Stabilizer Codes

Stabilizer codes are a class of quantum codes with construction based on the stabilizer formalism. The idea was put forward by Gottesman (1996, 1997, 1998).

Make sure that the Q3 package is loaded to use the demonstrations in this documentation.

In[•]:=
```
<< Q3`
```

Throughout this document, symbol **s** will be used to denote qubits and Pauli operators on them. Similarly, symbol c will be used to denote complex-valued coefficients.

In[•]:=
```
Let[Qubit, S]
Let[Complex, c]
```


Error-Correction Conditions

Let $\mathcal{S}$ be the stabilizer of a code space $\mathcal{V}$. Suppose that $\mathcal{V}$ is corrupted by an error operator E in the Pauli group $\mathcal{P}(n)$. What would happen to the code space?

There are three distinctive cases:

1. E anti-commutes with one of the generators, say G, of the stabilizer $\mathcal{S}$.

In this case, E takes the code space $\mathcal{V}$ to a subspace orthogonal to $\mathcal{V}$ (see the Quantum Workbook for a proof). Therefore, in this case, one can detect the error with proper measurement (see below), and correct it.

2. E belongs to the stabilizer $\mathcal{S}$.

In this case, E itself stabilizes the code space, and the code space remains intact without any corruption.

3. E commutes with all all elements of the stabilizer, but does not belong to the stabilizer.

This is the most dangerous case. According to stabilizer formalism of measurement, measurement of such an operator E would reveal some information of the quantum states in the code space. This implies that E breaks the code space, and the corruption cannot be corrected.

The above observation implies the following proposition on the error-correction conditions of stabilizer codes.

Proposition: Let $\mathcal{S}$ be the stabilizer of a code space $\mathcal{V}$. Suppose that the errors are described by a set $\{E_\mu\}$ of operators in the Pauli group $\mathcal{P}(n)$. The necessary and sufficient condition for the code to correct the errors is that for all μ and ν, $E_\mu^\dagger E_\nu$ either belongs to $\mathcal{S}$ or anti-commutes with at least one generator of $\mathcal{S}$.

Error-Syndrome Detection and Recovery

Let $\{G_1, G_2, \ldots, G_k\}$ be a set of independent generators of the stabilizer $\mathcal{S}$. Suppose that $\{E_\mu\}$ is the set of error operators that are correctable by the code.

Error Syndrome

The first step is to detect error syndrome. This is achieved by **measuring the generators $G_1, G_2, \ldots, G_k$**. Each outcome ± 1 serves as a symptom of the errors. If an error E^μ occurs, then it causes a **syndrome $\epsilon^\mu := (\epsilon_1^\mu, \epsilon_2^\mu, \ldots, \epsilon_k^\mu)$ with $\epsilon_j^\mu = \pm 1$**.

The error and the syndrome are related by

$$E_\mu G_j E_\mu^\dagger = \epsilon_j^\mu G_j.$$

In short, detecting error-syndrome ϵ^μ by measurement of the generators $G_1, G_2, \ldots, G_k$ enables one to identify the associated error E_μ.

Recovery

1. When a syndrome ϵ^μ is associated with a single error E_μ, one can correct it simply by applying $E_\mu^\dagger$.

2. However, in general, two different errors E_μ and E_ν may cause the same syndrome, $\epsilon^\mu = \epsilon^\nu$.

In this case, we have

$$E_\mu P E_\mu^\dagger = E_\nu P E_\nu^\dagger,$$

where P is the projection onto the code space $\mathcal{V}$. This leads to

$$(E_\mu^\dagger E_\nu) P (E_\mu^\dagger E_\nu)^\dagger = P .$$

This means that $E_\mu^\dagger E_\nu$ is a member of the stabilizer $\mathcal{S}$. In other words, even if it was the error E_ν that actually occurred, it can still be corrected by applying $E_\mu^\dagger$.

In short, however many errors are associated with a given syndrome, one can choose any operator E_μ among them to apply $E_\mu^\dagger$ to recover the original state.

Encoding

We want to prepare a system of n physical qubits, say, in the logical basis state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$. This requires a procedure **without referring to an explicit form of the logical basis states**.

Encoding and decoding a general stabilizer code using unitary gates are possible and explained in detail in Cleve and Gottesman (1997). Here a **measurement-based method** is introduced as it is essentially the same as the error-syndrome detection and recovery procedure.

Summary: To measure the generators $G_1, G_2, \ldots, G_k$ and the logical operators $\overline{Z}_1, \overline{Z}_2, \ldots, \overline{Z}_{n-k}$.

Observations

1. The combined set $\{G_1, G_2, \ldots, G_k, \overline{Z}_1, \overline{Z}_2, \ldots, \overline{Z}_{n-k}\}$ consists of all mutually commuting and independent operators. It thus generates a stabilizer subgroup stabilizing a one-dimensional subspace spanned by nothing but the desired state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$.
2. Therefore, one can regard a measurement of the generators and logical operators as the error syndrome detection. Then, an initialization to the desired state corresponds to the recovery procedure.

Encoding Procedures

1. Each measurement of $G_1, G_2, \ldots, G_k$ and $\overline{Z}_1, \overline{Z}_2, \ldots, \overline{Z}_{n-k}$ yields measurement outcome ± 1.

Depending on the measurement results, the final state $\left|\psi\right\rangle$ after all measurements is stabilized by $\pm G_1, \pm G_2, \ldots, \pm G_k, \pm\overline{Z}_1, \pm\overline{Z}_2, \ldots, \pm\overline{Z}_{n-k}$.

2. For each observable M with measurement outcome -1, find a Pauli operator G that anti-commutes with only M but commutes with the rest and apply G on $\left|\psi\right\rangle$.

Recall that there exists a Pauli operator G that anti-commutes with only a particular one of the independent operators but commutes with all others.

3. Once the state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$ has been encoded, other logical basis states can be obtained by applying the logical operators $\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_{n-k}$ as necessary.

Examples

Bit-Flip Code

Consider the bit-flip code. The stabilizer $\mathcal{S}$ of the code space is generated by these generators.

```
In[•]:= gnr = {S[1, 3] ** S[2, 3], S[2, 3] ** S[3, 3]};
        gnr // PauliForm
Out[•]= {Z ⊗ Z ⊗ I, I ⊗ Z ⊗ Z}
```

This is a set $\{E_\mu\}$ of error operators correctable by the code.

```
In[•]:= err = {1, S[1, 1], S[2, 1], S[3, 1]};
        err // PauliForm
Out[•]= {I⊗I⊗I, X⊗I⊗I, I⊗X⊗I, I⊗I⊗X}
```

To check if the errors are indeed correctable by the code, we need to examine all combinations $E_\mu^\dagger E_\nu$.

```
In[•]:= chk = Union[Multiply @@@ Choices[err, {2}]];
        chk // PauliForm
Out[•]= {I⊗I⊗I, X⊗X⊗I, X⊗I⊗X, I⊗X⊗X, X⊗I⊗I, I⊗X⊗I, I⊗I⊗X}
```

As one can see, there is only one combination, **I⊗I⊗I**, of the error operators that commutes with all generators of the stabilizer. But it belongs to the stabilizer.

```
In[•]:= mat = Outer[GottesmanTest, gnr, chk];
        TableForm[mat, TableHeadings → {PauliForm@gnr, PauliForm@chk},
         TableAlignments → Right]
```

Out[•]//TableForm=

	I⊗I⊗I	X⊗X⊗I	X⊗I⊗X	I⊗X⊗X	X⊗I⊗I	I⊗X⊗I	I⊗I⊗X
Z⊗Z⊗I	1	1	−1	−1	−1	−1	1
I⊗Z⊗Z	1	−1	−1	1	1	−1	−1

This table displays the error syndrome of the bit-flip correction code.

```
In[•]:= syndrome = Map[(# ** gnr ** #) / gnr &, err];
        TableForm[syndrome, TableAlignments → Right,
         TableHeadings → {PauliForm@err, PauliForm@gnr}]
```

Out[•]//TableForm=

	Z⊗Z⊗I	I⊗Z⊗Z
I⊗I⊗I	1	1
X⊗I⊗I	−1	1
I⊗X⊗I	−1	−1
I⊗I⊗X	1	−1

Phase-Flip Code

Next, consider the phase-flip code. The stabilizer of the code space is generated by these generators.

```
In[•]:= gnr = {S[1, 1] ** S[2, 1], S[2, 1] ** S[3, 1]};
        gnr // PauliForm
Out[•]= {X⊗X⊗I, I⊗X⊗X}
```

This is a set $\{E_\mu\}$ of error operators correctable by the code.

```
In[•]:= err = {1, S[1, 3], S[2, 3], S[3, 3]};
        err // PauliForm
Out[•]= {I⊗I⊗I, Z⊗I⊗I, I⊗Z⊗I, I⊗I⊗Z}
```

To check if the errors are indeed correctable by the code, we need to examine all combinations $E_\mu^\dagger E_\nu$.

```
In[•]:= chk = Union[Multiply @@@ Choices[err, {2}]];
        chk // PauliForm
Out[•]= {I⊗I⊗I, Z⊗Z⊗I, Z⊗I⊗Z, I⊗Z⊗Z, Z⊗I⊗I, I⊗Z⊗I, I⊗I⊗Z}
```

As one can see, there is only one combination, **I⊗I⊗I**, of the error operators that commutes with all generators of the stabilizer. But it belongs to the stabilizer.

```
In[•]:= mat = Outer[GottesmanTest, gnr, chk];
        TableForm[mat, TableHeadings → {PauliForm@gnr, PauliForm@chk},
         TableAlignments → Right]
```

Out[•]//TableForm=

	I⊗I⊗I	Z⊗Z⊗I	Z⊗I⊗Z	I⊗Z⊗Z	Z⊗I⊗I	I⊗Z⊗I	I⊗I⊗Z
X⊗X⊗I	1	1	−1	−1	−1	−1	1
I⊗X⊗X	1	−1	−1	1	1	−1	−1

This table displays the error syndrome of the phase-flip correction code.

```
In[•]:= syndrome = Map[(# ** gnr ** #) / gnr &, err];
        TableForm[syndrome, TableAlignments → Right,
         TableHeadings → {PauliForm@err, PauliForm@gnr}]
```

Out[•]//TableForm=

	X⊗X⊗I	I⊗X⊗X
I⊗I⊗I	1	1
Z⊗I⊗I	−1	1
I⊗Z⊗I	−1	−1
I⊗I⊗Z	1	−1

Nine-Qubit Code

Shor's code needs 9 physical qubits to encode a single-qubit quantum states.

```
In[•]:= jj = Range[9];
```

This is a generating set of the stabilizer of the code.

```
In[•]:= gnr = {
          S[1, 3] ** S[2, 3], S[2, 3] ** S[3, 3],
          S[4, 3] ** S[5, 3], S[5, 3] ** S[6, 3],
          S[7, 3] ** S[8, 3], S[8, 3] ** S[9, 3],
          Multiply @@ S[{1, 2, 3, 4, 5, 6}, 1],
          Multiply @@ S[{4, 5, 6, 7, 8, 9}, 1]}
Out[•]= {S_1^z S_2^z, S_2^z S_3^z, S_4^z S_5^z, S_5^z S_6^z, S_7^z S_8^z, S_8^z S_9^z, S_1^x S_2^x S_3^x S_4^x S_5^x S_6^x, S_4^x S_5^x S_6^x S_7^x S_8^x S_9^x}
```

These are logical operators of the code.

```
In[•]:= opX = Multiply @@ S[jj, 1]
        opZ = Multiply @@ S[jj, 3]
Out[•]= S_1^x S_2^x S_3^x S_4^x S_5^x S_6^x S_7^x S_8^x S_9^x
Out[•]= S_1^z S_2^z S_3^z S_4^z S_5^z S_6^z S_7^z S_8^z S_9^z
```

These are the error operators E_μ correctable by the code.

```
In[•]:= err = Prepend[S[jj, All], 1]
```

Out[•]=
$$\{1, S_1^x, S_1^y, S_1^z, S_2^x, S_2^y, S_2^z, S_3^x, S_3^y, S_3^z, S_4^x, S_4^y, S_4^z, S_5^x, S_5^y, S_5^z, S_6^x, S_6^y, S_6^z, S_7^x, S_7^y, S_7^z, S_8^x, S_8^y, S_8^z, S_9^x, S_9^y, S_9^z\}$$

To check if the errors are indeed correctable by the code, we need to examine all combinations $E_\mu^\dagger E_\nu$.

```
In[•]:= chk = Union[Multiply @@@ Choices[err, {2}]];
        Length[chk]
```

Out[•]= 379

This shows a partial list of those combinations.

```
In[•]:= PauliForm[chk[[ ;; 8]], S@jj] // TableForm
```

Out[•]//TableForm=

$I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes X\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes Y\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes Z\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes I\otimes X\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes I\otimes Y\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes I\otimes Z\otimes I\otimes I\otimes I\otimes I\otimes I\otimes I$
$X\otimes I\otimes I\otimes X\otimes I\otimes I\otimes I\otimes I\otimes I$

Examine the commutation of the above combinations with the generators of the stabilizer.

```
In[•]:= Timing[mat = Outer[GottesmanTest, chk, gnr];]
```

Out[•]= {5.34516, Null}

This displays a *part* of the commutation relation. One can see that each of them either commutes or anti-commutes with the generators.

```
In[•]:= TableForm[mat[[ ;; 5, ;; 6]],
          TableAlignments → Right, TableHeadings → {chk[[ ;; 5]], gnr}]
```

Out[•]//TableForm=

	$S_1^z S_2^z$	$S_2^z S_3^z$	$S_4^z S_5^z$	$S_5^z S_6^z$	$S_7^z S_8^z$	$S_8^z S_9^z$
1	1	1	1	1	1	1
$S_1^x S_2^x$	1	−1	1	1	1	1
$S_1^x S_2^y$	1	−1	1	1	1	1
$S_1^x S_2^z$	−1	1	1	1	1	1
$S_1^x S_3^x$	−1	−1	1	1	1	1

The most dangerous case is that the check operator commutes with all of the generators but does not belong to the stabilizer. To check if there is such case, we examine the check operators that commute with all of the generators.

```
In[•]:= kk = Catenate@MapIndexed[If[ContainsOnly[#1, {1}], #2, Nothing] &, mat]
```

Out[•]= {1, 52, 55, 118, 223, 226, 262, 313, 316, 325}

This shows that such check operators actually belong to the stabilizer. The considered errors are thus correctable by the code.

```
In[•]:= chk[[kk]]
```

Out[•]= $\{1, S_1^z S_2^z, S_1^z S_3^z, S_2^z S_3^z, S_4^z S_5^z, S_4^z S_6^z, S_5^z S_6^z, S_7^z S_8^z, S_7^z S_9^z, S_8^z S_9^z\}$

This table displays a *part* of the error syndrome of the code.

```
In[•]:= syndrome = Map[(# ** gnr ** #) / gnr &, err];
        Style[TableForm[syndrome[[ ;; 8]],
          TableAlignments → Right, TableHeadings → {err, gnr}], Small]
```

Out[•]=

	$S_1^z S_2^z$	$S_2^z S_3^z$	$S_4^z S_5^z$	$S_5^z S_6^z$	$S_7^z S_8^z$	$S_8^z S_9^z$	$S_1^x S_2^x S_3^x S_4^x S_5^x S_6^x$	$S_4^x S_5^x S_6^x S_7^x S_8^x S_9^x$
1	1	1	1	1	1	1	1	1
S_1^x	−1	1	1	1	1	1	1	1
S_1^y	−1	1	1	1	1	1	−1	1
S_1^z	1	1	1	1	1	1	−1	1
S_2^x	−1	−1	1	1	1	1	1	1
S_2^y	−1	−1	1	1	1	1	−1	1
S_2^z	1	1	1	1	1	1	−1	1
S_3^x	1	−1	1	1	1	1	1	1

Five-Qubit Code

We consider a code encoded on 5 physical qubits.

```
In[•]:= jj = Range[5];
```

Here are the generators of the stabilizer of the five-qubit code.

```
In[•]:= gnr = {
            S[1, 1] ** S[2, 3] ** S[3, 3] ** S[4, 1],
            S[2, 1] ** S[3, 3] ** S[4, 3] ** S[5, 1],
            S[1, 1] ** S[3, 1] ** S[4, 3] ** S[5, 3],
            S[1, 3] ** S[2, 1] ** S[4, 1] ** S[5, 3]};
        PauliForm[gnr]
```

Out[•]= $\{X \otimes Z \otimes Z \otimes X \otimes I, I \otimes X \otimes Z \otimes Z \otimes X, X \otimes I \otimes X \otimes Z \otimes Z, Z \otimes X \otimes I \otimes X \otimes Z\}$

These are logical operators of the code.

```
In[•]:= opX = Multiply @@ S[jj, 1];
        opZ = Multiply @@ S[jj, 3];
        PauliForm[opX]
        PauliForm[opZ]
```

Out[•]= $X \otimes X \otimes X \otimes X \otimes X$

Out[•]= $Z \otimes Z \otimes Z \otimes Z \otimes Z$

Here are the error operators E_μ correctable by the code.

```
In[•]:= err = Prepend[S[jj, All], 1];
        PauliForm[err]
```

Out[•]= $\{I \otimes I \otimes I \otimes I \otimes I, X \otimes I \otimes I \otimes I \otimes I, Y \otimes I \otimes I \otimes I \otimes I, Z \otimes I \otimes I \otimes I \otimes I,$
$I \otimes X \otimes I \otimes I \otimes I, I \otimes Y \otimes I \otimes I \otimes I, I \otimes Z \otimes I \otimes I \otimes I, I \otimes I \otimes X \otimes I \otimes I,$
$I \otimes I \otimes Y \otimes I \otimes I, I \otimes I \otimes Z \otimes I \otimes I, I \otimes I \otimes I \otimes X \otimes I, I \otimes I \otimes I \otimes Y \otimes I,$
$I \otimes I \otimes I \otimes Z \otimes I, I \otimes I \otimes I \otimes I \otimes X, I \otimes I \otimes I \otimes I \otimes Y, I \otimes I \otimes I \otimes I \otimes Z\}$

To check if the errors are indeed correctable by the code, we need to examine all combinations

$E_\mu^\dagger E_\nu$.

```
In[•]:= chk = Union[Multiply @@@ Choices[err, {2}]];
        Length[chk]
Out[•]= 121
```

Examine the commutation of the above combinations with the generators of the stabilizer.

```
In[•]:= Timing[mat = Outer[GottesmanTest, chk, gnr];]
Out[•]= {1.10635, Null}
```

This displays a *part* of the commutation relations of some of the check operators. One can see that each of them either commutes or anti-commutes with the generators.

```
In[•]:= TableForm[mat[[ ;; 5]], TableAlignments → Right,
         TableHeadings → PauliForm@{chk[[ ;; 5]], gnr}]
```

Out[•]//TableForm=

	X⊗Z⊗Z⊗X⊗I	I⊗X⊗Z⊗Z⊗X	X⊗I⊗X⊗Z⊗Z	Z⊗X⊗I⊗X⊗Z
I⊗I⊗I⊗I⊗I	1	1	1	1
X⊗X⊗I⊗I⊗I	−1	1	1	−1
X⊗Y⊗I⊗I⊗I	−1	−1	1	1
X⊗Z⊗I⊗I⊗I	1	−1	1	1
X⊗I⊗X⊗I⊗I	−1	−1	1	−1

The most dangerous case is that the check operator commutes with all of the generators but does not belong to the stabilizer. To check if there is such case, we examine the check operators that commute with all of the generators.

```
In[•]:= kk = Catenate@MapIndexed[If[ContainsOnly[#1, {1}], #2, Nothing] &, mat]
Out[•]= {1}
```

This shows that such check operators actually belong to the stabilizer. The considered errors are thus correctable by the code.

```
In[•]:= chk[[kk]]
Out[•]= {1}
```

This shows the error syndrome of the code upon the measurement of the generators of the stabilizer.

In[•]:=
```
syndrome = Map[(# ** gnr ** #) / gnr &, err];
TableForm[syndrome, TableAlignments → Right,
 TableHeadings → {PauliForm@err, PauliForm@gnr}]
```

Out[•]//TableForm=

	X⊗Z⊗Z⊗X⊗I	I⊗X⊗Z⊗Z⊗X	X⊗I⊗X⊗Z⊗Z	Z⊗X⊗I⊗X⊗Z
I⊗I⊗I⊗I⊗I	1	1	1	1
X⊗I⊗I⊗I⊗I	1	1	1	−1
Y⊗I⊗I⊗I⊗I	−1	1	−1	−1
Z⊗I⊗I⊗I⊗I	−1	1	−1	1
I⊗X⊗I⊗I⊗I	−1	1	1	1
I⊗Y⊗I⊗I⊗I	−1	−1	1	−1
I⊗Z⊗I⊗I⊗I	1	−1	1	−1
I⊗I⊗X⊗I⊗I	−1	−1	1	1
I⊗I⊗Y⊗I⊗I	−1	−1	−1	1
I⊗I⊗Z⊗I⊗I	1	1	−1	1
I⊗I⊗I⊗X⊗I	1	−1	−1	1
I⊗I⊗I⊗Y⊗I	−1	−1	−1	−1
I⊗I⊗I⊗Z⊗I	−1	1	1	−1
I⊗I⊗I⊗I⊗X	1	1	−1	−1
I⊗I⊗I⊗I⊗Y	1	−1	−1	−1
I⊗I⊗I⊗I⊗Z	1	−1	1	1

Five-Qubit Code: Encoding

In this example, we follow the measurement-based procedure to encode logical basis state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$ in five physical qubits.

In[•]:=
```
jj = Range[5];
```

As we recall, this is a generating set of the stabilizer of the five-qubit code.

In[•]:=
```
gnr = {
    S[1, 1] ** S[2, 3] ** S[3, 3] ** S[4, 1],
    S[2, 1] ** S[3, 3] ** S[4, 3] ** S[5, 1],
    S[1, 1] ** S[3, 1] ** S[4, 3] ** S[5, 3],
    S[1, 3] ** S[2, 1] ** S[4, 1] ** S[5, 3]};
PauliForm[gnr] // TableForm
```

Out[•]//TableForm=
```
X⊗Z⊗Z⊗X⊗I
I⊗X⊗Z⊗Z⊗X
X⊗I⊗X⊗Z⊗Z
Z⊗X⊗I⊗X⊗Z
```

And these are logical operators of the code.

In[•]:=
```
opX = Multiply @@ S[jj, 1];
opZ = Multiply @@ S[jj, 3];
PauliForm@{opX, opZ} // TableForm
```

Out[•]//TableForm=
```
X⊗X⊗X⊗X⊗X
Z⊗Z⊗Z⊗Z⊗Z
```

In the measurement-based method, we will perform measurement of the following extended set $\{G_1, G_2, \ldots, G_4, \overline{Z}\}$ of generators. This set generates a subgroup stabilizing the one-dimensional

subspace, $P_1^+ P_2^+ \cdots P_5^+ \mathcal{H}$, where $P_j^\pm := \left(I \pm G_j\right)/2$ with $G_5 \equiv \overline{Z}$ are the projections and $\mathcal{H}$ is the Hilbert space associated with the five physical qubits. Subspace $P_1^+ P_2^+ \cdots P_5^+ \mathcal{H}$ is spanned by logical basis state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$.

In[•]:=
```
ext = Join[gnr, {opZ}];
ext // PauliForm // TableForm
```

Out[•]//TableForm=

$X \otimes Z \otimes Z \otimes X \otimes I$
$I \otimes X \otimes Z \otimes Z \otimes X$
$X \otimes I \otimes X \otimes Z \otimes Z$
$Z \otimes X \otimes I \otimes X \otimes Z$
$Z \otimes Z \otimes Z \otimes Z \otimes Z$

Prepare the five-qubit system in a linear superposition of all computational basis states. This state is useful because it is likely to have support in every eigenspace of the observables that we are going to measure below.

In[•]:=
```
in = Multiply @@ S[{1, 2, 3, 4, 5}, 6] ** Ket[];
in // SimpleForm
```

Out[•]=

$$\frac{|00000\rangle}{4\sqrt{2}} + \frac{|00001\rangle}{4\sqrt{2}} + \frac{|00010\rangle}{4\sqrt{2}} + \frac{|00011\rangle}{4\sqrt{2}} + \frac{|00100\rangle}{4\sqrt{2}} + \frac{|00101\rangle}{4\sqrt{2}} + \frac{|00110\rangle}{4\sqrt{2}} + \frac{|00111\rangle}{4\sqrt{2}} +$$
$$\frac{|01000\rangle}{4\sqrt{2}} + \frac{|01001\rangle}{4\sqrt{2}} + \frac{|01010\rangle}{4\sqrt{2}} + \frac{|01011\rangle}{4\sqrt{2}} + \frac{|01100\rangle}{4\sqrt{2}} + \frac{|01101\rangle}{4\sqrt{2}} + \frac{|01110\rangle}{4\sqrt{2}} + \frac{|01111\rangle}{4\sqrt{2}} +$$
$$\frac{|10000\rangle}{4\sqrt{2}} + \frac{|10001\rangle}{4\sqrt{2}} + \frac{|10010\rangle}{4\sqrt{2}} + \frac{|10011\rangle}{4\sqrt{2}} + \frac{|10100\rangle}{4\sqrt{2}} + \frac{|10101\rangle}{4\sqrt{2}} + \frac{|10110\rangle}{4\sqrt{2}} + \frac{|10111\rangle}{4\sqrt{2}} +$$
$$\frac{|11000\rangle}{4\sqrt{2}} + \frac{|11001\rangle}{4\sqrt{2}} + \frac{|11010\rangle}{4\sqrt{2}} + \frac{|11011\rangle}{4\sqrt{2}} + \frac{|11100\rangle}{4\sqrt{2}} + \frac{|11101\rangle}{4\sqrt{2}} + \frac{|11110\rangle}{4\sqrt{2}} + \frac{|11111\rangle}{4\sqrt{2}}$$

Now measure every observable in the extended set **ext** of generators. Instead of actually simulating the measurement procedure, here we will just assume that the measurement has yield an outcome (–1, 1, 1, 1, 1). That is, the post-measurement state belongs to the subspace $P_1^- P_2^+ \cdots P_5^+ \mathcal{H}$, instead of $P_1^+ P_2^+ \cdots P_5^+ \mathcal{H}$.

In[•]:=
```
prj = ((1 - First[ext]) / 2) ** Apply[Multiply, (1 + Rest[ext]) / 2];
out = KetNormalize[prj ** in];
out // SimpleForm
```

Out[•]=

$$\frac{|00000\rangle}{4} - \frac{|00011\rangle}{4} - \frac{|00101\rangle}{4} + \frac{|00110\rangle}{4} + \frac{|01001\rangle}{4} + \frac{|01010\rangle}{4} + \frac{|01100\rangle}{4} + \frac{|01111\rangle}{4} +$$
$$\frac{|10001\rangle}{4} - \frac{|10010\rangle}{4} + \frac{|10100\rangle}{4} - \frac{|10111\rangle}{4} + \frac{|11000\rangle}{4} + \frac{|11011\rangle}{4} - \frac{|11101\rangle}{4} - \frac{|11110\rangle}{4}$$

To bring the state from $P_1^- P_2^+ \cdots P_5^+ \mathcal{H}$ to $P_1^+ P_2^+ \cdots P_5^+ \mathcal{H}$, an additional operation is required. To do this, we note that $Z \otimes I \otimes Z \otimes I \otimes I$ commutes with all but the first element of the extended set **ext** of generators.

```
In[•]:= op = S[1, 3] ** S[3, 3];
        PauliForm[op, S@jj]
Out[•]= Z ⊗ I ⊗ Z ⊗ I ⊗ I
```

```
In[•]:= op ** ext ** Dagger[op] // PauliForm // TableForm
Out[•]//TableForm=
        -(X ⊗ Z ⊗ Z ⊗ X ⊗ I)
        I ⊗ X ⊗ Z ⊗ Z ⊗ X
        X ⊗ I ⊗ X ⊗ Z ⊗ Z
        Z ⊗ X ⊗ I ⊗ X ⊗ Z
        Z ⊗ Z ⊗ Z ⊗ Z ⊗ Z
```

Therefore, operating **op** on the post-measurement state **out** bring it to the desired subspace.

```
In[•]:= ket0 = op ** out;
        ket0 // SimpleForm
```

$$\text{Out[•]}= \frac{|00000\rangle}{4} - \frac{|00011\rangle}{4} + \frac{|00101\rangle}{4} - \frac{|00110\rangle}{4} + \frac{|01001\rangle}{4} + \frac{|01010\rangle}{4} - \frac{|01100\rangle}{4} - \frac{|01111\rangle}{4} - \frac{|10001\rangle}{4} + \frac{|10010\rangle}{4} + \frac{|10100\rangle}{4} - \frac{|10111\rangle}{4} - \frac{|11000\rangle}{4} - \frac{|11011\rangle}{4} - \frac{|11101\rangle}{4} - \frac{|11110\rangle}{4}$$

One check if this state is indeed the desired logical basis state $\left|\overline{0}\,\overline{0}\cdots\overline{0}\right\rangle$.

```
In[•]:= MeasurementOdds[opZ][ket0]
```

$$\text{Out[•]}= \left\langle \left| 0 \to \left\{ 1, \frac{1}{2}\left(\frac{|\_\rangle}{2} - \frac{1}{2}|1_{S_1}1_{S_2}\rangle - \frac{1}{2}|1_{S_1}1_{S_2}1_{S_3}1_{S_4}\rangle - \frac{1}{2}|1_{S_1}1_{S_2}1_{S_3}1_{S_5}\rangle - \frac{1}{2}|1_{S_1}1_{S_2}1_{S_4}1_{S_5}\rangle + \frac{1}{2}|1_{S_1}1_{S_3}\rangle - \frac{1}{2}|1_{S_1}1_{S_3}1_{S_4}1_{S_5}\rangle + \frac{1}{2}|1_{S_1}1_{S_4}\rangle - \frac{1}{2}|1_{S_1}1_{S_5}\rangle - \frac{1}{2}|1_{S_2}1_{S_3}\rangle - \frac{1}{2}|1_{S_2}1_{S_3}1_{S_4}1_{S_5}\rangle + \frac{1}{2}|1_{S_2}1_{S_4}\rangle + \frac{1}{2}|1_{S_2}1_{S_5}\rangle - \frac{1}{2}|1_{S_3}1_{S_4}\rangle + \frac{1}{2}|1_{S_3}1_{S_5}\rangle - \frac{1}{2}|1_{S_4}1_{S_5}\rangle \right) \right\}, 1 \to \{0, 0\} \right| \right\rangle$$

References

- D. Gottesman, Physical Review A 54, 1862 (1996), “Class of quantum error- correcting codes saturating the quantum Hamming bound”.
- D. Gottesman, Stabilizer Codes and Quantum Error Correction (Ph.D. Thesis, California Institute of Technology, 1997).
- D. Gottesman, Physical Review A 57, 127 (1998), “Theory of fault-tolerant quantum computation”.
- R. Cleve and D. Gottesman, Physical Review A 56 (1), 76 (1997), “Efficient computations of encodings for quantum error correction.”
- *M. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information (Cambridge University Press, New York, 2011)*.
- Mahn-Soo Choi, A Quantum Computation Workbook (Springer, 2022).