MPHYG002: Assignment 1
Generating Stabilizer Groups with C++

Padraic Calpin
PHDCA33

1 Stabilizer Groups in Quantum Information

The Stabilizer group is an important concept in quantum information theory. They were first developed by Gottesman, as a technique for studying quantum error correcting codes[1], but the formalism has found applications in a wide range of contexts in the field[2].

Stabilizers are built up out of Kronecker products of Pauli operators, a set of operators that occur frequently in the study of quantum theory as they are also the generating elements of all single qubit Unitary matrices[3]

$$U = e^{i\sum_{j=1}^{3}\theta_j\sigma_j} : \; UU^\dagger = \mathbb{I}$$

where

$$\sigma_1 \equiv X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

$$\sigma_2 \equiv Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2}$$

$$\sigma_3 \equiv Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{3}$$

$$\tag{4}$$

These matrices, and their tensor products, for a closed set under multiplication and are all self-inverse : $U^2 = \mathbb{I}$, and thus they define a group called the Pauli group $\mathcal{P}$. A Stabilizer group $\mathcal{S}$ is then defined as a subgroup of the Pauli group, with a particular property that $\left[s_i, s_j\right] \equiv s_i s_j - s_j s_i = 0 \forall s_i, s_j \in \mathcal{S}$.

This commutation property is significant due to a property of observables in quantum mechanics. Any unitary matrix that is also Hermitian, i.e. $U^\dagger = U$, defines a measurable property of the system, and commuting observables can be simultaneously observed. Each observable $s_i \in \mathcal{S}$ defines a linear constraint on the system, such that a Stabilizer group with 2^n elements acting on n qubits uniquely defines a quantum state $|\psi_\mathcal{S}\rangle$[4].

These states, called Stabilizer states, are of particular interest to Quantum information scientists. They exhibit many 'quantum' features such as entanglement, but as a result of the properties of the Pauli group they can also be efficiently described classically. In my project, we use a decomposition of arbitrary quantum states into stabilizer states as a way to tackle simulation of quantum systems. Thus, having access to the set of Stabilizer states on n qubits is of great interest.

1.1 Computational Representation of Stabilizer States

In the following section we will denote a tensor product of different Pauli matrices as a 'Pauli Literal', an n-letter string drawn from the alphabet "I,X,Y,Z", made up of the Pauli matrices and the two-qubit Identity matrix. Up to a complex phase, which is ignored in quantum mechanics, we can use a relation that $Y \approx XZ$ to write an n-qubit Pauli Matrix as a $2n$ bit string $\mathbf{s}$, such that the resulting matrix is written [5].

$$P_{\mathbf{s}} = \bigotimes_{i=1}^{n} X^{s_{i+n}} Z^{s_i}$$

The commutation requirement can be encoded as requiring the Symplectic inner product between two matrices to be equal to 0. Defining the first n bits of $\mathbf{s}$ as the 'Z-bits', and the other n as the 'x' bits, this inner product is given by [5]

$$S(P_{\mathbf{s}}, P_{\mathbf{s}'}) = \left(\sum_{i=1}^{n} x_i z'_i \oplus z_i x'_i \right) \mod_2.$$

This is commonly called the 'Symplectic' representation of a Pauli operator. It has some other conveniences, such as matrix multiplication being given by XORing the two bitstrings s, s'.

In this programme, the class `SymplecticPauli` implements the representation, using the Boost `dynamic_bitset` class to hold the 'x' and 'z' bit strings as it provides convenience methods for the XOR and AND operations. We can generate all n qubit Pauli operators by converting the numbers between $[0, 2^n - 1]$ to bit strings.

It can be proven that any Pauli with 2^n elements can be fully described by a set of n linearly independent generators. As a convenience, to avoid implementing a form of Gaussian elimination on the bitsets, we instead define a class `Stabilizer Group`, which is composed of two unordered_sets callled `generators` and `members`. Each time we try and add an element to the set, if it is unique we then try and generate all new members in the set by recursively multiplying it against the other Pauli operators in the set, and add it to the set of generators. This guarantees that we end up with linearly independent Pauli operators in the `generators` member. We can then verify our Stabilizer group by simply checking it's order $\|\mathcal{S}\| = 2^n$ for n qubits.

2 StabilizerCPP

The StabilizerCPP project is unfortunately not currently complete. Currently missing is code for saving a given Stabilizer group to a file, and a command line interface for the main executable. However, the code has been tested for generating the Stabilizer states on 2 and 3 qubits. The code is demonstrably faster than an existing python implementation, taking ≈ 3 seconds to generate the 135 unqique 3-qubit stabilzier groups, compared to $30s$. The number of Stabilizer states grows exponentially in n, so we might expect this advantage to grow with n.

The code has been built successfully on Windows 10 using MinGW, and Mac OSX using brewed gcc, boost and Eigen installs. The testing was implemented using google_test, which is contained as a submodule of the repository. This code is currently missing release mode optimizations and a backup SuperBuild option. Eigen is included as a dependency for generating the corresponding matrices of the Symplectic Paulis, a necessary step in extending the code to return the associated Stabilizer States $|\psi_{\mathcal{S}}\rangle$ themselves.

References

[1] D. Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, **2008quant-ph/9705052**, 114 (1997). quant-ph/9705052.

[2] D. Gottesman. The Heisenberg Representation of Quantum Computers. *Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, **1quant-ph/9807006**, 32 (1999). quant-ph/9807006.

[3] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information* (2000).

[4] H. J. Garcia, I. L. Markov, and A. W. Cross. Efficient Inner-product Algorithm for Stabilizer States. *arXiv preprint arXiv:1210.6646*, pages 1–14 (2012). 1210.6646.

[5] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Physical Review A - Atomic, Molecular, and Optical Physics*, **70quant-ph/0406196**, 1 (2004). quant-ph/0406196.