Construction of Quantum Process Matrices with Monte Carlo

Kevin Young
Sandia National Laboratories

June 22, 2011

1 Introduction

This document discusses the computation of quantum process matrices for one- and two-qubit systems evolving under the action of classical controls and classical noise.

Quantum process matrix

We define the process matrix, χ, for a quantum operation, $\rho \to \mathcal{K}_t[\rho]$, through

$$\mathcal{K}_t[\rho] = \sum_{i,j} \chi_{ij}(t) E_i \rho E_j^\dagger.$$

Here the set $\{E_i\}_{i=1}^{N^2}$ is a basis for $SU(N)$, where N is the Hilbert space dimension of ρ. For a single qubit, we chose an orthonormal basis under the Hilbert-Schmidt inner product,

$$\{E\}_1 = \frac{1}{\sqrt{2}}\{I, \sigma_x, \sigma_y, \sigma_z\},$$

the Pauli matrices and the identity. For two qubits, we take the direct product of the single qubit basis with itself:

$$\{E\}_2 = \{E\}_1 \otimes \{E\}_1 = \frac{1}{2}\{I\otimes I, I\otimes\sigma_x, I\otimes\sigma_y, \ldots, \sigma_x\otimes I, \sigma_x\otimes\sigma_x, \ldots, \sigma_z\otimes\sigma_z\},$$

Thus the single qubit process matrix, $\chi^{(1)}$, is a 4×4 matrix.

Calculating the process matrix

In the absence of noise, our system dynamics can be described entirely by a Hamiltonian which depends parametrically on a set of control fields,

$$H(t) = H(t; \{c(t)\})$$

Together with the Schrodinger equation, the Hamiltonian provides a map from the space of control fields to the space of unitary operators on quantum states,

$$U(t) = \mathcal{T}\exp\left(-i\int_0^t H(s; \{C(s)\})ds\right)$$

Here, $\mathcal{T}$ is the Dyson time-ordering operator. Under such unitary evolution, the density matrix evolves as:

$$\mathcal{K}_t[\rho(0)] = U(t)\rho(0)U^\dagger(t).$$

Because the matrices $\{E_i\}$ form a basis for $SU(N)$, we can express the unitary evolution matrix as,

$$U(t) = \sum_{i=1}^{N^2} a_i(t) E_i$$

So the density matrix evolution becomes,

$$\mathcal{K}_t[\rho(0)] = \sum_{i,j} a_i(t) a_j^*(t) E_i \rho(0) E_j \equiv \sum_{i,j=1}^{N^2} \chi_{ij} E_i \rho(0) E_j$$

Which defines the process matrix for unitary evolution in terms of the expansion coefficients $\{a_i\}_{i=1}^{N^2}$.

In the presence of noise (external or due to uncertainties in the control-field), the Hamiltonian becomes a function of time parameterized by both the control fields, $\{c(t)\}$, as well as a set of unmeasured stochastic parameters, $\{\Gamma(t)\}$,

$$H(t) = H(t; \{c(t)\}, \{\Gamma(t)\}).$$

To compute the final state of the system, one must average over all possible noise realizations,

$$\mathcal{K}_t[\rho(0)] = \left\langle U(t; \{\Gamma(t)\})\ \rho(0)\ U^\dagger(t; \{\Gamma(t)\}) \right\rangle_\Gamma .$$

Where we have used the notation $\langle \cdot \rangle$ to be the statistical average over the noise realizations, $\{\Gamma\}$. To perform this average, we resort to Monte Carlo techniques. The Monte Carlo approach proceeds by choosing an instance of the noise (in accordance to its statistical properties) then integrating the resulting Schrödinger equation to produce a process matrix. The procedure is repeated many times and the resulting process matrices are averaged.

$$\mathcal{K}_t[\rho] = \frac{1}{M} \sum_r U_r(t) \rho U_r^\dagger(t) = \frac{1}{M} \sum_r \left(\sum_{i,j} \chi_{i,j}^{(r)}(t) E_i \rho E_j \right) = \sum_{i,j} \left(\frac{1}{M} \sum_r \chi_{ij}^{(r)}(t) \right) E_i \rho E_j^\dagger = \sum_{i,j} \bar{\chi}_{ij}(t) E_i \rho E_j^\dagger$$

So we approximate the process matrix averaged over the noise instances as

$$\langle \chi(t) \rangle \simeq \bar{\chi}(t) \equiv \frac{1}{M} \sum_{r=1}^{M} \chi^{(r)}(t)$$

Simulating stochastic processes

The noise processes, $\{\Gamma(t)\}$, discussed above are modeled as discrete time, Gaussian, wide-sense stationary stochastic processes. By this we mean the the unconditioned probability distribution of the $\Gamma(t)$ is Gaussian (we choose the mean to be zero, $\langle \Gamma(t) \rangle = 0$), and the correlation function depends only on the difference in sampled times:

$$\langle \Gamma(t_1)\Gamma(t_2) \rangle_\Gamma = \langle \Gamma(0)\Gamma(|t_2 - t_1|) \rangle_\Gamma \equiv R_\Gamma(|t_2 - t_1|)$$

By the Wiener-Khinchin theorem, we can relate this correlation function to the power spectral density,

$$S_\Gamma(\omega) = \int_{-\infty}^{\infty} R_\Gamma(t) e^{-i\omega t} dt$$

$$R_\Gamma(t) = \int_{-\infty}^{\infty} C_\Gamma(t) e^{-i\omega t} dt$$

We simulate such processes using a Karhunen-Loéve (KL) filter. The KL filter constructs a discrete-time noise vector, $\mathbf{z}$, whose correlation matrix (autocovariance) is given by

$$\langle z_i z_j \rangle = R_\Gamma(|j - i|\, \delta_t) \equiv A_{ij}$$

where the time-descretization is chosen as δ_t. To accomplish this, we note that the correlation function, $R_\Gamma(t)$, is a real, symmetric, positive function, so A is a positive semi-definite Toeplitz matrix, so may be diagonalized by a real orthogaonal matrix, M, as

$$A = M\Lambda M^\mathsf{T}$$

with Λ a diagonal matrix of non-negative numbers. Consider a vector, $\mathbf{x}$, with each element drawn iid from a normal distribution, $P(x_i) = \mathcal{N}(x_i; \mu = 0, \sigma = 1)$. From this, define the vector $\mathbf{z}$ as,

$$z_i = M_{ij}\sqrt{\Lambda_j}x_j$$

Here we have used the abbreviated notation, $\Lambda_j \equiv \Lambda_{jj}$ to refer to the diagonal elements of Λ. The expected mean of this new process is

$$\langle z_i \rangle = M_{ij}\sqrt{\Lambda_j}\,\langle x_j \rangle = 0$$

and its expected autocovariance is

$$\begin{aligned}
\langle z_i z_j \rangle &= \sum_{m,n} M_{im}\sqrt{\Lambda_m}M_{jn}\sqrt{\Lambda_n}\,\langle x_m x_n \rangle \\
&= \sum_{m,n} M_{im}\sqrt{\Lambda_m}M_{jn}\sqrt{\Lambda_n}\delta_{mn} \\
&= \sum_{m} M_{im}\Lambda_m M_{jm} \\
&= (M\Lambda M^\mathsf{T})_{ij} \\
&= A_{ij}
\end{aligned}$$

Evolution Superoperator

Some physical processes, such as spontaneous emission, cannot be accurately modeled by averaging over noisy Hamiltonian evolution. In such a case, we appeal to the Lindblad equation, the most general form of Markovian master equation which is both trace-preserving and completely-positive,

$$\dot{\rho} = -\frac{i}{\hbar}[H, \rho] + \sum_i \gamma_i \left(L_i \rho L_i^\dagger - \frac{1}{2} L_i^\dagger L_i \rho - \frac{1}{2} \rho L_i L_i^\dagger \right)$$

To compute the process matrix under such evolution, we proceed by vectorizing the density matrix,

$$\rho = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1N} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N1} & \rho_{N2} & \cdots & \rho_{NN} \end{pmatrix} \rightarrow \tilde{\rho} = \begin{pmatrix} \rho_{11} \\ \vdots \\ \rho_{N1} \\ \rho_{12} \\ \vdots \\ \rho_{N2} \\ \vdots \\ \rho_{NN} \end{pmatrix}$$

The left and right action of an operator on the density matrix transforms as,

$$A\rho \rightarrow \widetilde{A\rho} = (I \otimes A)\,\tilde{\rho}$$

$$\rho A \rightarrow \widetilde{\rho A} = \left(A^\mathsf{T} \otimes I\right)\tilde{\rho}$$

Thus the Lindblad evolution equation can be written as

$$\dot{\tilde{\rho}} = \left[\frac{-i}{\hbar} \left(I \otimes H - H^{\mathsf{T}} \otimes I \right) + \sum_i \gamma_i \left(L_i^{\mathsf{T}} \otimes L_i^{\dagger} - \frac{1}{2} I \otimes L_i^{\dagger} L_i - \frac{1}{2} \left(L_i^{\dagger} L_i \right)^{\mathsf{T}} \otimes I \right) \right] \tilde{\rho} = \tilde{\mathcal{L}} \tilde{\rho}(0)$$

This can be exponentiated to give,

$$\tilde{\rho}(t) = e^{\tilde{\mathcal{L}} t} \tilde{\rho}(0).$$

Averages over stochastic processes can then be taken,

$$\langle \tilde{\rho}(t) \rangle_{\{\Gamma\}} = \left[\frac{1}{M} \sum_{r=1}^{M} e^{\tilde{\mathcal{L}}_r t} \right] \tilde{\rho}(0),$$

and the process matrix computed,

$$\bar{\chi}_{ij}(t) = \mathrm{tr} \left(\left(\frac{1}{N} \sum_{k=1}^{N} e^{\tilde{\mathcal{L}}_k t} \right) \left(E_j^{\dagger} \otimes E_i \right) \right).$$

Python code to compute process matrix

Example Python code implementing the above Monte Carlo procedure will be made available by Sandia National Laboratories. The algorithm is implemented as `processMC.py`, a Python routine written using the SciPy libraries. The code is self documenting using documentation strings (docstrings). For each of the Solid State, Neutral Atom, and Trapped Ion PMDs, and for each of the basic quantum error correction gates (X,Z, S, T, H, and CPHASE, CNOT, or GEOPHASE), a separate script has been written, also in Python. Each script defines the relevant Hamiltonian, control fields, and noise terms and then computes the process matrix for the target gate. For example, the following terminal command will compute the process matrix for the X gate using the solid state PMD.

```
>> python solidstate_X.py
```

The process matrix will be written to the file, `process_matrix_solidstate_X.out`