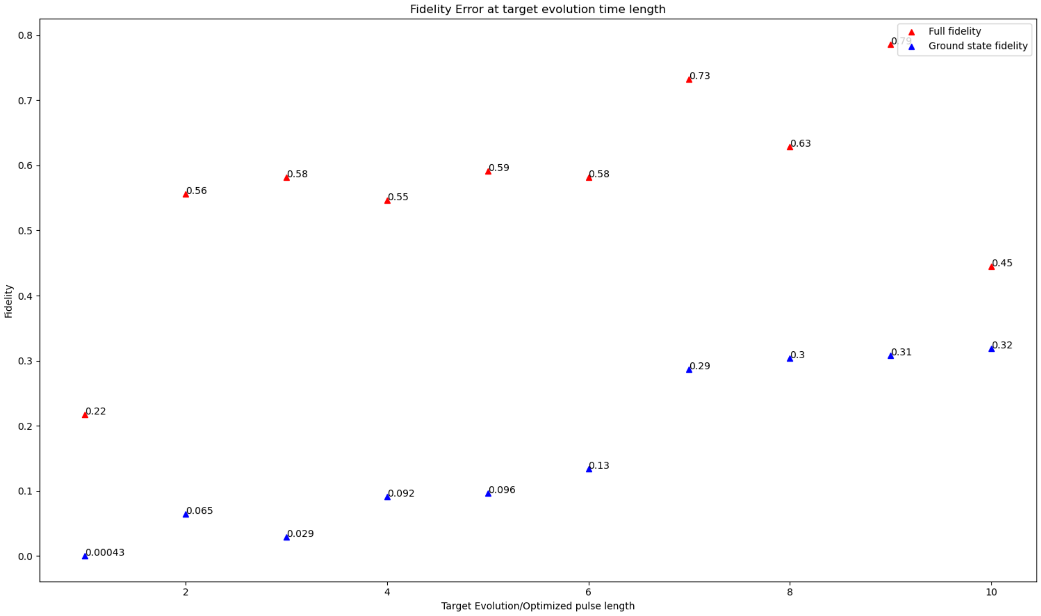

Figure 9: Evolution fidelity error vs $n = T/t$. Fidelity calculation in algorithm using f_1.