QC Architecture and Electronics (CESE4080) Homework 6

Tongyun Yang (5651794, tongyunyang@tudelft.nl)

2023.03.31

1 Exercise A

1.1 Question

Build a system to drive a π-rotation on Q1 while keeping Q2 idle and ensuring a fidelity above 99.9% for both qubits. Minimize the number of bits (N) of the DAC(s).

Answer: In short, the number of bits of the DACs are minimized to 5 bits each.

With the idea and the structure from figure 1. Signals I & Q of frequency spectrum 0.1GHz are generated respectively using the code below. The original idea was to use an envelop signal which follows the Gaussian distribution to generate the baseband signals, however, it is not listed in the available components, hence the idea was abandoned.

The result of the qubit spin is as shown in figure 2. The fidelity of both qubits are above 99.9% at the same time. For more detail information, please refer to the code in file "HW6_NEW.m". Note that there might be several existing versions which has similar names in the corresponding folder, please refer to the file which is mentioned above.

In the implementation, two DACs are introduced, they sample I signal and Q signal respectively, then they each mixed with either cos or sin signal. In the end, the output of the mixed signals are added together, and a gain of $\frac{1}{sinc(\frac{fin(1)}{fsample})}$ is introduced as well. Note that the question did not specify how the π rotation should be realized, hence I defined it as a π rotation around the x-axis.

```
I = A(1)*cos(2*pi*fin(1)*t+ph_in(1)) + A(2)*cos(2*pi*fin(2)*t+ph_in(2));
Q = A(1)*sin(2*pi*fin(1)*t+ph_in(1)) + A(2)*sin(2*pi*fin(2)*t+ph_in(2));

%sampling the input signal I
I_sampled=I(1:1/(fsample*dt):end);

%sampling the input signal Q
Q_sampled=Q(1:1/(fsample*dt):end);

%quantize the signal on N-bit
%first, scale the signal to fit in [0:2^N-1] range, then round it
D_I=round((I_sampled-min(I))/(max(I)-min(I))*(2^N-1));
D_Q=round((Q_sampled-min(Q))/(max(Q)-min(Q))*(2^N-1));

%DAC output range
DAC_or=1; %[V]
%DAC step
LSB=DAC_or/(2^N-1);
%define DAC array
DAC_I=LSB*[0 ones(1,2^N-1)];
output_levels_I=cumsum(DAC_I);

DAC_Q=LSB*[0 ones(1,2^N-1)];
output_levels_Q=cumsum(DAC_Q);

VDAC_I=output_levels_I(D_I+1);
VDAC_Q=output_levels_Q(D_Q+1);
```

2 Exercise B

2.1 Question

In the signal gain, you need a factor 1/sinc(fin(1)/fsample). Which component in the system introduces the (inverse) factor sinc(fin(1)/fsample)? What is causing this?

Answer: The factor $sinc(\frac{fin(1)}{fsample})$ in the signal gain is introduced by the process of sampling a continuous signal.

When a continuous-time signal is sampled at a rate of fsample, the resulting discrete-time signal is given by $x[n] = x(t)|_{t=nT}$, where $T = \frac{1}{fsample}$ is the sampling period. The Fourier transform of x[n] is the periodic function $X(f) = \frac{1}{T} \times sum_k \times X(f - kf_s)$, where $f_s = \frac{1}{T}$ is the sampling frequency

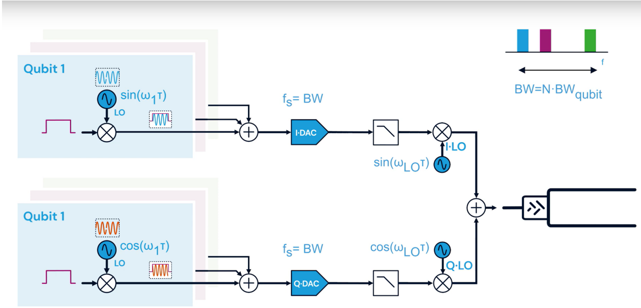

Figure 1: FDMA Method

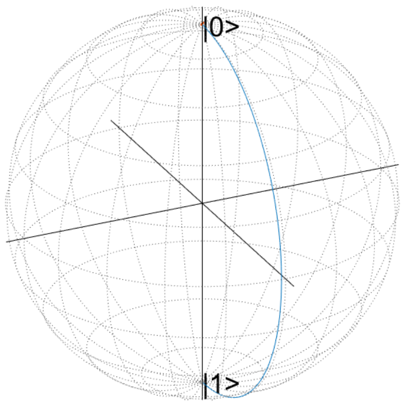

Figure 2: Qubit 1 rotate π, and qubit 2 remain idle

and sum_k denotes a summation over all integer values of k. The frequency domain representation of the sampled signal shows that the original continuous-time signal is replicated periodically in the frequency domain with a period of f_s.

To avoid aliasing, the continuous-time signal must be band-limited to below half the sampling frequency, $\frac{f_s}{2}$, before sampling. This is achieved by passing the continuous-time signal through an analog anti-aliasing filter with a cutoff frequency below $\frac{f_s}{2}$. The transfer function of the anti-aliasing filter is typically a low-pass filter with a cutoff frequency below $\frac{f_s}{2}$.

The anti-aliasing filter introduces a frequency-dependent gain factor that is equal to the inverse of the filter's frequency response, which is given by the sinc function $sinc(\frac{fin(1)}{fsample})$. Therefore, the factor is introduced by the anti-aliasing filter and represents the loss of signal power due to the frequency filtering required for sampling to avoid aliasing.

3 Exercise C

3.1 Question

Now we want to obtain a fidelity of 99.99% for Q1 and Q2 for the same operation as in point A. In addition to the above-mentioned components, first-order linear filters are available as well, for which you can choose the bandwidth. For an example on how to implement a filter in Matlab, refer to the file first_order_filter.m on Brightspace. Explain where the filter(s) must be added, why, and give a range for their specifications. You are allowed to modify the baseband signals and the number of bits N of the DAC(s) with respect to point A. Minimize N.

Answer: In short, keeping the baseband signals unchanged compared to A, N is minimized to 6.

The first order filter is added as follows.

```
% pole (cutoff) frequency
fp=0.5e9;

%generate filter parameters
a=[1 2*pi*fp*dt-1];
```

(a) Unfiltered Signal

(b) Filtered Signal

Figure 3: Unfiltered VS. Filtered Signals

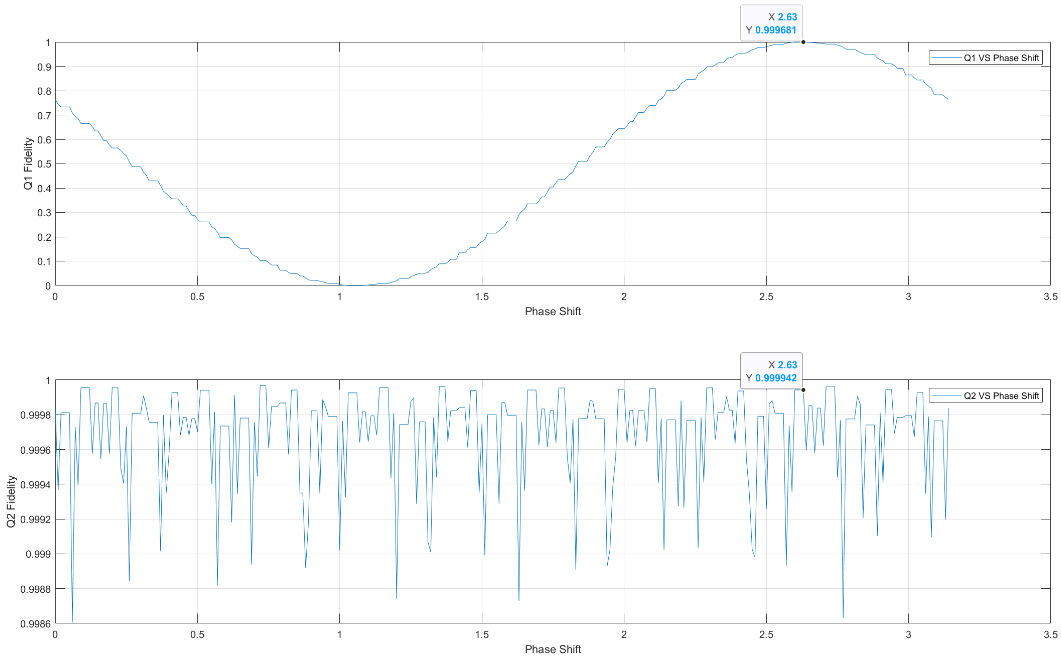

Figure 4: Effect of Phase Shift

```
b=[0 2*pi*fp*dt];

% apply filter to VoutDAC_I
VoutDAC_I_filtered = filter(b, a, VoutDAC_I);

% apply filter to VoutDAC_Q
VoutDAC_Q_filtered = filter(b, a, VoutDAC_Q);
```

The comparison of unfiltered signal and filtered signals is shown in figure 3.

The effect of Phase Shift, Gain1 and Bits of DAC is as shown in figure 4, figure 5 and figure 6.

Last but not the least, the fidelity of the qubits are kept above 99.9%. Moreover, as shown in figure 7, the π rotation also remains unchanged.

Please refer to file "HW6_NEW.m" for more detials. By running the script, the final result could directly be obtained. The other scripts used to determine the result is also in the folder, they are named after their functionalities clearly.

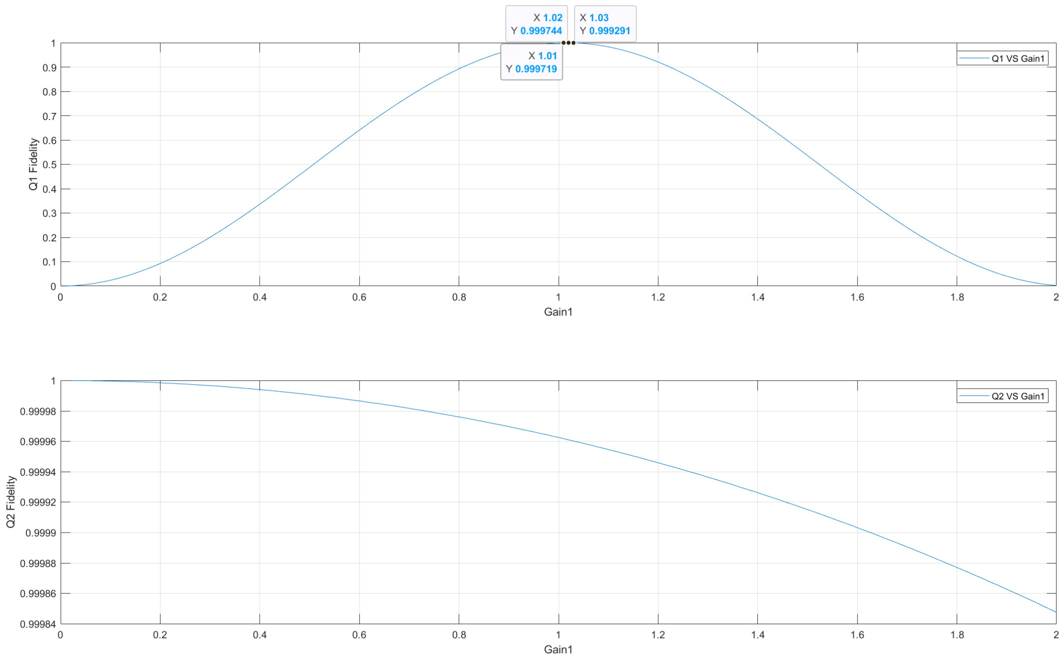

Figure 5: Effect of Gain1

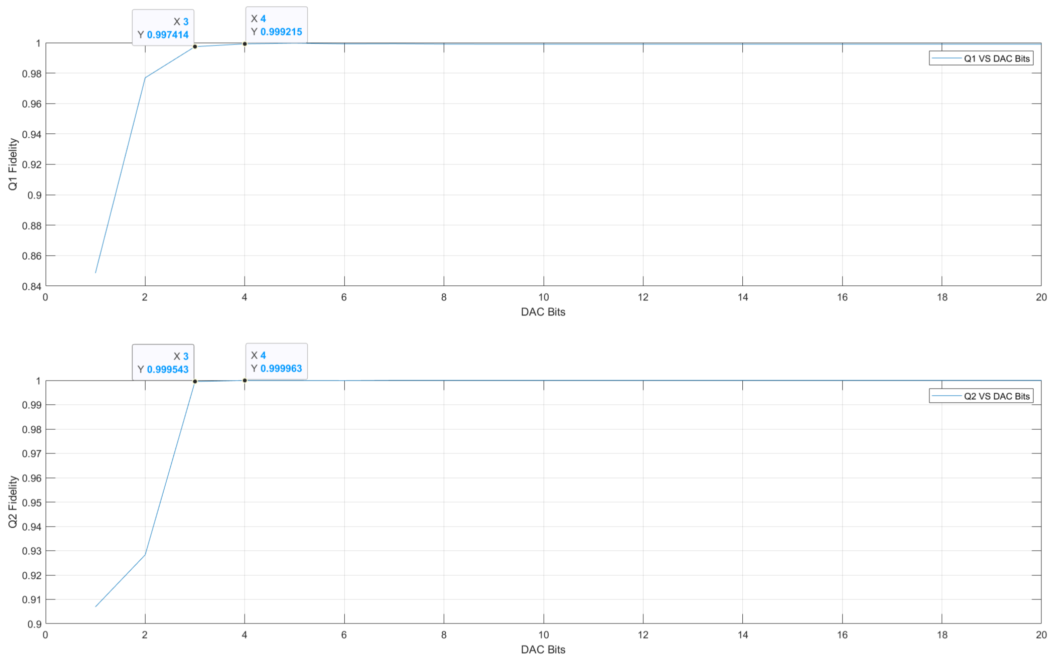

Figure 6: Effect of DAC Bits

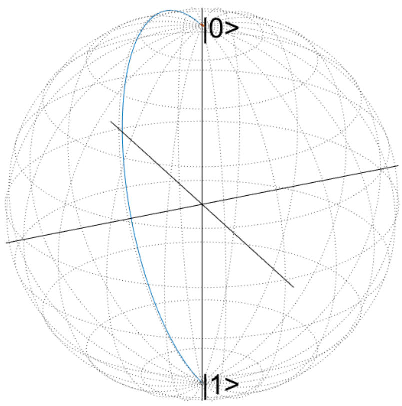

Figure 7: Qubit 1 rotate π, and qubit 2 remain idle

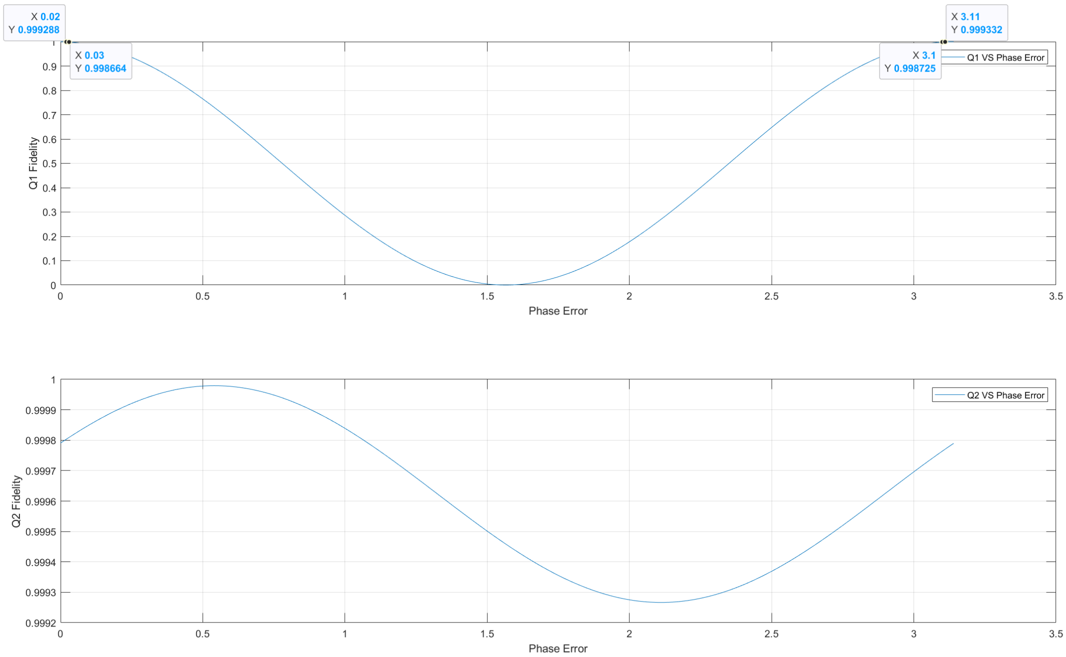

Figure 8: Phase error VS. Qubits Fidelity

4 Exercise D

Use the same system developed at point B but now allow a fidelity of 99.9% on Q1 and Q2 for the same operation as above. However, the local oscillator cannot be considered ideal anymore: its outputs show amplitude and phase errors.

As a general method to be applied in the two following questions. The method to inject amplitude and phase errors is as follows.

```
phase_error = 0; % Phase error in radians
amplitude_error = 0; % Amplitude error
I_mix = (1 + amplitude_error) * 2 * VoutDAC_I .* cos(2 * pi * fosc * t + phase_error);
Q_mix = (1 + amplitude_error) * 2 * VoutDAC_Q .* sin(2 * pi * fosc * t + phase_error);
Mix_out = I_mix + Q_mix;
```

4.1 Question

Find the maximum allowed phase mismatch, when assuming no amplitude mismatch.

Answer: Running script "HW6_D_Q1.m", the result obtained is shown in figure 8. The maximum allowed phase mismatch is 0.02 rad, while not considering amplitude error.

4.2 Question

Find the maximum allowed combined amplitude and phase mismatch.

Answer: Running script "HW6_D_Q2.m", the result obtained is shown in figure 9. The relation between amplitude error and qubits fidelity is linear. As the amplitude error increases, the fidelity decreases linearly. While not considering phase mismatch error, the biggest value allowed for amplitude error is around 2%.

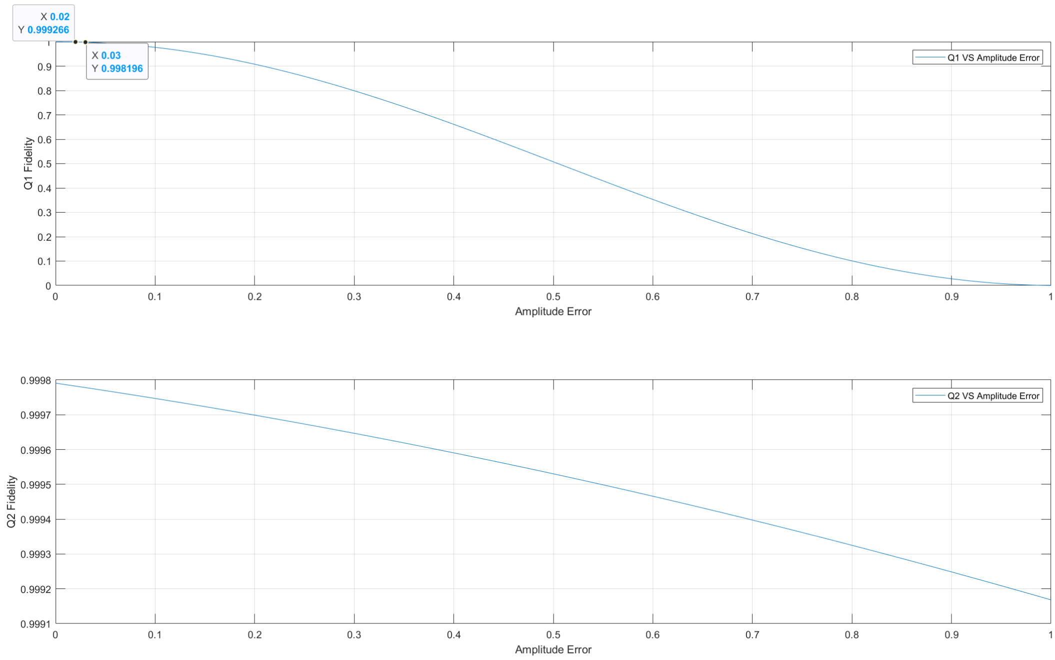

Figure 9: Amplitude error VS. Qubits Fidelity

We resue the script in the previous question while setting the amplitude error to 1%, 2%, 3% and 4% respectively. The results achieved are as shown in figure 10, figure 11, figure 12 and figure 13.

We conclude that when amplitude error and phase error combined, it could be either **Amp error = 1%, Phase error = 0.02 rad** or **Amp error = 2%, Phase error = 0.01 rad**.

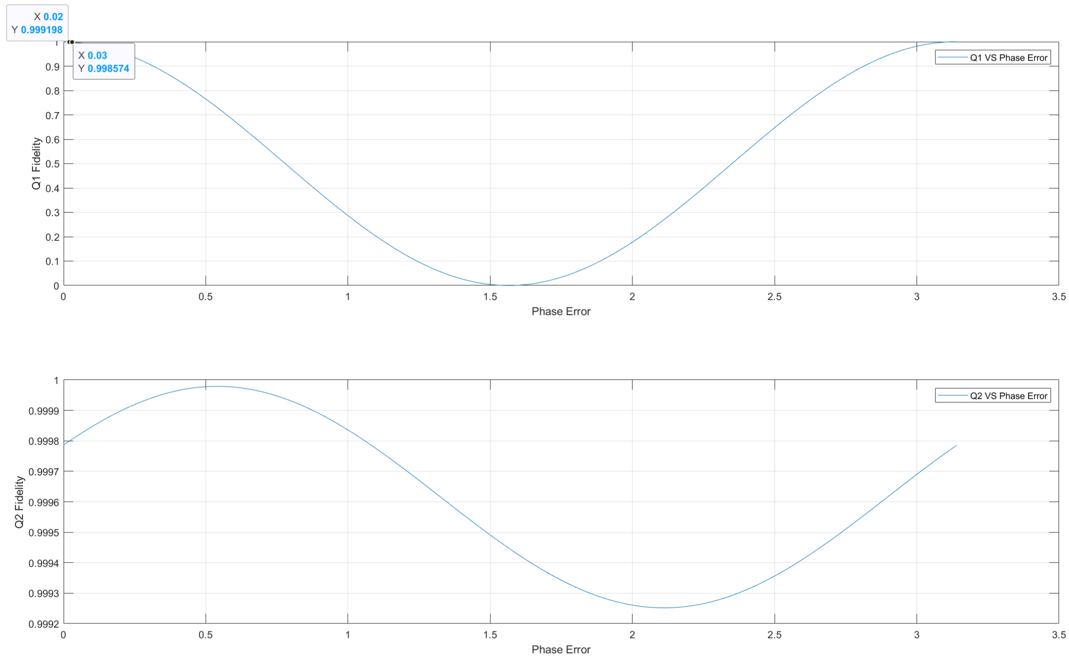

Figure 10: Phase error (with Amp error = 1%) VS. Qubits Fidelity

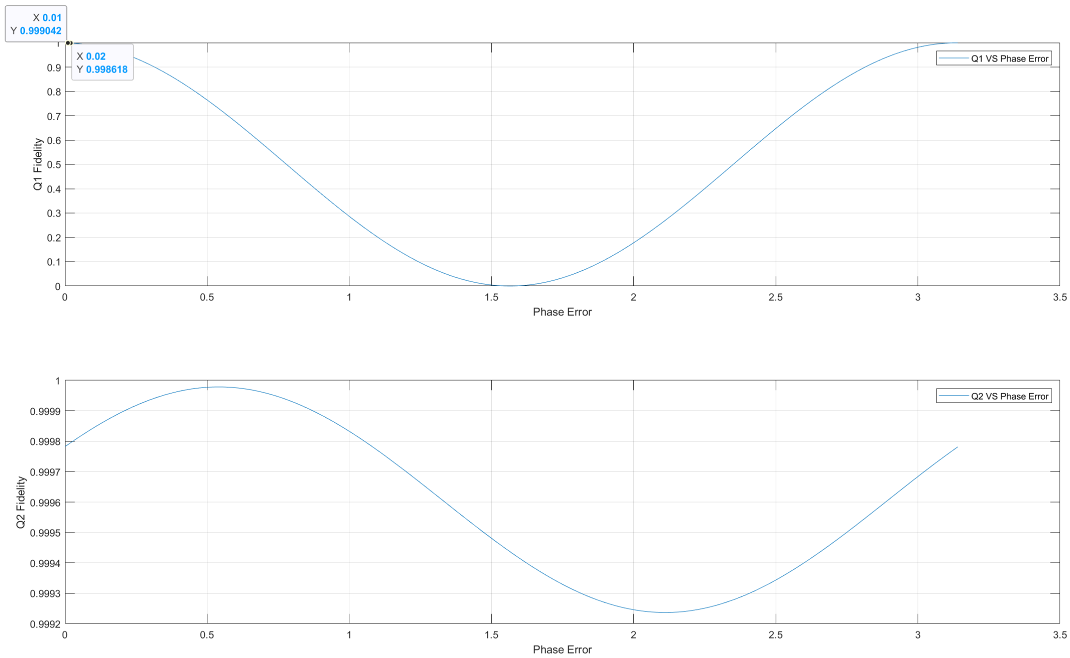

Figure 11: Phase error (with Amp error = 2%) VS. Qubits Fidelity

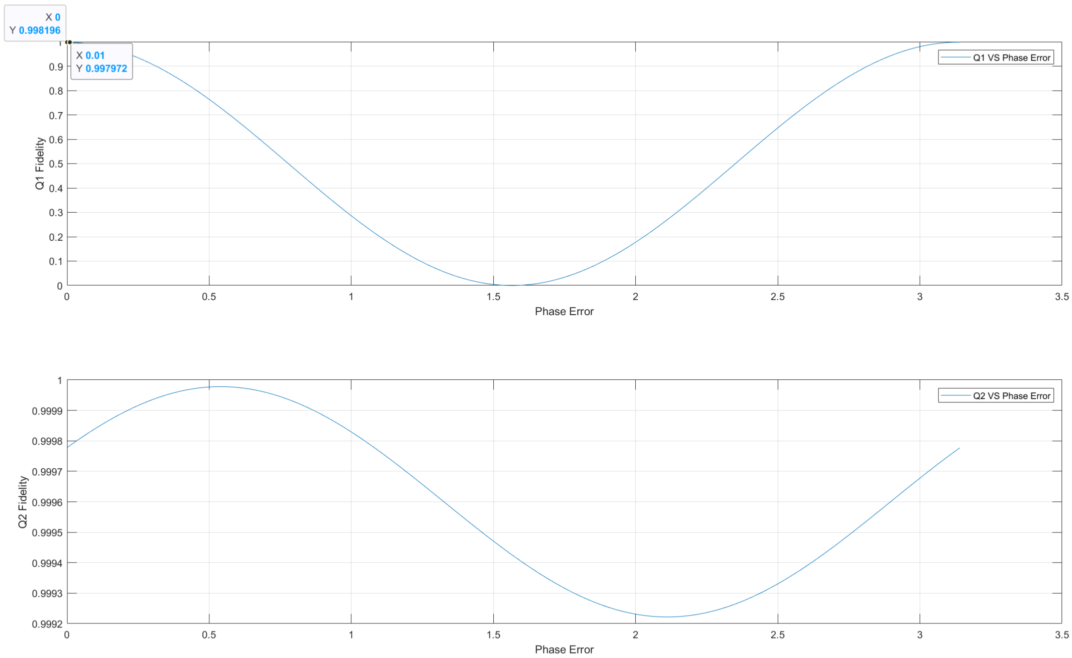

Figure 12: Phase error (with Amp error = 3%) VS. Qubits Fidelity

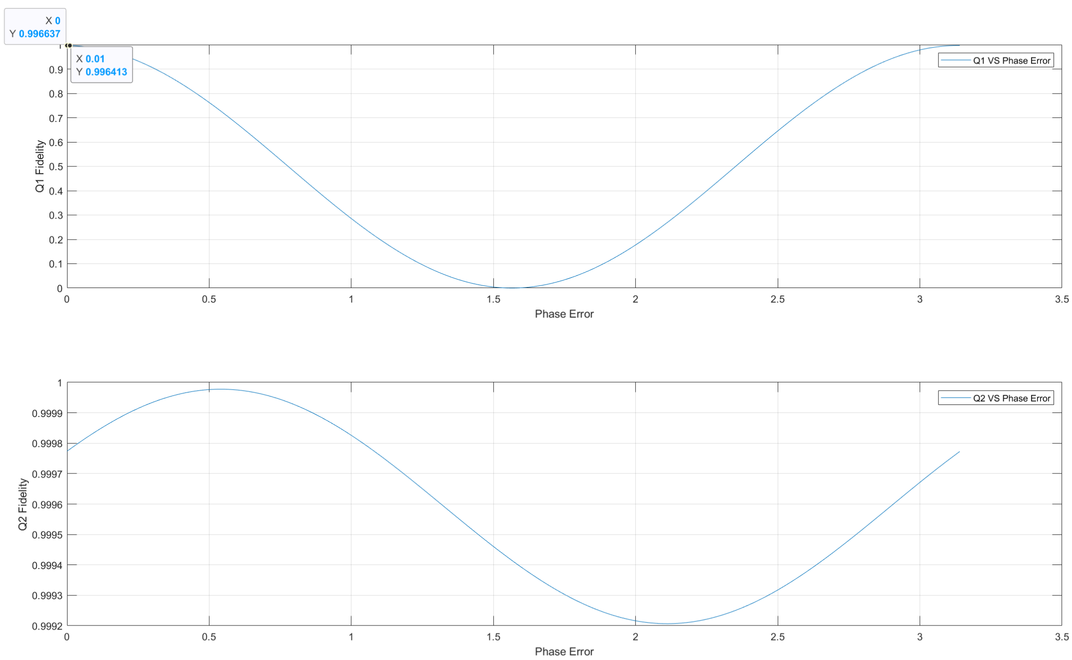

Figure 13: Phase error (with Amp error = 4%) VS. Qubits Fidelity

This homework is done with the help gained from [1].

References

[1] J. van Dijk, "Designing the electronic interface for qubit control," Ph.D. dissertation, TUDelft, 2021.