Frequentist and Bayesian Implementations of Gate Set Tomography

by

Daniel Berman

August 2015

Abstract

A current approach to determining the quality of quantum logic gates, as well as the error on those logic gates, is to use Gate Set Tomography (GST). To attempt to reconstruct the logic gates, GST leverages an estimation technique that attempts to reconstruct the logic gates using a frequentist approach, typically a maximum likelihood estimator (MLE). However, there are many benefits to using a Bayesian approach to estimation, including the inherent uncertainty estimates calculated in the estimation process. The application of Bayesian parameter estimation to GST follows a similar process, but without using the MLE approach. The result is an estimation process that does not require a large amount of data and that produces standard deviations on estimated parameters with similar accuracy.

1 Introduction

An essential step in developing quantum computing capabilities is the capability to verify and validate quantum computing logic gates. Understanding and characterizing these logic gates accurately allows us to identify, diagnose, and minimize any potential errors that arise due to noise sources such as imperfect pulses, qubit-qubit coupling, and environmental noise. Metrics for quantum error correction (QEC) can help determine whether the system meets the threshold for logic gate error rates suitable for QEC.

A method of performing this characterization is Gate Set Tomography (GST). GST is a more general form of Quantum Process Tomography (QPT), which makes the assumption that the initial and final states are subject to the same errors that the logic gates are (i.e., state preparation and measurement (SPAM) errors). SPAM errors come in two types: intrinsic and extrinsic. Intrinsic errors are inherent to the state preparation and measurement process, like thermal noise and dark counts. Extrinsic errors are caused by errors in the logic gates that transform the initial state to the starting state. Conversely, QPT assumes that the initial state and final state are perfectly prepared.

Blume-Kohout et al. (2013) detail the process of GST, including the linear-inversion process for linear gate set tomography (LGST). LGST is not subject to local maxima, unlike maximum-likelihood estimation (MLE) algorithms. An overview of LGST is presented in the following section.

2 Linear GST

The method for performing linear GST (LGST) was first published by Blume-Kohout et al. (2013). This method is limited in that it does not physically constrain the estimates, as a non-physical estimate may be better than a physical estimate. Therefore, a second method for estimating the logic gates must be used that physically constrains the gate estimates. The estimates from LGST, can therefore, be used as starting points for that second method. Since the final and initial states are assumed to be faulty, multiplying them by the SPAM gates corrects this: $\langle\langle E_j| = \langle\langle E|F_j$ and $|\rho_i\rangle\rangle = F_i|\rho\rangle\rangle$. Since the SPAM gates, F, are composed of the logic gates G, they must include the null gate $\{\}$ and the set must have enough gates to form a complete set of measurements. Additionally, all gates are estimated at the same time when using LGST, whereas in QPT, each gate is estimated individually.

2.1 Gram matrix and gauge transformation

The probabilities that are estimated for each set of SPAM gates and logic gates are: $p_{ikj} = \langle\langle E|F_i G_k F_j|\rho\rangle\rangle$. We can introduce a completeness statement to obtain

$$p_{ikj} = \Sigma_{rs}\langle\langle E|F_i|r\rangle\rangle\langle\langle r|G_k|s\rangle\rangle\langle\langle s|F_j|\rho\rangle\rangle \equiv \Sigma_{rs}A_{ir}(G_k)_{rs}B_{sj} : \tag{1}$$

where A and B are matrices

$$\begin{aligned} A &= \Sigma_i|i\rangle\rangle\langle\langle E|F_j \\ B &= \Sigma_j F_j|\rho\rangle\rangle\langle\langle j|. \end{aligned} \tag{2}$$

It is important to note that for all i and j, Equation 1 can be simplified as

$$\tilde{G}_k = AG_kB. \tag{3}$$

In the case of the null gate (the identity matrix, G_o), when $k = 0$, Equation 3 becomes

$$g = \tilde{G}_o = AB. \tag{4}$$

The matrix g is the Gram matrix. Performing simple matrix algebra on $\tilde{G}_k = AG_kB$, we find that:

$$\hat{G}_k = g^{-1}\tilde{G}_k = B^{-1}A^{-1}AG_kB = B^{-1}G_kB, \tag{5}$$

where $\hat{G}_k$ is the LGST estimate and B is an unobservable similarity transformation.

This is because as $\langle\langle E| \to \langle\langle E'| = \langle\langle E|B,\ |\rho\rangle\rangle \to \langle\langle \rho'| = B^{-1}|\rho\rangle\rangle$, and $G_k \to G_k^{'} = B^{-1}G_kB$, it is clear that $\langle\langle E^{'}|G_{i1}^{'}G_{i2}^{'}...G_{iL}^{'}|\rho'\rangle\rangle = \langle\langle E|G_{i1}G_{i2}...G_{iL}|\rho\rangle\rangle$. Therefore, we can construct a set of logic gates consistent up to a gauge transform B with

$$\begin{aligned}\langle\langle \hat{E}| &= \langle\langle \hat{E}|(\hat{B})^{-1}\\ |\hat{\rho}\rangle\rangle &= \hat{B}|\hat{\rho}\rangle\rangle\\ \hat{G}_k &= \hat{B}\hat{G}_k(\hat{B})^{-1}.\end{aligned} \tag{6}$$

We can determine B, which for our purposes is indistinguishable from the actual gauge, by comparing the LGST gates $\hat{G}_k$ to the ideal gates T_k by minimizing over

$$\hat{B}^* = \Sigma_{k=1}^{K+1} Tr\big((\hat{G}_k - \hat{B}^{-1}T_k\hat{B})^T(\hat{G}_k - \hat{B}^{-1}T_k\hat{B})\big), \tag{7}$$

where $G_{K+1} = |\rho\rangle\rangle\langle\langle E|$. This additional gate improves the fit on the initial and final states, making our final LGST estimate:

$$\begin{aligned}\langle\langle \hat{E}^*| &= \langle\langle \hat{E}|(\hat{B}^*)^{-1}\\ |\hat{\rho}^*\rangle\rangle &= \hat{B}^*|\hat{\rho}\rangle\rangle\\ \hat{G}_k^* &= \hat{B}^*\hat{G}_k(\hat{B}^*)^{-1}.\end{aligned} \tag{8}$$

3 Estimation Methods

3.1 Frequentist Methods

The standard methods for performing GST and QPT require the use of classical estimation techniques, typically MLE. Under these methods, we are looking to estimate $\hat{p}_{ijk}$, which is written in terms of the SPAM gates, F and the logic gate, G:

$$\hat{p}_{ijk} = \langle\langle \hat{E}|\hat{F}_i\hat{G}_k\hat{F}_j|\hat{\rho}\rangle\rangle. \tag{9}$$

MLE proceeds to estimate the values of $\hat{E}$,$\hat{\rho}$,$\hat{F}_i$,$\hat{G}_k$, and $\hat{F}_j$ that minimize the likelihood function:

$$L(\hat{E},\hat{\rho},\hat{G}) = \Pi_{ijk}(\hat{p}_{ijk})^{m_{ijk}}(1-\hat{p}_{ijk})^{1-m_{ijk}}. \tag{10}$$

In this form, m_{ijk} is the expected value (either simulated or measured) for each SPAM and logic gate combination. For large numbers of measurements, with the central limit theorem, we can instead use a normal distribution to obtain a likelihood function:

$$L(\hat{E},\hat{\rho},\hat{G}) = \Pi_{ijk}exp[-(m_{ikj}-p_{ijk})^2/\sigma^2_{ijk}]. \tag{11}$$

The log likelihood form of this function is

$$l(\hat{E},\hat{\rho},\hat{G}) = \Sigma_{ijk}(m_{ijk}-\hat{p}_{ijk})^2/\sigma^2_{ikj}. \tag{12}$$

In the case of finite measurements, it is possible for $\sigma^2 = 0$ in the case where the measured ratio $m_{ijk} = 1$. Therefore, in order to prevent any significant problems with estimation, the variance from the maximum likelihood is simply treated as a constant (i.e., equal to 1). As shown later, this does not significantly impact estimates. To impose physicality constraints, the logic gates have to be in the form of either the process matrix or the Pauli Transfer Matrix (PTM). This paper will discuss the process matrix approach, as the formulation is much simpler.

The definition of a process matrix for a logic gate $G(\rho)$ is

$$G(\rho) = \Sigma_{i,j=1}^{d^2}(\chi_G)_{ij}P_i\rho P_j, \tag{13}$$

where P_i and P_j are the Pauli operators. A requirement of χ is that it is Hermitian positive semidefinite, which means it can be written in terms of a Cholesky decomposition, $\chi = LL^\dagger$. In addition to Hermiticity and positive semidefiniteness, trace must be conserved: $\mathrm{Tr}\,(G(\rho)) = \mathrm{Tr}\,(\rho)$. This brings about a completeness condition that:

$$\Sigma_{ij}\chi_{ij}TrP_iP_rP_j = \delta_{r0}, k = 0, ..., d^2. \tag{14}$$

Additionally, ρ and E can be parametrized by a Cholesky decomposition, ρ must have unit trace, and E must be such that $I - E$ is also positive semidefinite.

Finally, a gauge transformation, as done in equations 7 and 8 is performed on the estimates of the logic gates, initial state, and final state with the starting point being the identity matrix. This is to correct any rotation that occurs during the MLE process. The starting point is the identity matrix because the rotation is expected to be small.

Since this is a frequentist method, the estimates do not come with standard deviations. Therefore, to get error bars on the MLE estimates, methods such as resampling or Monte Carlo simulation must be performed. A Monte Carlo approach works when using simulated data. However, when dealing with experimental data, resampling is the only option. These methods require large data sets to ensure accurate estimates. Since each resampling requires executing the entire ML-GST process repeatedly, it is very time intensive.

3.2 Bayesian Method

This paper examines the use of a Bayesian parameter estimation technique to estimate the values of the logic gates G_k, and the initial and final states ρ and E. Bayesian parameter estimation is a process by which a parameter of a given probability distribution is estimated using Bayes' Rule:

$$p(\theta|s,n) = \frac{p(s|\theta,n)p(\theta)}{p(s|n)} = \frac{p(s|\theta,n)p(\theta)}{\int p(s|\theta,n)p(\theta)d\theta}, \tag{15}$$

where

$$p(s|n) = \int p(s|\theta, n)p(\theta)d\theta. \tag{16}$$

In Equations 15 and 16, s is the data, θ are the parameters of interest, and n is data from any independent variables. For example, for the number of heads in a series of coin flips, s is the number of heads, n is the number of coin flips, and θ is the probability of getting a heads on a single coin flip. In Equation 15, $p(\theta)$ is the prior distribution, reflecting prior knowledge of the parameter θ. Generally, a more constrained prior distribution indicates more knowledge about the parameter θ. In the case where we know nothing about the prior, referred to as an uninformed prior, a uniform distribution is typically used (Gelman, Carlin, Stern, & Rubin, 2014). The term $p(s|\theta, n)$ is known as the likelihood, or the probability, of the data given the parameter θ. As a result, this represents the model we believe explains the data. The term $p(\theta|s, n)$ is known as the posterior probability and represents our most recent knowledge of θ. The denominator contains the marginal likelihood term. This is the probability of the data occurring given all parameter values and models. Therefore, it can be seen simply as a constant and Bayes' Rule can be rewritten as

$$p(\theta|s, n) \propto p(s|\theta, n)p(\theta), \tag{17}$$

which means that the posterior probability is proportional to the prior probability multiplied by the likelihood.

As new data is collected, θ can be updated, independent of the order of the data. In doing this, the posterior probability from the old data becomes the prior probability for the new data. By integrating over the posterior, uncertainty in our prediction is accounted for, unlike with methods such as MLE, making Bayesian estimation a better way of developing predictions (Aitchison & Dunsmore, 1980).

Since many optimization problems do not have closed form solutions, this requires the use of Markov Chain Monte Carlo (MCMC) methods. MCMC is a type of algorithm that is used to sample a probability distribution in order to estimate the parameters that define it. A Markov chain is a sequence of random variables x_1, x_2,... for which the random variable x^i depends only on the previous xs through the random variable immediately before it, x_{i-1}. There are a number of MCMC methods including Gibbs sampling, Metropolis-Hastings, and Hamiltonian Monte Carlo (HMC) algorithms. A description of these methods is beyond the scope of this paper. Code written for this paper makes use of the HMC method (Duane, Kennedy, Pendleton, & Roweth, 1987; Neal, 1994, 2011).

Our Bayesian parameter estimation uses the software Stan (Stan Development Team, 2015b), a Bayesian statistical inference programming language that uses MCMC sampling, specifically, the HMC algorithm, with interfaces for Matlab, Python, and R. This code allows for the definition of prior distributions and sampling. One limitation of Stan is that it cannot use complex numbers. Additionally, Stan defines covariance and correlation matrices as positive definite. Therefore, positive semidefinite matrices are not possible. There are workarounds for both of these problems.

There are three parameters that need to be estimated: G_k, ρ, and E. Since each of these have complex components, they need to be written in a form that can be calculated in Stan. Since ρ and E are both required to be Hermitian positive semidefinite matrices, ρ and E can be rewritten in block form. In this method, a complex valued, $n \times n$ dimensional, Hermitian, positive semidefinite matrix A can be rewritten as a $2n \times 2n$ symmetric, real, positive semidefinite matrix B:

$$\tilde{A} = \begin{pmatrix} \mathrm{Re}(A) & -\mathrm{Im}(A) \\ \mathrm{Im}(A) & \mathrm{Re}(A) \end{pmatrix} \succeq 0, \tag{18}$$

where both Re and $\mathrm{Im}(A)$ are positive semidefinite (Higham, 1998). Since Re(A) contains the real values, it is symmetric. Conversely, $\mathrm{Im}(A)$ must be antisymmetric since A is Hermitian. Additionally, a property of positive semidefinite matrices is that they can be decomposed into a Cholesky matrix. A Cholesky matrix is a lower triangular matrix, L, which, when multiplied with its conjugate transpose, results in a positive semidefinite matrix $A = LL^\dagger$. Stan has a built-in distribution for Cholesky correlation matrices, with unit diagonal elements. Therefore, the parameters ρ and E can be estimated using a model with ρ and E dependent on the Cholesky decomposition of their real and imaginary parts: $\rho_{\_real\_chol}$, ρ_{imag_chol}, E_{real_chol}, and E_{imag_chol}. However, we have no prior knowledge about the distribution of the actual states relative to the LGST estimates, so it is best to use a uniform distribution for the Cholesky matrices $\rho_{\_real\_chol}$, ρ_{imag_chol}, E_{real_chol}, and E_{imag_chol}. This is done using a Lewandowski, Kurowicka, and Joe (LKJ) correlation distribution, which takes a parameter η and generates Cholesky correlation matrices. In the case where $\eta = 1$, the LKJ correlation distribution acts as a uniform distribution (Lewandowski, Kurowicka, & Joe, 2009).

Additional requirements for ρ and E are that unit trace of ρ and the logic gates G must be trace preserving. This results in the conditions $tr(\rho) = tr(E) = 1$. This means that we can derive two constraints: 1) the imaginary values along the diagonal must be zero, and 2) the traces of ρ and E must be real and unit. Stan does not allow for explicit constraints, but there is a way to implement these. The first constraint can be enforced in defining $\rho_{imag} = \rho_{imag\_chol} - \rho^\dagger_{imag\_chol}$ and $E_{imag} = E_{imag\_chol} - E^\dagger_{imag\_chol}$. This results in an antisymmetric matrix with zeros along the diagonal, called an antisymmetric hollow matrix.

The second constraint can be enforced through the use of simplexes π. Simplexes are K-dimensional vectors containing nonnegative elements that sum to one, giving them K-1 parameters. A Dirichlet distribution can be used as the prior for the simplex. A Dirichlet distribution is a distribution of order $K \geq 2$ with parameters $a_1, a_2, ..., a_K > 0$ and with functional form:

$$f(x_1, x_2, ...x_{K-1}; a_1, a_2, ..., a_K) = \frac{1}{B(a)} \Pi_{i=1}^{K} x_i^{a_i - 1}, \tag{19}$$

where $B(a)$ is the multinomial Beta Function:

$$B(a) = \frac{\Pi_{i=1}^{k} \Lambda(a_i)}{\Lambda(\Sigma_{i=1}^{K} a_i)}, \tag{20}$$

where $\Lambda(t)$ is the gamma function. In the case where all $a_i = 1$, the Dirichlet distribution simplifies to $f(x_1, ...x_{K-1}; a_1, ..., a_K) = \Lambda(K)$, yielding a uniform distribution for generating simplexes. Projecting this simplex onto a K-dimensional identity matrix, and inserting it in between the product of $LL^\dagger$ to become $L\pi L^\dagger$, produces a matrix with unit trace. This is used to generate ρ_{real} and E_{real}. These two approaches allow for the creation of ρ and E.

The values of the logic gates G_k do not possess as many constraints as ρ and E. Stan does not have a distribution for generating random matrices. As a result, our approach was to estimate each element of the logic gates individually. The prior on each element of the block form of G_k was assumed to be normally distributed about the LGST estimate with a standard deviation sigma .1.

Since we have broken G_k, ρ, and E into block forms, we only retain the real portions of $\langle\langle E|F_i G_k F_j|\rho\rangle\rangle$ while discarding the imaginary portions, as shown in the following example:

$$p_{ijk} = \begin{pmatrix} \mathrm{Re}(E)\,\mathrm{Im}(E) \end{pmatrix} \begin{pmatrix} \mathrm{Re}(F_i) & -\mathrm{Im}(F_i) \\ \mathrm{Im}(F_i) & \mathrm{Re}(F_i) \end{pmatrix} \begin{pmatrix} \mathrm{Re}(G_k) & -\mathrm{Im}(G_k) \\ \mathrm{Im}(G_k) & \mathrm{Re}(G_k) \end{pmatrix}$$
$$X \begin{pmatrix} \mathrm{Re}(F_j) & -\mathrm{Im}(F_k) \\ \mathrm{Im}(F_j) & \mathrm{Re}(F_k) \end{pmatrix} \begin{pmatrix} \mathrm{Re}(\rho) \\ \mathrm{Im}(\rho) \end{pmatrix} \tag{21}$$

Notice that these would all be real, forming p_{ijk}. The input data for the measurements is binomially distributed. Because the equation above only sums the real portions in the block form, there is no constraint on the imaginary parts. In order to constrain the imaginary portions, we simply switch the blocks in the G_k block matrix to

$$\begin{pmatrix} -\mathrm{Im}(G_k) & \mathrm{Re}(G_k) \\ \mathrm{Re}(G_k) & \mathrm{Im}(G_k) \end{pmatrix}. \tag{22}$$

Therefore, the imaginary part of the probabilities is:

$$\begin{aligned}\tilde{p}_{ijk} = \begin{pmatrix} \mathrm{Re}(E)\,\mathrm{Im}(E) \end{pmatrix} \begin{pmatrix} -\mathrm{Im}(F_i) & \mathrm{Re}(F_i) \\ \mathrm{Re}(F_i) & \mathrm{Im}(F_i) \end{pmatrix} \begin{pmatrix} -\mathrm{Im}(G_k) & \mathrm{Re}(G_k) \\ \mathrm{Re}(G_k) & \mathrm{Im}(G_k) \end{pmatrix} \\ X \begin{pmatrix} -\mathrm{Im}(F_k) & \mathrm{Re}(F_j) \\ \mathrm{Re}(F_k) & \mathrm{Im}(F_j) \end{pmatrix} \begin{pmatrix} \mathrm{Re}(\rho) \\ \mathrm{Im}(\rho) \end{pmatrix}\end{aligned} \tag{23}$$

Therefore, the priors placed on the probabilities are:

$$\begin{aligned} m_{ijk} &\sim N(p_{ijk}, .05^2) \\ 0 &\sim N(\tilde{p}_{ijk},, .001^2) \end{aligned} \tag{24}$$

for all ikj.

Since Stan cannot explicitly constrain these values to be 0 for all ijk, an alternative is to use zero as a data point and have the sum of the imaginary components act as the mean of a normal distribution with standard deviation .001. This reduces the imaginary component to zero by limiting how much the mean of the normal distribution can stray from 0.

Finally, the physicality constraints were enforced using the same approach that restricted the imaginary portion of the probabilities to zero beginning at Equation 23. Since the physicality constraint requires the process matrix form of the superoperators, we must transform the gates within Stan from superoperators to process matrices. The matrix V that is used to transform a superoperator G into a process matrix χ is formed using the Pauli matrices and requires the kronecker product. As a result, the matrix V is constant depends only on the number of qubits and since the Pauli matrices are not required to have real elements, is computed in MATLAB and imported into Stan. Each row of V is the kronecker product

$$V[n,] = \mathrm{vec}(P_j^T \otimes P_i), \tag{25}$$

where i and j are the ith and jth Pauli matrices and n is the number of Pauli matrices squared. Then, a vectorized superoperator, vec(G), can be converted into a vectorized process matrix vec(χ) through

$$\text{vec}(\chi) = V^{-1}\,\text{vec}(G). \tag{26}$$

However, this matrix V is allowed to have imaginary values. Therefore, we must perform this operation in the same manner that was done in Equation 21, forming block matrices and vectors. Then we can apply Equation 14 for each of the k gates to constrain the estimates. There is another trick to applying the constraints within Stan. Since imaginary values are not permitted within Stan, we have to take the Tr(P_iP_rPj) and separate it into real and imaginary parts and then input it into Stan as constants. From there we take the kth gate and taking the real and imaginary portions of the process matrix we can rewrite the constraint as

$$\delta_{r,0} = \Sigma_{ij}(\text{Re}\,(\chi_{ij}) + \text{Im}(\chi_{ij}))(\text{Re}\,(Tr(P_iP_rPj)) + \text{Im}\,(Tr(P_iP_rPj))). \tag{27}$$

The imaginary parts of the RHS are equal to 0 and the real parts of the RHS are equal to δ_{r0}. For each iteration, the value of the constraint is calculated and set to be the mean of a normal distribution with standard deviation .01. Since the required value of the constraint (for real parts is δ_{r0} and imaginary parts is 0) is then set to be a "data" which is generated from that normal distribution, the mean is constrained, forcing the gate to be physical. This appears in the code as:

$$\begin{aligned}\text{Re}(\delta_{r,0}) &\sim N(\Sigma_{ij}(\text{Re}\,(\chi_{ij})\,\text{Re}\,(Tr(P_iP_rPj)) - \text{Im}\,(Tr(P_iP_rPj))\,\text{Im}\,(\chi_{ij}), .01^2)\\ \text{Im}(\delta_{r,0}) &\sim N(\Sigma_{ij}(\text{Re}\,(\chi_{ij})\,\text{Im}\,(Tr(P_iP_rPj)) + \text{Im}(\chi_{ij})\,\text{Re}(Tr(P_iP_rPj)), .01^2)\end{aligned} \tag{28}$$

for all r and all gates k.

In order to correct for any rotation that occurs during the estimation process, a gauge transformation is performed again, as shown in Equations 7 and 8. Stan does not perform this task, but fminunc is well suited for this task. Each sampling of the distribution for the logic gates, initial state, and final state produced by Stan are gauge transformed. From these, the variance on each element of the logic gates and states are calculated. Additionally, the gauge-transformed logic gates are used to calculate metrics for the closeness of the estimated logic gates to the actual logic gates. Since the closeness can be calculated for each of the samples, the variance can also be calculated.

There are two metrics for calculating the distance between two quantum states: fidelity and trace distance (Nielsen & Chuang, 2010). Fidelity is a measure of closeness between two states:

$$F(\rho, \sigma) = \mathrm{Tr}(\sqrt{\rho^{1/2}\sigma\rho^{1/2}}). \tag{29}$$

This measure, which is symmetric and whose values range from 0 to 1, is at a maximum when the states are the same. The second measure is called trace distance. Trace distance, whose values range from 0 to 1, has the form:

$$D(q, \sigma) = \mathrm{Tr}(\sqrt{(q-\sigma)^{\dagger}(q-\sigma)}). \tag{30}$$

Since it is not possible to require that the Bayesian estimate be physical, we can use the Bayesian estimate to compute the closest possible gates that meet the physical requirements. From there, we can calculate the actual trace and fidelity in MATLAB. Additionally, Stan returns the sampling distribution for each parameter, which we can use to calculate the fidelity mean and standard deviation.

3.2.1 Settings

The code for this project was written in MATLAB (MATLAB, 2015) and R (R Core Team, 2012), using the cvx toolbox (Grant et al., 2008) in MATLAB and the packages Rcpp (Eddelbuettel & François, 2011), inline (Sklyar et al., 2007), Rstan (Stan Development Team, 2015a), MASS (Venables & Ripley, 2002), ggmcmc (Xavier Fernandez i Marin, n.d.), coda (Plummer et al., 2006), and R.matlab (Bengtsson & Riedy, 2013) in R. Rstan was used rather than MatlabStan because was developed earlier and so is more extensively tested.

The majority of the code was written in MATLAB, including the LGST, MLE, and plotting, while the Bayesian parameter estimation, extraction, and processing of the raw MCMC data was performed in R.

This model was specifically tested for the one qubit scenario. Since there are a large number of parameters to estimate, it is necessary to have a large number of iterations. However, the more iterations and chains there are, the more time the model takes to run. Therefore, for the purposes of this paper, the number of iterations was set to 2500 and the number of chains was set to 1 to optimize run time and iterations. Convergence of the model can be improved with increased iterations.

In testing the model, it was important to verify that the model converged properly. This means that the values of R_hat produced by the sampling() function in R needed to be less than 1.1 (Gelman, Carlin, Stern, & Rubin, 2014), with the exception of values constrained to zero in the hollow matrix, which were NA. This was true for a significant number of the parameters, indicating that the model was behaving well.

4 Results

To test the code, a simulated experimental scenario was created using the logic gate set $G_{target} = \{I, X(\pi/2), Y(\pi/2), X(\pi)\}$. The source of the error was depolarization error with probability values q of 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. The operator form of depolarization channel for a single qubit is:

$$\Lambda_{dep}(rho) = (1 - \frac{3}{4}p)\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z). \tag{31}$$

In superoperator form, the depolarization channel can be expressed as the matrix:

$$\begin{pmatrix} 1-\frac{p}{2} & 0 & 0 & \frac{p}{2} \\ 0 & 1-p & 0 & 0 \\ 0 & 0 & 1-p & 0 \\ \frac{p}{2} & 0 & 0 & 1-\frac{p}{2} \end{pmatrix} \tag{32}$$

The depolarization error was introduced into the logic gates G_{target} by simply multiplying each logic gate by the depolarization channel. In addition to the variable number of errors p, the number of measurements N used to estimate the ijk logic gate pairing varied between $N = 500, 1000$, and 5000. Therefore, there are 24 different measurements for each logic gate. This paper only presents visualizations of the logic gates for the error value $p = 10^{-6}$ and one example of an error value of $p = 10^{-1}$. However, this paper presents measurements for all error values and sampling sizes for fidelity, relative fidelity error, and trace distance. Figures 1 through 6 show the plots of the fidelity, relative fidelity error, and trace distance for sampling sizes 500, 1000, and 5000. The relative fidelity error is

$$\frac{F_{actual} - F_{est}}{F_{actual}}, \tag{33}$$

where the F_{actual} is the fidelity of the actual logic gates (with the error) and the ideal logic gates, while F_{est} is the fidelity of the estimated logic gates and the ideal logic gates. The trace distance and fidelity calculated in this paper are in reference to the ideal logic gates.

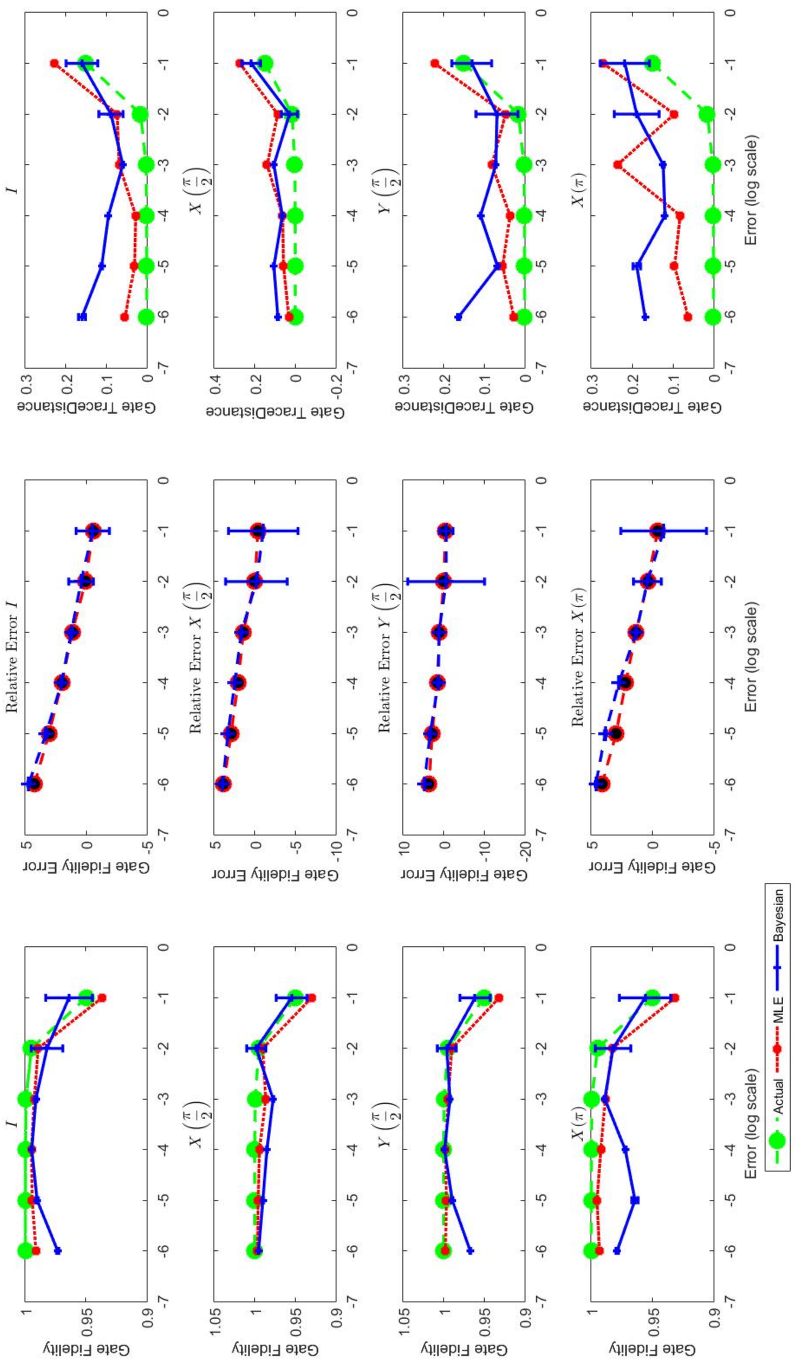

Figure 1: Plot of the metrics for the maximum likelihood estimates, actual logic gates, and Bayesian estimates with N=500 as compared to the target logic gates. The left column of this plot shows the fidelity as compared to the target logic gates. The center column is the relative fidelity error of the estimated logic gates on a logarithmic scale. The right column is the trace distance.

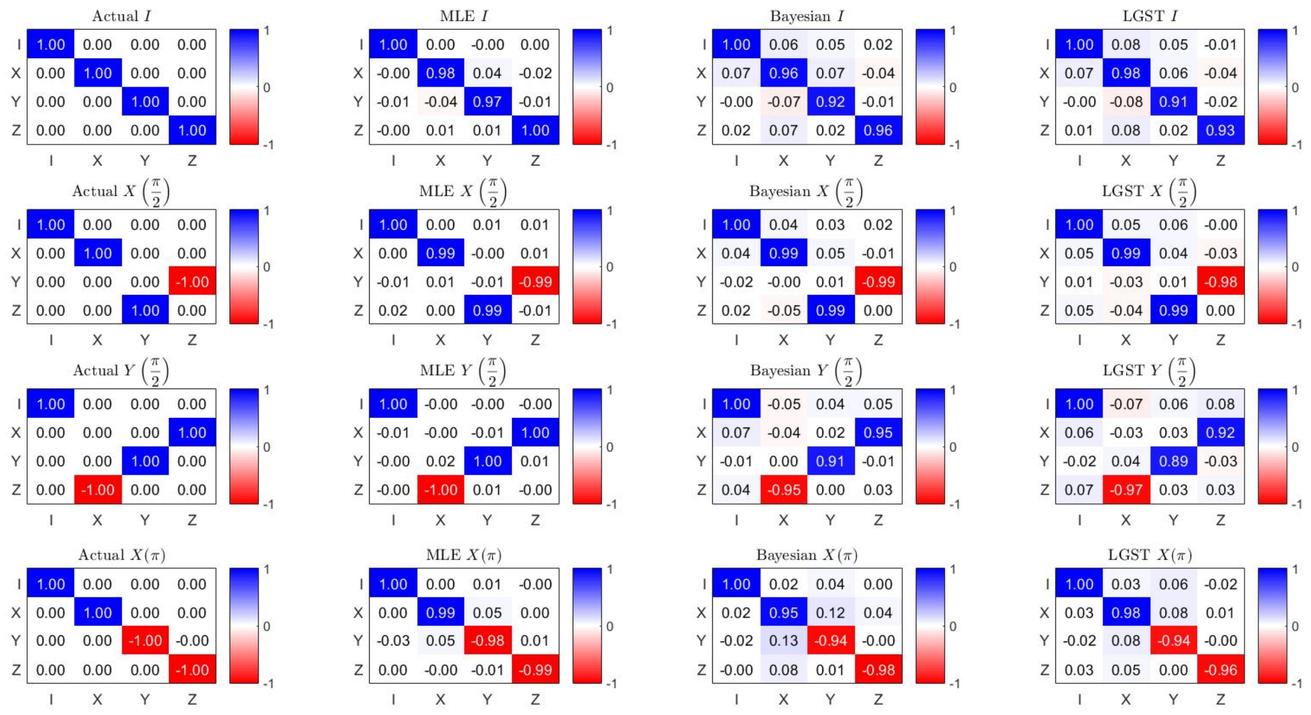

Figure 2: Plot of the PTM matrix for the actual logic gates including a depolarization error of 10^{-6}. This is followed by the ML, Bayesian, and LGST estimates for the logic gates in G. The number of data samples for the Bayesian estimate is N=500.

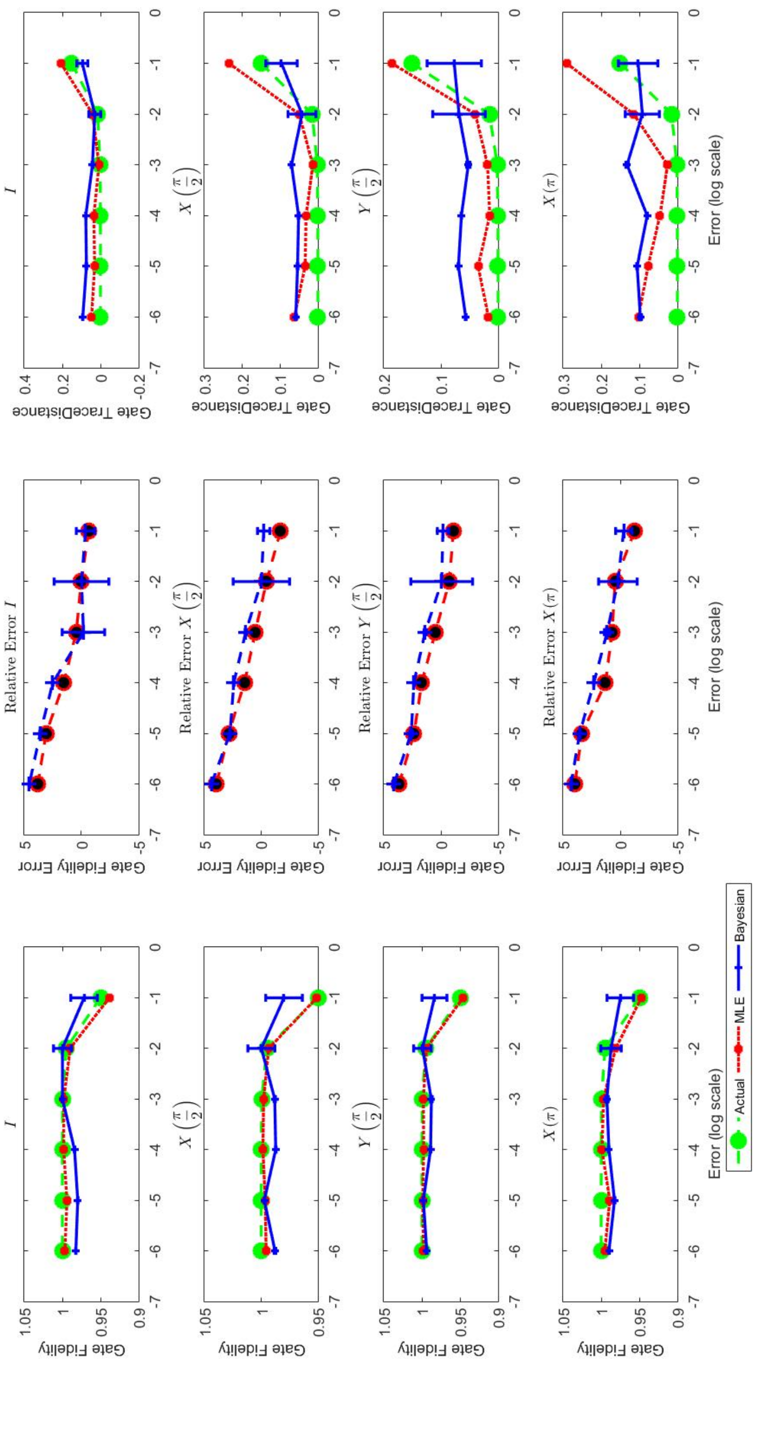

Figure 3: Plot of the metrics for the maximum likelihood estimates, actual logic gates, and Bayesian estimates with N=1000 as compared to the target logic gates. The left column of this plot shows the fidelity as compared to the target logic gates. The center column is the relative fidelity error of the estimated logic gates on a logarithmic scale. The right column is the trace distance.

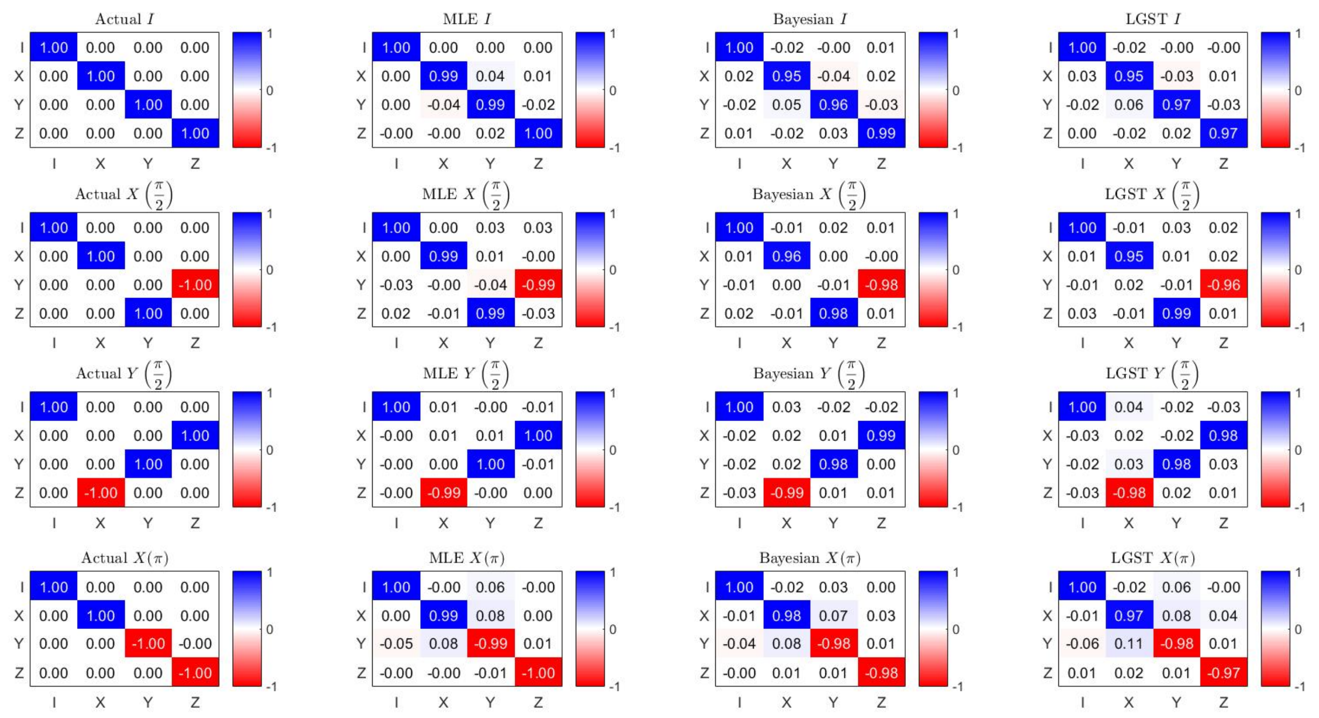

Figure 4: Plot of the PTM matrix for the actual logic gates including a depolarization error of 10^{-6}. This is followed by the ML, Bayesian, and LGST estimates for the logic gates in G. The number of data samples for the Bayesian estimate is N=1000.

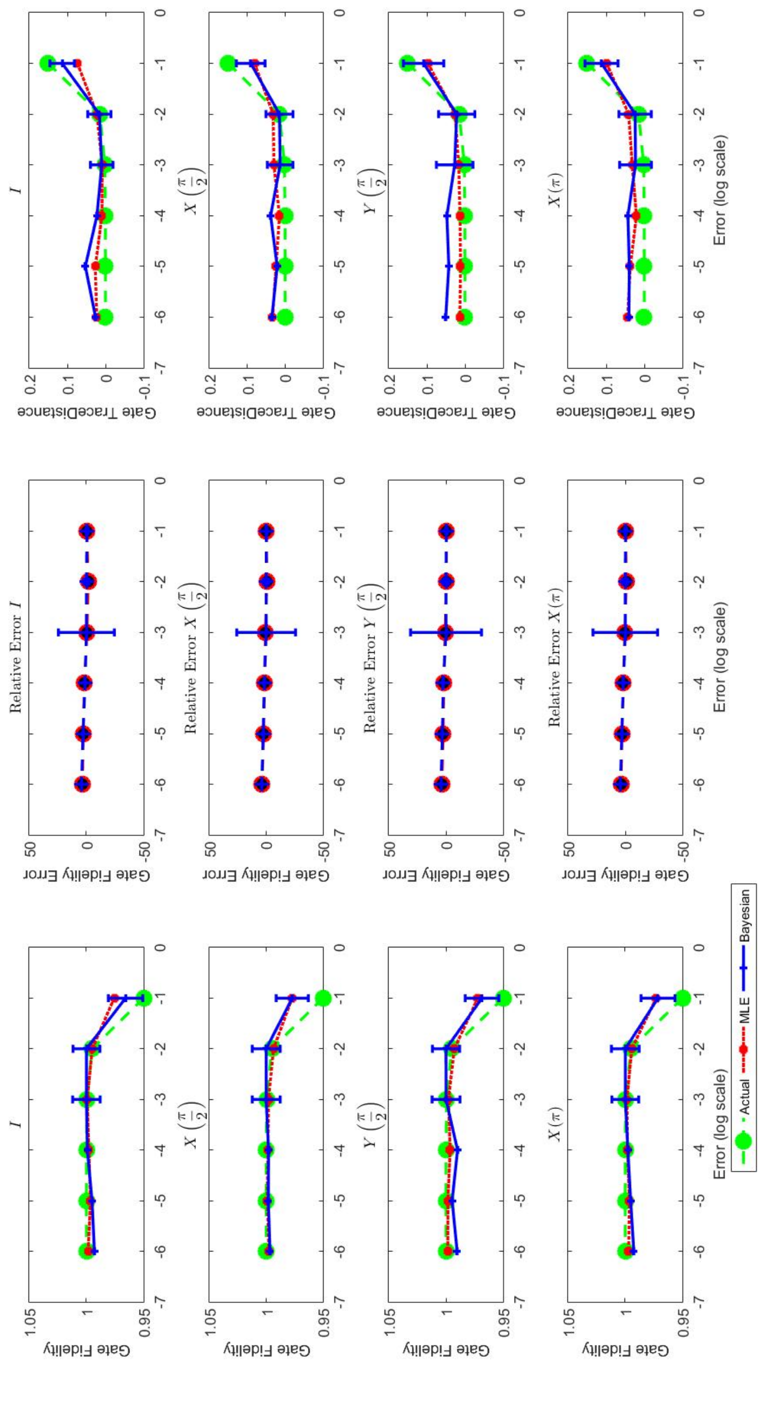

Figure 5: Plot of the metrics for the maximum likelihood estimates, actual logic gates, and Bayesian estimates with N=5000 as compared to the target logic gates. The left column of this plot shows the fidelity as compared to the target logic gates. The center column is the relative fidelity error of the estimated logic gates on a logarithmic scale. The right column is the trace distance.

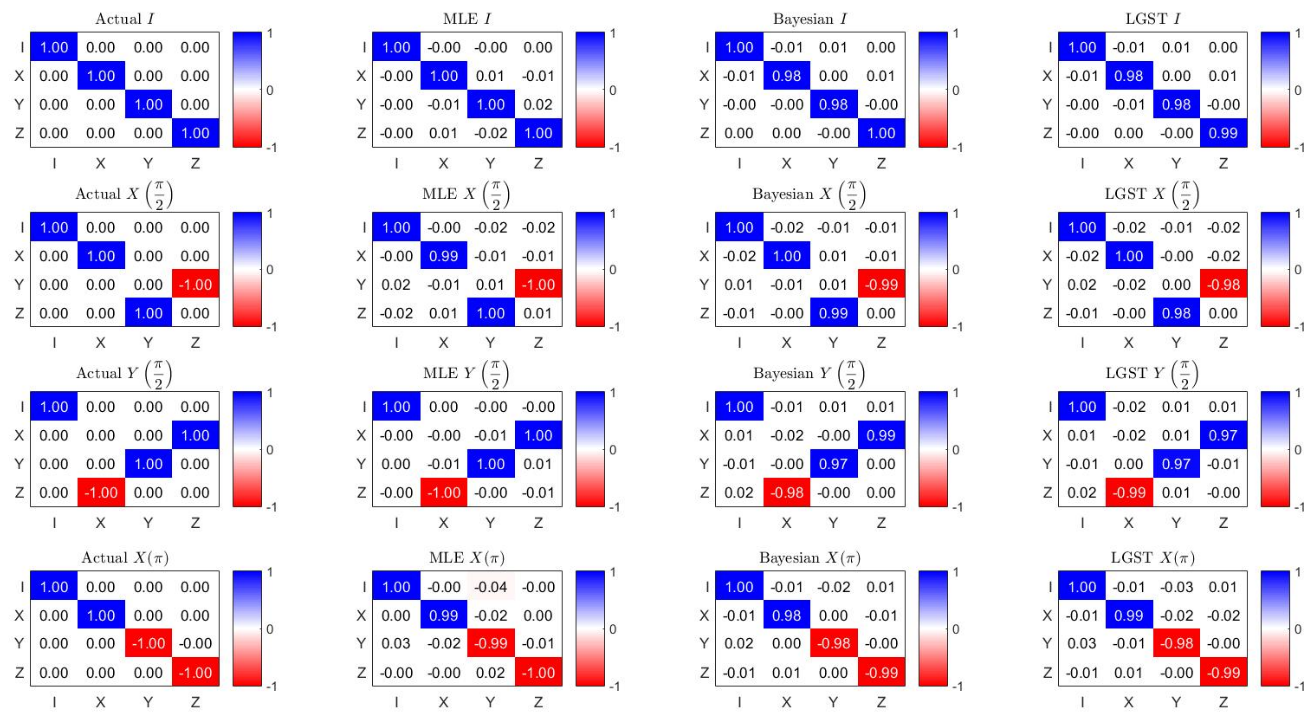

Figure 6: Plot of the PTM matrix for the actual logic gates including a depolarization error of 10^{-6}. This is followed by the ML, Bayesian, and LGST estimates for the logic gates in G. The number of data samples for the Bayesian estimate is N=5000.

Notice that in Figures 1, 3, and 5, the gate fidelity and trace distance approach 1 as the error decrease for all logic gates. However, for higher error, the Bayesian estimates of the logic gates tend to underestimate the error and tends more towards the ideal logic gates. Additionally, like the ML-GST estimates and the actual logic gates, the Bayesian estimates tend to approach the ideal estimates as the error decreases and the sampling number N increases. Note that the relative error for the Bayesian estimates in the logic gates decreases significantly for large errors in the logic gates with the number of samples N taken, improving to the point where it approaches the the error rate of the ML-GST estimates.

Figures 2, 4, and 6 show the PTM representation of the actual and estimated logic gates with the depolarization channel having a value of $p = 10^{-6}$. The PTM representation of an operator is defined as:

$$(R_\Lambda)_{ij}) = \frac{1}{d} Tr(P_i \Lambda(P_j)), \tag{34}$$

where Λ is a quantum operation, and P_i and P_j are the Pauli matrices. All elements of R_Λ are real numbers and range between [-1,1], and $d = 2^n$ where n is the number of qubits. Since the PTM matrices are all real, they can be easily visualized. The figures show that the matrix estimates appear to be good estimates of the actual logic gates and that as the number of samples N increases, the gate estimates get better.

5 Discussion

The comparisons between the maximum likelihood estimates and the Bayesian estimates, in both the plots of the PTM matrices and the fidelity, relative fidelity error, and trace distance indicate that the estimates produced by the Bayesian approach are strong estimates of logic gates, and get better as the error decreases. The Bayesian estimates for N=500 were shown to have very similar measures to the MLE with N measurements. Given that logic gates have been produced that have failure probabilities of around 2 x10^{-5} (Brown et al., 2011), and Blume-Kohout et al. (2013) required N=1900 to perform ML-GST, Bayesian estimation is a very useful approach. Additionally, this approach naturally produces error bars, which are not in the ML approach. To accomplish this, some type of resampling method, like bootstrapping or jack knifing, would have to be performed.

A problem with this Bayesian approach is that there is no way to enforce the conditions ensuring that the logic gates estimated in Stan are physical. As a result, the trace distance may not be the most revealing metric. This can be observed in that trace distance and fidelity do not appear to correlate strongly. Additionally, the standard deviations among the trace distance measurements and relative error differ greatly. Further investigation is necessary to understand the large fluctuation in the size of the error bars, as well as to investigate whether different priors would create a more suitable model. If Stan or an alternative piece of Bayesian software capable of handling complex numbers or enforcing those constraints.

References

Aitchison, J., & Dunsmore, I. R. (1980). *Statistical prediction analysis.* Cambridge University Press.

Bengtsson, H., & Riedy, J. (2013). R. matlab: Read and write of mat files together with r-to-matlab connectivity. *R package version*, *1*(3).

Blume-Kohout, R., Gamble, J. K., Nielsen, E., Mizrahi, J., Sterk, J. D., & Maunz, P. (2013). Robust, self-consistent, closed-form tomography of quantum logic gates on a trapped ion qubit. *arXiv preprint arXiv:1310.4492*.

Brown, K., Wilson, A., Colombe, Y., Ospelkaus, C., Meier, A., Knill, E., . . . Wineland, D. (2011). Single-qubit-gate error below 10- 4 in a trapped ion. *Physical Review A*, *84*(3), 030303.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, *195*(2), 216–222.

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18. Retrieved from `http://www.jstatsoft.org/v40/i08/`

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2014). *Bayesian data analysis* (Vol. 2). Taylor & Francis.

Grant, M., Boyd, S., & Ye, Y. (2008). *Cvx: Matlab software for disciplined convex programming.*

Higham, N. (1998). Factorizing complex symmetric matrices with positive definite real and imaginary parts. *Mathematics of Computation of the American Mathematical Society*, *67*(224), 1591–1599.

Lewandowski, D., Kurowicka, D., & Joe, H. (2009). Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis*, *100*(9), 1989–2001.

MATLAB. (2015). *version 8.5.0 (r2015a).* Natick, Massachusetts: The MathWorks Inc.

Neal, R. M. (1994). An improved acceptance procedure for the hybrid monte carlo algorithm. *Journal of Computational Physics*, *111*(1), 194–203.

Neal, R. M. (2011). Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, *2*.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information.* Cambridge university press.

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). Coda: Convergence diagnosis and output analysis for mcmc. *R News*, *6*(1), 7–11. Retrieved from `http://CRAN.R-project.org/doc/Rnews/`

R Core Team. (2012). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org`

Sklyar, O., Murdoch, D., & Smith, M. (2007). *inline: Inline c, c++, fortran function calls from r. r package version 0.3. 3.*

Stan Development Team. (2015a). *Rstan: the r interface to stan, version 2.7.0.* Retrieved from `http://mc-stan.org/rstan.html`

Stan Development Team. (2015b). *Stan: A c++ library for probability and sampling, version 2.7.0.* Retrieved from `http://mc-stan.org/`

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with s* (Fourth ed.). New York: Springer. Retrieved from `http://www.stats.ox.ac.uk/pub/MASS4` (ISBN 0-387-95457-0)

Xavier Fernandez i Marin. (n.d.). *ggmcmc: Graphical tools for analyzing markov chain monte carlo simulations from bayesian inference.* Retrieved from `http://xavier-fim.net/packages/ggmcmc`