Critical noise parameters for fault-tolerant quantum computation

Pavithran Iyer & David Poulin
Funded by ARO QCVV W911NF-14-C-0048

Équipe de Recherche sur la Physique de l’Information Quantique
Département de Physique
Université de Sherbrooke

ThinkQ 2015, IBM T. J. Watson Research, Yorktown Heights NY, December 2015
Outline

1. Introduction
2. Syndrome sampling
3. Logical error vs physical error
4. Machine learning of critical parameters
We want to execute a quantum algorithm with $N$ logical gates.

- $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $Fe_2S_2$.
- Each gate is error-corrected to accuracy $\delta$, so errors build up to
  - $N\delta$ if they add coherently (worst case, systematic bias).
  - $\sqrt{N}\delta$ if they add stochastically.
- $\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.
  - $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).
- If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.

- $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $\text{Fe}_2\text{S}_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to

- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\operatorname{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.

- $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to

- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
Fault-tolerant overhead

- We want to execute a quantum algorithm with $N$ logical gates.
  - $N \sim 10^{12} - 10^{15}$ to simulate a small molecule like $Fe_2S_2$.
- Each gate is error-corrected to accuracy $\delta$, so errors build up to
  - $N\delta$ if they add coherently (worst case, systematic bias).
  - $\sqrt{N}\delta$ if they add stochastically.
- $\delta$ needs to be $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.
  - $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).
- If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead polylog($1/\delta$).

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.

- $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to

- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.
- $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to
- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to be $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.
- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead polylog($\frac{1}{\delta}$).

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.

- $N \approx 10^{12}$-$10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to

- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates. $N \sim 10^{12}-10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to $N\delta$ if they add coherently (worst case, systematic bias).

$\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to me $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

$10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?
We want to execute a quantum algorithm with $N$ logical gates.

- $N \sim 10^{12}$-$10^{15}$ to simulate a small molecule like $Fe_2S_2$.

Each gate is error-corrected to accuracy $\delta$, so errors build up to

- $N\delta$ if they add coherently (worst case, systematic bias).
- $\sqrt{N}\delta$ if they add stochastically.

$\delta$ needs to be $\sim 1/\sqrt{N}$ to $1/N$ to prevent harmful error build up.

- $10^{-6}$ to $10^{-15}$ for quantum chemistry (pretty vague).

If the physical noise rate $\epsilon$ is sub threshold, then fault-tolerant error correction can produce logical gates of accuracy $\delta$ with overhead $\text{polylog}(\frac{1}{\delta})$.

**Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?**
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $>1/\delta$ samples, so inaccessible for relevant $\delta$. 

Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $> 1/\delta$ samples, so inaccessible for relevant $\delta$. 
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $\gg 1/\delta$ samples, so inaccessible for relevant $\delta$. 
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.

- Also depends on details of noise model.
  - Analytical answers usually grossly overestimate the cost.
  - Numerical simulations can in principle provide decent answers, but...
    - Typically limited to unphysical Pauli noise models.
    - Monte Carlo simulations usually require $> 1/\delta$ samples, so inaccessible for relevant $\delta$. 
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $>1/\delta$ samples, so inaccessible for relevant $\delta$. 

D. Poulin (Sherbrooke)  IBM 2015  4 / 31
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $> 1/\delta$ samples, so inaccessible for relevant $\delta$. 
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.

- Also depends on details of noise model.

- Analytical answers usually grossly overestimate the cost.

- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $> 1/\delta$ samples, so inaccessible for relevant $\delta$. 
Given a physical noise rate $\epsilon$, how much error correction do I need to achieve a logical noise rate $\delta$?

- Depends on the error-correction scheme.
  - Optimal to reach target logical failure rate $\neq$ optimal asymptotic scaling.
- Also depends on details of noise model.
- Analytical answers usually grossly overestimate the cost.
- Numerical simulations can in principle provide decent answers, but...
  - Typically limited to unphysical Pauli noise models.
  - Monte Carlo simulations usually require $> 1/\delta$ samples, so inaccessible for relevant $\delta$. 

D. Poulin (Sherbrooke)
Method to efficiently probe low-noise rates for arbitrary uncorrelated noise models for concatenated codes.

Use of this method study the logical failure rate $\delta$ as a function of physical noise rate $\epsilon$.

Need to know more details of the noise model.

Use of machine learning techniques to learn the critical parameters of the noise model (preliminary).
In this talk

- Method to efficiently probe low-noise rates for arbitrary uncorrelated noise models for concatenated codes.
- Use of this method study the logical failure rate $\delta$ as a function of physical noise rate $\epsilon$.
  - Need to know more details of the noise model.
- Use of machine learning techniques to learn the critical parameters of the noise model (preliminary).
In this talk

- Method to efficiently probe low-noise rates for arbitrary uncorrelated noise models for concatenated codes.
- Use of this method study the logical failure rate $\delta$ as a function of physical noise rate $\epsilon$.
  - Need to know more details of the noise model.
- Use of machine learning techniques to learn the critical parameters of the noise model (preliminary).
Method to efficiently probe low-noise rates for arbitrary uncorrelated noise models for concatenated codes.

Use of this method study the logical failure rate $\delta$ as a function of physical noise rate $\varepsilon$.

- Need to know more details of the noise model.

Use of machine learning techniques to learn the critical parameters of the noise model (preliminary).
Outline

1. Introduction
2. Syndrome sampling
3. Logical error vs physical error
4. Machine learning of critical parameters
Outline

1. Introduction
2. Syndrome sampling
3. Logical error vs physical error
4. Machine learning of critical parameters
Quantum error correction

We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.

The measurement outcome is called the error syndrome.

Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:
- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:
- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.

The measurement outcome is called the error syndrome.

Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.

The measurement outcome is called the error syndrome.

Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:

- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.

The measurement outcome is called the error syndrome.

- Computing the most likely recovery $V$ given the syndrome is called decoding.
We never really apply $U^\dagger$, this is a Heisenberg picture of what’s going on:
- Instead of measuring $Z_j$, we measure $UZ_jU^\dagger$.
- The measurement outcome is called the error syndrome.
- Computing the most likely recovery $V$ given the syndrome is called decoding.
Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = CNOT$ or $T$. 
Standard Monte Carlo QEC benchmark

Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
  - The logical failure rate is $\delta = \max_Y \delta_Y$.
    - Typically dominated by $X = \text{CNOT}$ or $T$. 
Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Input:
- A physical noise model for every gate $X$, described by a probability distribution $P_X(E)$: the gate $X$ is to be followed by an error $E$ drawn from $P$.
  - E.g. Depolarizing noise, $P(E) = p^{|E|}(1 - p)^{n-|E|}$.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$.
Estimating $\delta$ requires $O\left(\frac{1}{\delta}\right)$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$. Estimating $\delta$ requires $O\left(\frac{1}{\delta}\right)$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$. Estimating $\delta$ requires $O\left(\frac{1}{\delta}\right)$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$. Estimating $\delta$ requires $O\left(\frac{1}{\delta}\right)$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$. Estimating $\delta$ requires $\mathcal{O}\left(\frac{1}{\delta}\right)$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$.

Estimating $\delta$ requires $\mathcal{O}(\frac{1}{\delta})$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Sprinkle errors in the circuit: follow every gate $X$ in the circuit by an error $E$ drawn from $P_X$.
3. Compute syndrome associated to error pattern.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. Check if the combination circuit+error+correction implement logical gate $Y$.

Frequency of incorrect implementation of $Y$ estimates $\delta_Y$. Estimating $\delta$ requires $O\left(\frac{1}{\delta}\right)$ samples.
Has the circuit failed or not?
Syndrome sampling

Standard Monte Carlo QEC benchmark

Has the circuit failed or not?
Syndrome sampling

Standard Monte Carlo QEC benchmark

Has the circuit failed or not?

D. Poulin (Sherbrooke)
Has the circuit failed or not?
Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = CNOT$ or $T$. 

D. Poulin (Sherbrooke)
Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Syndrome sampling

Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
  - The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 

D. Poulin (Sherbrooke)
Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Monte Carlo syndrome sampling

Input:
- A physical noise model for every gate $X$, described by a CPTP map $\mathcal{E}_X$: the gate $X$ is realized as $\mathcal{E}_X$.
  - E.g. $\mathcal{E}_X$ could be obtained by quantum process tomography.
- A QECC and associated FT circuit and decoding algorithm.

Output:
- A logical failure rate $\delta_Y$ for each logical gate $Y$.
- The logical failure rate is $\delta = \max_Y \delta_Y$.
  - Typically dominated by $X = \text{CNOT}$ or $T$. 
Monte Carlo syndrome sampling

To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}_Y^s$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \|\mathcal{E}_Y^s - Y\|_1 \rangle_s$.

Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
Syndrome sampling

Monte Carlo syndrome sampling

To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born's rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}^s_Y$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}^s_Y - Y \| ? \rangle_s$.

Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}_Y^s$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}_Y^s - Y \| \rangle_s$. Estimating $\delta$ with relative accuracy $\eta$ requires $O\left(\frac{1}{\eta^2}\right)$ samples.
Monte Carlo syndrome sampling

To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}_Y^s$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}_Y^s - Y \|_\diamond \rangle_s$. Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
Monte Carlo syndrome sampling

To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}_Y^s$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}_Y^s - Y \| \rangle_s$.

Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}_s^Y$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}_s^Y - Y \| \rangle_s$.

Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
To estimate $\delta_Y$ repeat:

1. Display the circuit to implement a FT logical gate $Y$.
2. Replace every gate $X$ in the circuit by the CPTP map $\mathcal{E}_X$.
3. Draw a syndrome $s$ at random according to the Born’s rule.
4. Execute decoding algorithm given the syndrome to obtain a correction.
5. The combination noisy circuit+correction results in a noisy logical gate $\mathcal{E}^s_Y$ (which depends on the syndrome).

$\delta_Y$ is the average noise of the resulting logical map, $\langle \| \mathcal{E}^s_Y - Y \| ? \rangle_s$. Estimating $\delta$ with relative accuracy $\eta$ requires $\mathcal{O}(\frac{1}{\eta^2})$ samples.
What transformation $\mathcal{E}^S$ was applied to the encoded data?
What transformation $\mathcal{E}^S$ was applied to the encoded data?
What transformation $\mathcal{E}^s$ was applied to the encoded data?
What transformation $E^S$ was applied to the encoded data?
Syndrome sampling

Concatenated codes

- Step 3 above requires a full numerical simulation of an \( n \)-qubit noisy process.
  - This can be realized by brute force for \( \sim 15 \) qubits.
  - Enough to study concatenated codes

1. For every gate \( Y \), use the Monte Carlo syndrome sampling protocol with gates \( \mathcal{E}_X \) to generate a level-1 gate population \( \mathcal{E}_Y^{s_j,1} \), \( j = 1, 2, \ldots, N \).

2. For every gate \( Y \), use the Monte Carlo syndrome sampling protocol with gates \( \mathcal{E}_X^{s_j,k-1} \) to generate a level-\( k \) gate population \( \mathcal{E}_Y^{s_j,k} \), \( j = 1, 2, \ldots, N \).
Step 3 above requires a full numerical simulation of an $n$-qubit noisy process.

This can be realized by brute force for $\sim 15$ qubits.

Enough to study concatenated codes

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X$ to generate a level-1 gate population $\mathcal{E}_{Y}^{s_j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X^{s_j,k-1}$ to generate a level-$k$ gate population $\mathcal{E}_{Y}^{s_j,k}$, $j = 1, 2, \ldots, N$. 
Step 3 above requires a full numerical simulation of an $n$-qubit noisy process.

This can be realized by brute force for $\sim 15$ qubits.

Enough to study concatenated codes

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X^{s_j,1}$ to generate a level-1 gate population $\mathcal{E}_Y^{s_j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X^{s_j,k-1}$ to generate a level-$k$ gate population $\mathcal{E}_Y^{s_j,k}$, $j = 1, 2, \ldots, N$. 
Step 3 above requires a full numerical simulation of an $n$-qubit noisy process. This can be realized by brute force for $\sim 15$ qubits. Enough to study concatenated codes.

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X$ to generate a level-1 gate population $\mathcal{E}_{Y}^{S_j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X^{S_j,k-1}$ to generate a level-$k$ gate population $\mathcal{E}_{Y}^{S_j,k}$, $j = 1, 2, \ldots, N$. 
Step 3 above requires a full numerical simulation of an $n$-qubit noisy process.

This can be realized by brute force for $\sim 15$ qubits.

Enough to study concatenated codes

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $E_X$ to generate a level-1 gate population $E_{Y,j}^{s_j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $E_X^{s_j,k-1}$ to generate a level-$k$ gate population $E_{Y,j}^{s_j,k}$, $j = 1, 2, \ldots, N$. 

D. Poulin (Sherbrooke)
Step 3 above requires a full numerical simulation of an $n$-qubit noisy process. This can be realized by brute force for $\sim 15$ qubits. Enough to study concatenated codes.

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $E_X$ to generate a level-1 gate population $E_{Y,j}^{s_j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $E_{X,j}^{s_j,k-1}$ to generate a level-$k$ gate population $E_{Y,j}^{s_j,k}$, $j = 1, 2, \ldots, N$.

$\rightarrow$ Replace every gate $X$ in the circuit by the CPTP map $E_X$. 
Concatenated codes

- Step 3 above requires a full numerical simulation of an $n$-qubit noisy process.
- This can be realized by brute force for $\sim 15$ qubits.
- Enough to study concatenated codes

1. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_X$ to generate a level-1 gate population $\mathcal{E}_{Y,j}^{s,j,1}$, $j = 1, 2, \ldots, N$.

2. For every gate $Y$, use the Monte Carlo syndrome sampling protocol with gates $\mathcal{E}_{Y,j}^{s,j,k-1}$ to generate a level-$k$ gate population $\mathcal{E}_{Y,j}^{s,j,k}$, $j = 1, 2, \ldots, N$.

→ Replace every gate $X$ in the circuit by a CPTP map $\mathcal{E}_{Y,j}^{s,j,k-1}$ with $j$ chosen uniformly in $1, 2, \ldots, N$. 
Syndrome sampling

Concatenated codes

Full simulation

QEC Syndrome = \( \beta \)

\( E_{a_1} E_{a_2} E_{a_3} E_{a_4} E_{a_5} E_{a_6} E_{a_7} \)

\( \epsilon_\beta \)

\( \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6, \epsilon_7, ... \)

\( \epsilon_{t_1}, \epsilon_{t_2}, \epsilon_{t_3}, \epsilon_{t_4}, \epsilon_{t_5}, \epsilon_{t_6}, \epsilon_{t_7}, ... \)

\( \epsilon_{t+1_1}, \epsilon_{t+1_2}, \epsilon_{t+1_3}, \epsilon_{t+1_4}, \epsilon_{t+1_5}, \epsilon_{t+1_6}, \epsilon_{t+1_7}, ... \)

Physical noise model

Sample of noise after \( \ell \) QEC levels
Outline

1. Introduction
2. Syndrome sampling
3. Logical error vs physical error
4. Machine learning of critical parameters
Logical error vs physical error

Level 1

9996 samples of Random channel, at Level 1.

$D_B(J)$
$S(J)$
$p_{err}(J)$
$||J-id||_1$
$||E-id||_\tri$
$D_{tr}(J)$
Depolarizing
Logical error vs physical error

Level 2

9996 samples of Random channel, at Level 2.

- $D_B(\mathcal{J})$
- $S(\mathcal{J})$
- $p_{err}(\mathcal{J})$
- $\|\mathcal{J} - \text{id}\|_1$
- $\|\mathcal{E} - \text{id}\|$
- $D_{tr}(\mathcal{J})$
- Depolarizing

Level 0 metrics

- $10^{-7}$
- $10^{-6}$
- $10^{-5}$
- $10^{-4}$
- $10^{-3}$
- $10^{-2}$
- $10^{-1}$
- $10^{0}$

$||J-id||_1$

$||E-id||$

Critical noise parameters

IBM 2015 19 / 31
Logical error vs physical error

Level 2

Physical channels with trace distance = 0.2 from perfect

9996 samples of Random channel, at Level 2.

$D_B(\mathcal{J})$

$S(\mathcal{J})$

$p_{err}(\mathcal{J})$

$\|\mathcal{J} - \text{id}\|_1$

$\|\mathcal{E} - \text{id}\|_1$

$D_{tr}(\mathcal{J})$

Depolarizing

$\|\mathcal{J} - \text{id}\|_1$

Physical channels with trace distance = 0.2 from perfect
Logical error vs physical error

Level 2

Physical channels with trace distance = 0.2 from perfect Depolarizing noise yields logical channel with > 0.1 failure rate

9996 samples of Random channel, at Level 2.

Depolarizing noise yields logical channel with > 0.1 failure rate

Physical channels with trace distance = 0.2 from perfect

Level 0 metrics
Logical error vs physical error

Level 2

Physical channels with trace distance = 0.2 from perfect Depolarizing noise yields logical channel with > 0.1 failure rate

Other channels with same noise rate yield logical failure rate between 0.5 and 10\(^{-6}\)

Physical channels with trace distance = 0.2 from perfect

9996 samples of Random channel, at Level 2.

Depolarizing noise yields logical channel with > 0.1 failure rate

Other channels with same noise rate yield logical failure rate between 0.5 and 10\(^{-6}\)
Logical error vs physical error

Level 3

9996 samples of Random channel, at Level 3.

- $D_B(J)$
- $S(J)$
- $p_{err}(J)$
- $\|J - \text{id}\|_1$
- $\|\mathcal{E} - \text{id}\|_\diamond$
- $D_{tr}(J)$
- Depolarizing

Critical noise parameters

D. Poulin (Sherbrooke)
Level 4

9996 samples of Random channel, at Level 4.

- $D_B(\mathcal{J})$
- $S(\mathcal{J})$
- $p_{err}(\mathcal{J})$
- $\|\mathcal{J} - \text{id}\|_1$
- $\|\mathcal{E} - \text{id}\|_\Diamond$
- $D_{tr}(\mathcal{J})$
- Depolarizing
Predictability

Conclusion

It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

- Need a new type of “metric”?
  - These simulations provide a metric, but it is not very intuitive or informative.
- Need more than a single number (noise rate) to predict logical failure rate.
  - Combination of metrics?
  - Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
**Predictability**

**Conclusion**

It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

- Need a new type of “metric”?
  - These simulations provide a metric, but it is not very intuitive or informative.
  - Need more than a single number (noise rate) to predict logical failure rate.
    - Combination of metrics?
    - Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

Need a new type of “metric”? These simulations provide a metric, but it is not very intuitive or informative.

Need more than a single number (noise rate) to predict logical failure rate.

Combination of metrics? Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
Predictability

Conclusion

It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

- Need a new type of “metric”?
  - These simulations provide a metric, but it is not very intuitive or informative.
- Need more than a single number (noise rate) to predict logical failure rate.
  - Combination of metrics?
  - Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

- Need a new type of “metric”?
  - These simulations provide a metric, but it is not very intuitive or informative.

- Need more than a single number (noise rate) to predict logical failure rate.
  - Combination of metrics?
  - Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
Predictability

Conclusion

It is not possible to even very crudely predict the logical failure rate of a FT scheme given only the noise rate of the physical channel, as measured by any of the standard error metrics.

- Need a new type of “metric”?  
  - These simulations provide a metric, but it is not very intuitive or informative.
- Need more than a single number (noise rate) to predict logical failure rate.
  - Combination of metrics?
  - Which ones?

For upper bounds, see Wallman, Granade, Harper, Flammia, arXiv:1503.07865
Outline

1. Introduction
2. Syndrome sampling
3. Logical error vs physical error
4. Machine learning of critical parameters
We have a large collection of

1. physical noise models $\mathcal{E}_j$ and
2. their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E}), f_2(\mathcal{E}), \ldots, f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of physical noise models $\mathcal{E}_j$ and their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, \ldots, $f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of physical noise models $\mathcal{E}_j$ and their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, …, $f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of

1. physical noise models $\mathcal{E}_j$ and
2. their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E}), f_2(\mathcal{E}), \ldots, f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of

1. physical noise models $\mathcal{E}_j$ and
2. their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, $\ldots$, $f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of
1. physical noise models $\mathcal{E}_j$ and
2. their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, $\ldots$, $f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
We have a large collection of
1 physical noise models $\mathcal{E}_j$ and
2 their logical failure rate $\delta(\mathcal{E}_j, k)$ at level $k$.

A noise model is a list of 16 real coefficients (in the simplest case).

Are there “simple” functions $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, $\ldots$, $f_h(\mathcal{E})$ that correlate with $\delta(\mathcal{E}, k)$?

If yes, these are what experimentalists should be measuring and reporting.

We can use a computer to search for such functions.
Let $f(\mathcal{E})$ be a single quadratic function of $\mathcal{E}$ that can best predict $\delta(\mathcal{E}_j, k)$.

- Searching over such functions is just a quadratic fit to the data.
- The function $f$ could also depend on other features of the channel:
  - The standard metrics themselves, 
    $$f(\mathcal{E}) = Q(\mathcal{E}) + \alpha_1 F(\mathcal{E}) + \alpha_2 D_{tr}(\mathcal{E}) + \alpha_3 \ldots$$
  - The eigenvalues of the channel, 
    $$f(\mathcal{E}) = Q(\mathcal{E}) + A(\mathcal{E}) + \beta_1 \lambda_1(\mathcal{E}) + \beta_2 \lambda_2(\mathcal{E}) + \beta_3 \ldots$$
Let $f(E)$ be a single quadratic function of $E$ that can best predict $\delta(E_j, k)$.

Searching over such functions is just a quadratic fit to the data.

The function $f$ could also depend on other features of the channel:

- The standard metrics themselves,
  
  $f(E) = Q(E) + \alpha_1 F(E) + \alpha_2 D_{tr}(E) + \alpha_3 \ldots$

- The eigenvalues of the channel,
  
  $f(E) = Q(E) + A(E) + \beta_1 \lambda_1(E) + \beta_2 \lambda_2(E) + \beta_3 \ldots$
Let \( f(\mathcal{E}) \) be single a quadratic function of \( \mathcal{E} \) that can best predict \( \delta(\mathcal{E}_j, k) \).

Searching over such functions is just a quadratic fit to the data.

The function \( f \) could also depend on other features of the channel:

- The standard metrics themselves,
  \[
  f(\mathcal{E}) = Q(\mathcal{E}) + \alpha_1 F(\mathcal{E}) + \alpha_2 D_{tr}(\mathcal{E}) + \alpha_3 \ldots
  \]

- The eigenvalues of the channel,
  \[
  f(\mathcal{E}) = Q(\mathcal{E}) + A(\mathcal{E}) + \beta_1 \lambda_1(\mathcal{E}) + \beta_2 \lambda_2(\mathcal{E}) + \beta_3 \ldots
  \]
Let $f(\mathcal{E})$ be a single quadratic function of $\mathcal{E}$ that can best predict $\delta(\mathcal{E}_j, k)$.

Searching over such functions is just a quadratic fit to the data.

The function $f$ could also depend on other features of the channel:

- The standard metrics themselves,
  \[ f(\mathcal{E}) = Q(\mathcal{E}) + \alpha_1 F(\mathcal{E}) + \alpha_2 D_{tr}(\mathcal{E}) + \alpha_3 \ldots \]
- The eigenvalues of the channel,
  \[ f(\mathcal{E}) = Q(\mathcal{E}) + A(\mathcal{E}) + \beta_1 \lambda_1(\mathcal{E}) + \beta_2 \lambda_2(\mathcal{E}) + \beta_3 \ldots \]
Let $f(\mathcal{E})$ be a single quadratic function of $\mathcal{E}$ that can best predict $\delta(\mathcal{E}_j, k)$.

Searching over such functions is just a quadratic fit to the data.

The function $f$ could also depend on other features of the channel:

- The standard metrics themselves,
  $$f(\mathcal{E}) = Q(\mathcal{E}) + \alpha_1 F(\mathcal{E}) + \alpha_2 D_{tr}(\mathcal{E}) + \alpha_3 \ldots$$

- The eigenvalues of the channel,
  $$f(\mathcal{E}) = Q(\mathcal{E}) + A(\mathcal{E}) + \beta_1 \lambda_1(\mathcal{E}) + \beta_2 \lambda_2(\mathcal{E}) + \beta_3 \ldots$$
Comparing quadratic fit with trace distance.

Quadratic fit $D_{tr}(\mathcal{J})$

Input metric

$||\mathcal{J} - \text{id}||_1$
Comparing quadratic fit with diamond norm.
Comparing quadratic fit with error probability.

- Quadratic fit
- $p_{err}(\mathcal{J})$
Comparing quadratic fit with entropy.

\[ S(J) \]

Quadratic fit

Input metric

\[ ||J - \text{id}||_1 \]
Comparing quadratic fit with Bures distance.

Quadratic fit

$D_B(\mathcal{J})$

Input metric

$\|\mathcal{J} - \text{id}\|$
## Quadratic fit – Level 1

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average relative variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bures distance</td>
<td>3.19</td>
</tr>
<tr>
<td>Entropy</td>
<td>1.31</td>
</tr>
<tr>
<td>Error probability</td>
<td>6.27</td>
</tr>
<tr>
<td>Trace norm</td>
<td>3.09</td>
</tr>
<tr>
<td>Diamond norm</td>
<td>2.93</td>
</tr>
<tr>
<td>Quadratic fit</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- Small improvement, but not enough.
- Need more than one parameter.
### Quadratic fit – Level 1

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average relative variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bures distance</td>
<td>3.19</td>
</tr>
<tr>
<td>Entropy</td>
<td>1.31</td>
</tr>
<tr>
<td>Error probability</td>
<td>6.27</td>
</tr>
<tr>
<td>Trace norm</td>
<td>3.09</td>
</tr>
<tr>
<td>Diamond norm</td>
<td>2.93</td>
</tr>
<tr>
<td>Quadratic fit</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- Small improvement, but not enough.
- Need more than one parameter.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Average relative variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bures distance</td>
<td>0.27</td>
</tr>
<tr>
<td>Entropy</td>
<td>0.12</td>
</tr>
<tr>
<td>Infidelity</td>
<td>0.11</td>
</tr>
<tr>
<td>Error probability</td>
<td>0.76</td>
</tr>
<tr>
<td>Frobenious norm</td>
<td>0.26</td>
</tr>
<tr>
<td>Trace norm</td>
<td>0.27</td>
</tr>
<tr>
<td>Diamond norm</td>
<td>0.27</td>
</tr>
<tr>
<td>Features</td>
<td>0.032</td>
</tr>
<tr>
<td>Metric</td>
<td>Average relative variance</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Bures distance</td>
<td>1.33</td>
</tr>
<tr>
<td>Entropy</td>
<td>1.12</td>
</tr>
<tr>
<td>Infidelity</td>
<td>1.85</td>
</tr>
<tr>
<td>Error probability</td>
<td>1.71</td>
</tr>
<tr>
<td>Frobenious norm</td>
<td>1.42</td>
</tr>
<tr>
<td>Trace norm</td>
<td>1.36</td>
</tr>
<tr>
<td>Diamond norm</td>
<td>1.48</td>
</tr>
<tr>
<td>Features</td>
<td>0.14</td>
</tr>
</tbody>
</table>
## Machine learning of critical parameters

### 3 features – Level 3

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average relative variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bures distance</td>
<td>1.60</td>
</tr>
<tr>
<td>Entropy</td>
<td>1.26</td>
</tr>
<tr>
<td>Infidelity</td>
<td>1.51</td>
</tr>
<tr>
<td>Error probability</td>
<td>2.66</td>
</tr>
<tr>
<td>Frobenious norm</td>
<td>1.71</td>
</tr>
<tr>
<td>Trace norm</td>
<td>1.64</td>
</tr>
<tr>
<td>Diamond norm</td>
<td>1.50</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td><strong>0.969</strong></td>
</tr>
</tbody>
</table>
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

- Do these features have an intuitive meaning?
- Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

Do these features have an intuitive meaning?

Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

Preliminary results show up to one order of magnitude improvement in predictive power (variance).

Problem gets harder with more concatenations.

Do these features have an intuitive meaning?

Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

Do these features have an intuitive meaning?

Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

- Do these features have an intuitive meaning?
- Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

- Do these features have an intuitive meaning?
- Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

Do these features have an intuitive meaning?

- Can these features be measured?
We have conceived a numerical tool to efficiently simulate a concatenated FT scheme.

We have found that, for a fixed physical noise rate, the logical failure rate can fluctuate by several orders of magnitude.

- Use numerical tool to check the difference between noise model $\mathcal{E}$ and best Pauli approximation $\tilde{\mathcal{E}}$ (c.f. Cory et al.)

We have started to use machine learning techniques to find features of the channel that better predict the logical failure rate.

- Preliminary results show up to one order of magnitude improvement in predictive power (variance).
- Problem gets harder with more concatenations.

Do these features have an intuitive meaning?

Can these features be measured?