Statistical mechanical models and tensor network decoding for quantum codes

Christopher T. Chubb
ETH Zürich

Stat mech mapping (with Steve Flammia, AWS)
arXiv:2101.04125, to appear in AIHP:D

2D tensor network decoding
arXiv:2101.04125

Tensor contraction Julia package
Coming soon!

Contents

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode?
- What is the threshold?

The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, doi:10/cs2mtf, arXiv:quant-ph/0110143

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode optimally?
- What is the threshold?

The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, doi:10/cs2mtf, arXiv:quant-ph/0110143

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode optimally?
- What is the fault-tolerant threshold?

The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode optimally?
- What is the fault-tolerant threshold?

The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode optimally?
- What is the fault-tolerant threshold?

The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`

Quantum codes

More generally, two important questions arise for any quantum code:

- How do I decode optimally?
- What is the fault-tolerant threshold?

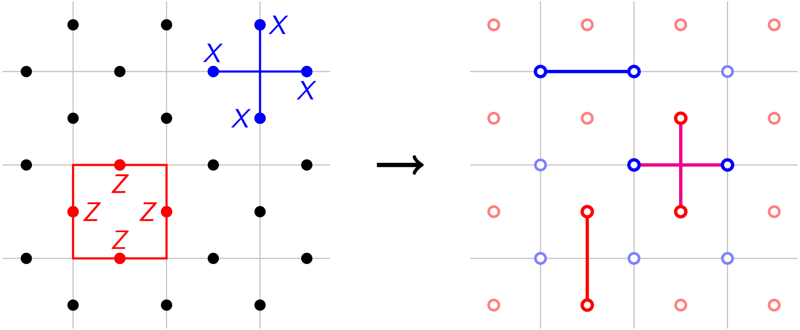
The statistical mechanical mapping[1] allows us to systematically address both questions for codes subject to stochastic Pauli noise.

It achieves this by mapping the answers onto the phase boundary of a disordered classical stat mech model.

By combining this with a novel tensor network contraction scheme we get a general and approximately optimal decoder for any 2D code.

[1] Dennis, Kitaev, Landahl, Preskill, JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`

Statistical mechanical mapping



Toric code

Eight-vertex model

Statistical mechanical mapping

Stabiliser code & Pauli noise	$\longrightarrow$	Disordered statistical mechanical model
Threshold	$\longleftrightarrow$	Phase transition
Decoding	$\longleftrightarrow$	Calculating partition functions

Allows us to reappropriate techniques for studying stat mech systems to study quantum codes, e.g.

Threshold approximation	$\longleftarrow$	Monte Carlo simulation
Optimal decoding	$\longleftarrow$	Partition function calculation

Statistical mechanical mapping

Stabiliser code & Pauli noise	$\longrightarrow$	Disordered statistical mechanical model
Threshold	$\longleftrightarrow$	Phase transition
Decoding	$\longleftrightarrow$	Calculating partition functions

Allows us to reappropriate techniques for studying stat mech systems to study quantum codes, e.g.

Threshold approximation	$\longleftarrow$	Monte Carlo simulation
Optimal decoding	$\longleftarrow$	Partition function calculation

Statistical mechanical mapping

Stabiliser code & Pauli noise	$\longrightarrow$	Disordered statistical mechanical model
Threshold	$\longleftrightarrow$	Phase transition
Decoding	$\longleftrightarrow$	Calculating partition functions

Allows us to reappropriate techniques for studying stat mech systems to study quantum codes, e.g.

Threshold approximation	$\longleftarrow$	Monte Carlo simulation
Optimal decoding	$\longleftarrow$	Partition function calculation

Our results

Stat mech mapping:

- Generalise the mapping to correlated noise for arbitrary subsystem codes
- Numerically study the toric code with mildly correlated errors
- Show how circuit noise can be studied, allowing for fault-tolerant thresholds to be approximated
- Generalise the tensor network decoder of Bravyi, Suchara and Vargo

Tensor network decoding:

- Develop a linearithmic contraction algorithm for 2D tensor networks
- Apply this to several surface/colour codes, including irregular codes
- Provide evidence for a conjecture about surface code thresholds

Our results

Stat mech mapping:

- Generalise the mapping to correlated noise for arbitrary subsystem codes
- Numerically study the toric code with mildly correlated errors
- Show how circuit noise can be studied, allowing for fault-tolerant thresholds to be approximated
- Generalise the tensor network decoder of Bravyi, Suchara and Vargo

Tensor network decoding:

- Develop a linearithmic contraction algorithm for 2D tensor networks
- Apply this to several surface/colour codes, including irregular codes
- Provide evidence for a conjecture about surface code thresholds

Independent case: Hamiltonian

Let $[\![A, B]\!]$ be the scalar commutator of two Paulis, such that $AB =: [\![A, B]\!]\, BA$.

For a stabiliser code generated by $\{S_k\}_k$, and an error Pauli E, the (disordered) Hamiltonian H_E is defined

$$H_E(\vec{s}) := -\sum_i \sum_{\sigma \in \mathcal{P}_i} \overbrace{J_i(\sigma)}^{\text{Coupling}} \overbrace{[\![\sigma, E]\!]}^{\text{Disorder}} \overbrace{\prod_{k:[\![\sigma, S_k]\!]=-1} s_k}^{\text{DoF}}$$

for $s_k = \pm 1$, and coupling strengths $J_i(\sigma) \in \mathbb{R}$.

Take-aways:

- Ising-type, with interactions corresponding to single-site Paulis σ
- Disorder E flips some interactions (Ferro $\leftrightarrow$ Anti-ferro)
- Local code $\implies$ local stat mech model

Independent case: Hamiltonian

Let $[\![A,B]\!]$ be the scalar commutator of two Paulis, such that $AB =: [\![A,B]\!]\, BA$.

For a stabiliser code generated by $\{S_k\}_k$, and an error Pauli E, the (disordered) Hamiltonian H_E is defined

$$H_E(\vec{s}) := -\sum_i \sum_{\sigma\in\mathcal{P}_i} \overbrace{J_i(\sigma)}^{\text{Coupling}} \overbrace{[\![\sigma,E]\!]}^{\text{Disorder}} \overbrace{\prod_{k:[\![\sigma,S_k]\!]=-1} s_k}^{\text{DoF}}$$

for $s_k = \pm 1$, and coupling strengths $J_i(\sigma) \in \mathbb{R}$.

Take-aways:

- Ising-type, with interactions corresponding to single-site Paulis σ
- Disorder E flips some interactions (Ferro $\leftrightarrow$ Anti-ferro)
- Local code $\implies$ local stat mech model

Independent case: Hamiltonian

Let $[\![A,B]\!]$ be the scalar commutator of two Paulis, such that $AB =: [\![A,B]\!]\, BA$.

For a stabiliser code generated by $\{S_k\}_k$, and an error Pauli E, the (disordered) Hamiltonian H_E is defined

$$H_E(\vec{s}) := -\sum_i \sum_{\sigma\in\mathcal{P}_i} \overbrace{J_i(\sigma)}^{\text{Coupling}} \overbrace{[\![\sigma,E]\!]}^{\text{Disorder}} \overbrace{\prod_{k:[\![\sigma,S_k]\!]=-1} s_k}^{\text{DoF}}$$

for $s_k = \pm 1$, and coupling strengths $J_i(\sigma) \in \mathbb{R}$.

Take-aways:

- Ising-type, with interactions corresponding to single-site Paulis σ
- Disorder E flips some interactions (Ferro $\leftrightarrow$ Anti-ferro)
- Local code $\implies$ local stat mech model

Independent case: Gauge symmetry

$$H_E(\vec{s}) = -\sum_i \sum_{\sigma\in\mathcal{P}_i} J_i(\sigma)[\![\sigma, E]\!] \prod_{k:[\![\sigma,S_k]\!]=-1} s_k$$

Using $[\![A, B]\!]\,[\![A, C]\!] = [\![A, BC]\!]$, we see this system has a gauge symmetry

$$s_k \to -s_k \qquad \text{and} \qquad E \to ES_k.$$

This gauge symmetry will capture the logical equivalence of errors, $Z_E = Z_{ES_k}$.

Independent case: Gauge symmetry

$$H_E(\vec{s}) = -\sum_i \sum_{\sigma\in\mathcal{P}_i} J_i(\sigma)[\![\sigma, E]\!] \prod_{k:[\![\sigma,S_k]\!]=-1} s_k$$

Using $[\![A, B]\!]\,[\![A, C]\!] = [\![A, BC]\!]$, we see this system has a gauge symmetry

$$s_k \to -s_k \qquad \text{and} \qquad E \to ES_k.$$

This gauge symmetry will capture the logical equivalence of errors, $Z_E = Z_{ES_k}$.

Independent case: Nishimori conditon

Suppose we have an independent error model

$$\Pr(E) = \prod_i p_i(E_i),$$

we now want $Z_E = \Pr(\overline{E})$, $\overline{E}$ is the error class

$$\overline{E} := \{ES | S \in \mathcal{S}\}.$$

Using the gauge symmetry we have that the partition function can be written as a sum stabiliser-equivalent errors

$$Z_E = \sum_{\vec{s}} e^{-\beta H_E(\vec{s})} = \sum_S e^{-\beta H_{ES}(\vec{1})} = \sum_{F \in \overline{E}} e^{-\beta H_F(\vec{1})}.$$

If we select the coupling strength such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$, then $Z_E = \Pr(\overline{E})$ will follow.

Independent case: Nishimori conditon

Suppose we have an independent error model

$$\Pr(E) = \prod_i p_i(E_i),$$

we now want $Z_E = \Pr(\overline{E})$, $\overline{E}$ is the error class

$$\overline{E} := \{ES | S \in \mathcal{S}\}.$$

Using the gauge symmetry we have that the partition function can be written as a sum stabiliser-equivalent errors

$$Z_E = \sum_{\vec{s}} e^{-\beta H_E(\vec{s})} = \sum_S e^{-\beta H_{ES}(\vec{1})} = \sum_{F \in \overline{E}} e^{-\beta H_F(\vec{1})}.$$

If we select the coupling strength such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$, then $Z_E = \Pr(\overline{E})$ will follow.

Independent case: Nishimori conditon

Suppose we have an independent error model

$$\Pr(E) = \prod_i p_i(E_i),$$

we now want $Z_E = \Pr(\overline{E})$, $\overline{E}$ is the error class

$$\overline{E} := \{ES | S \in \mathcal{S}\} .$$

Using the gauge symmetry we have that the partition function can be written as a sum stabiliser-equivalent errors

$$Z_E = \sum_{\vec{s}} e^{-\beta H_E(\vec{s})} = \sum_S e^{-\beta H_{ES}(\vec{1})} = \sum_{F \in \overline{E}} e^{-\beta H_F(\vec{1})}.$$

If we select the coupling strength such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$, then $Z_E = \Pr(\overline{E})$ will follow.

Independent case: Nishimori condition

We now want to pick our couplings such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$. Expanding this out, we get

$$\sum_i \log p_i(E) = -\sum_i \sum_\sigma \beta J_i(\sigma) [\![\sigma, E]\!].$$

Using the Fourier-like orthogonality relation $\frac{1}{4}\sum_\sigma [\![\sigma, \tau]\!] = \delta_{\tau, I}$, this becomes

Nishimori condition: $$\beta J_i(\sigma) = \frac{1}{4} \sum_{\tau \in \mathcal{P}} \log p_i(\tau) [\![\sigma, \tau]\!],$$

which implies $e^{-\beta H_E(\vec{1})} = \Pr(E)$, and therefore $Z_E = \Pr(\overline{E})$.

This intrinsically links the error correcting behavior of the code to the thermodynamic behavior of the model (along the Nishimori line).

Independent case: Nishimori condition

We now want to pick our couplings such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$. Expanding this out, we get

$$\sum_i \log p_i(E) = -\sum_i \sum_\sigma \beta J_i(\sigma) \llbracket \sigma, E \rrbracket.$$

Using the Fourier-like orthogonality relation $\frac{1}{4}\sum_\sigma \llbracket \sigma, \tau \rrbracket = \delta_{\tau,I}$, this becomes

Nishimori condition: $$\beta J_i(\sigma) = \frac{1}{4} \sum_{\tau \in \mathcal{P}} \log p_i(\tau) \llbracket \sigma, \tau \rrbracket ,$$

which implies $e^{-\beta H_E(\vec{1})} = \Pr(E)$, and therefore $Z_E = \Pr(\overline{E})$.

This intrinsically links the error correcting behavior of the code to the thermodynamic behavior of the model (along the Nishimori line).

Independent case: Nishimori condition

We now want to pick our couplings such that $e^{-\beta H_E(\vec{1})} = \Pr(E)$. Expanding this out, we get

$$\sum_i \log p_i(E) = -\sum_i \sum_\sigma \beta J_i(\sigma) [\![\sigma, E]\!].$$

Using the Fourier-like orthogonality relation $\frac{1}{4}\sum_\sigma [\![\sigma, \tau]\!] = \delta_{\tau, I}$, this becomes

Nishimori condition: $$\beta J_i(\sigma) = \frac{1}{4}\sum_{\tau \in \mathcal{P}} \log p_i(\tau) [\![\sigma, \tau]\!],$$

which implies $e^{-\beta H_E(\vec{1})} = \Pr(E)$, and therefore $Z_E = \Pr(\overline{E})$.

This intrinsically links the error correcting behavior of the code to the thermodynamic behavior of the model (along the Nishimori line).

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

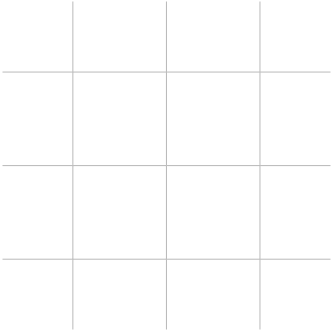
Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

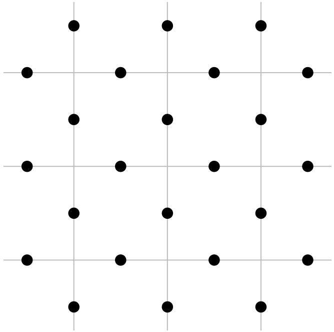
$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$



Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips



Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

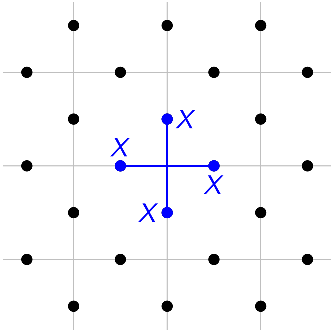
Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

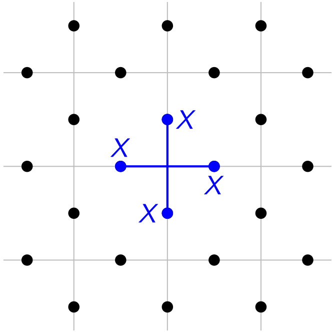


Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$$



Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

Toric code and the random-bond Ising model

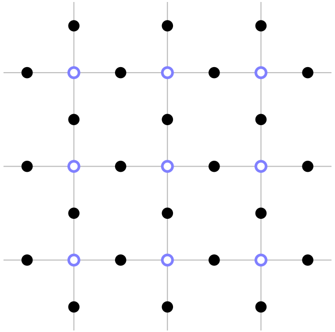
Step 0: Code and noise model

Toric code with iid bit-flips

$$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v



Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

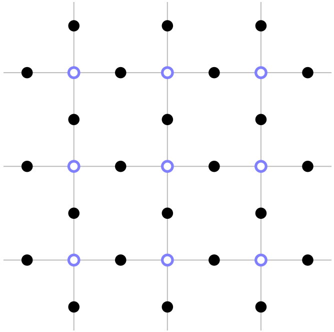
Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$



Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

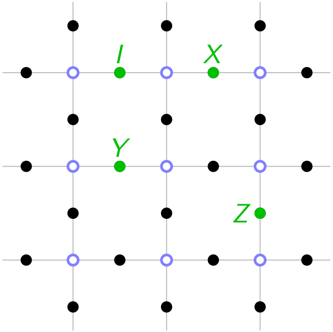
Step 2: Interactions

$$H_I = -\sum_{v\sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v\sim v'} Je_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$



$$[\![\sigma_e, S_v]\!] = \begin{cases} -1 & \text{if } \sigma = X, Y,\ v \in e, \\ +1 & \text{else.} \end{cases}$$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

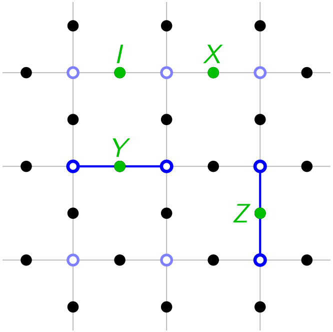
$$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$



Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

$$[\![\sigma_e, S_v]\!] = \begin{cases} -1 & \text{if } \sigma = X, Y,\ v \in e, \\ +1 & \text{else.} \end{cases}$$

Toric code and the random-bond Ising model

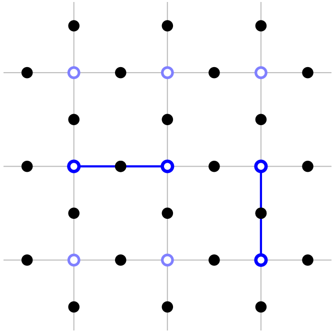
Step 0: Code and noise model

Toric code with iid bit-flips

$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v



Step 2: Interactions

$$H_I = -\sum_{v\sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v\sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

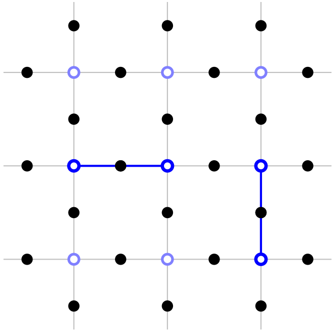
Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$



$[\![\sigma, S_k]\!] \rightarrow [\![\sigma, ES_k]\!]$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$\Pr(X_e) = p, \qquad \Pr(I_e) = 1 - p.$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

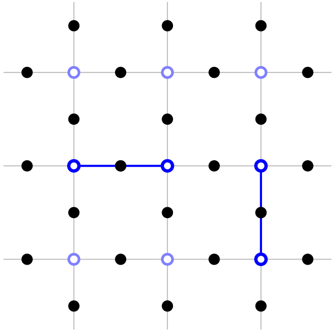
Step 2: Interactions

$$H_I = -\sum_{v \sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v \sim v'} J e_{vv'}\, s_v s_{v'}$$

$$\text{where } e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$$



$[\![\sigma, S_k]\!] \quad \to \quad [\![\sigma, ES_k]\!]$

Toric code and the random-bond Ising model

Step 0: Code and noise model

Toric code with iid bit-flips

$\Pr(+J) = p, \qquad \Pr(-J) = 1 - p.$

Step 1: Degrees of freedom

$s_v = \pm 1$ on each vertex v

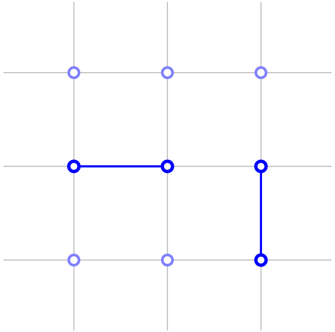
Step 2: Interactions

$$H_I = -\sum_{v\sim v'} J\, s_v s_{v'}$$

Step 3: Disorder

$$H_E = -\sum_{v\sim v'} J e_{vv'}\, s_v s_{v'}$$

where $e_{vv'} = \begin{cases} +1 & E_{vv'} = I, \\ -1 & E_{vv'} = X. \end{cases}$

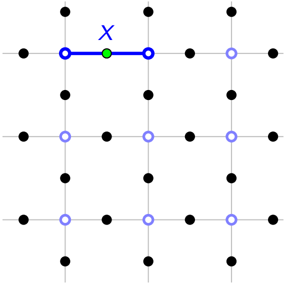


$\pm J$ Random-bond Ising Model

Other independent examples

Toric code

Bit-flip → Random-bond Ising[1]
Indep. $X\&Z$ → 2×Random-bond Ising
Depolarising → Random 8-vertex model[2]



Colour code

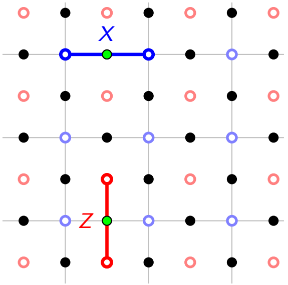
Bit-flip → Random 3-spin Ising
Indep. $X\&Z$ → 2×Random 3-spin Ising
Depolarising → Random interacting 8-vertex[2]

[1] Dennis et.al., JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`
[2] Bombin et.al., PRX 2012, `doi:10/crz5`, `arXiv:1202.1852`

Other independent examples

Toric code

Bit-flip → Random-bond Ising[1]
Indep. $X\&Z$ → 2×Random-bond Ising
Depolarising → Random 8-vertex model[2]



Colour code

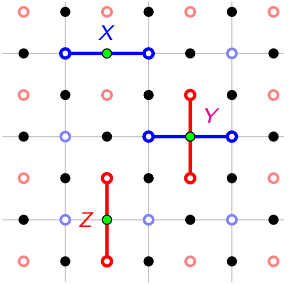
Bit-flip → Random 3-spin Ising
Indep. $X\&Z$ → 2×Random 3-spin Ising
Depolarising → Random interacting 8-vertex[2]

[1] Dennis et.al., JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`
[2] Bombin et.al., PRX 2012, `doi:10/crz5`, `arXiv:1202.1852`

Other independent examples

Toric code

Bit-flip → Random-bond Ising[1]
Indep. $X\&Z$ → 2×Random-bond Ising
Depolarising → Random 8-vertex model[2]



Colour code

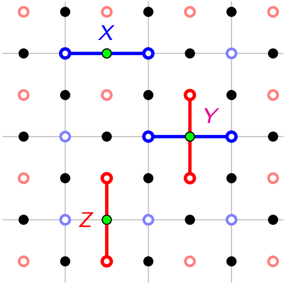
Bit-flip → Random 3-spin Ising
Indep. $X\&Z$→ 2×Random 3-spin Ising
Depolarising → Random interacting 8-vertex[2]

[1] Dennis et.al., JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`
[2] Bombin et.al., PRX 2012, `doi:10/crz5`, `arXiv:1202.1852`

Other independent examples

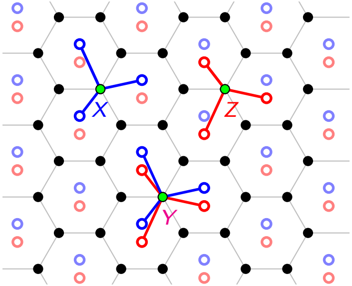
Toric code

Bit-flip → Random-bond Ising[1]
Indep. $X\&Z$ → 2×Random-bond Ising
Depolarising → Random 8-vertex model[2]



Colour code

Bit-flip → Random 3-spin Ising
Indep. $X\&Z$→ 2×Random 3-spin Ising
Depolarising → Random interacting 8-vertex[2]



[1] Dennis et.al., JMP 2002, `doi:10/cs2mtf`, `arXiv:quant-ph/0110143`
[2] Bombin et.al., PRX 2012, `doi:10/crz5`, `arXiv:1202.1852`

Error correction threshold as a quenched phase transition

Consider the free energy cost of a logical error L,

$$\Delta_E(L) = -\frac{1}{\beta}\log Z_{EL} + \frac{1}{\beta}\log Z_E.$$

Along the Nishimori line

$$\Delta_E(L) = \frac{1}{\beta}\log\frac{\Pr(\overline{E})}{\Pr(\overline{EL})},$$

which implies

$$\begin{aligned}\text{Below threshold}: \quad & \Delta_E(L)\to\infty \quad \text{(in mean)}\\ \text{Above threshold}: \quad & \Delta_E(L)\to 0 \quad \text{(in prob.)}\end{aligned}$$

Error correction threshold as a quenched phase transition

Consider the free energy cost of a logical error L,

$$\Delta_E(L) = -\frac{1}{\beta}\log Z_{EL} + \frac{1}{\beta}\log Z_E.$$

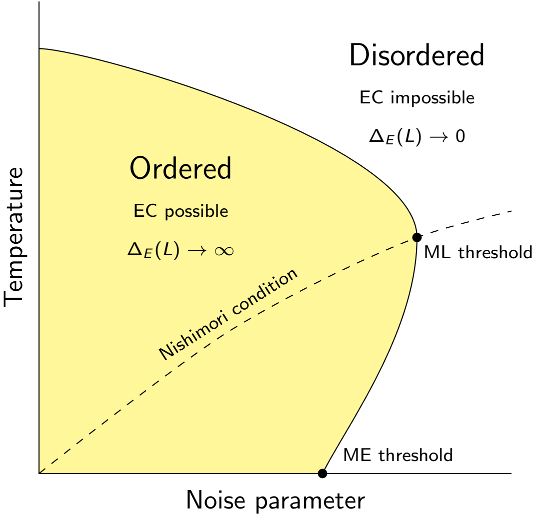
Along the Nishimori line

$$\Delta_E(L) = \frac{1}{\beta}\log\frac{\Pr(\overline{E})}{\Pr(\overline{EL})},$$

which implies

$$\begin{aligned} \text{Below threshold}: &\quad \Delta_E(L) \to \infty \text{ (in mean)} \\ \text{Above threshold}: &\quad \Delta_E(L) \to 0 \quad \text{(in prob.)} \end{aligned}$$

Phase diagram sketch



Correlated case

The key point independence gave us was the ability to factor our noise model

$$\Pr(E) = \prod_i p_i(E_i).$$

We can generalise this to correlated models:

Factored distribution

An error model factors over regions $\{R_j\}_j$ if there exist $\phi_j : \mathcal{P}_{R_j} \to \mathbb{R}$ such that

$$\Pr(E) = \prod_j \phi_j\left(E_{R_j}\right)$$

This model includes many probabilistic graphical models, such as Bayesian Networks and Markov/Gibbs Random Fields.

Correlated case

The key point independence gave us was the ability to factor our noise model

$$\Pr(E) = \prod_i p_i(E_i).$$

We can generalise this to correlated models:

Factored distribution

An error model factors over regions $\{R_j\}_j$ if there exist $\phi_j : \mathcal{P}_{R_j} \to \mathbb{R}$ such that

$$\Pr(E) = \prod_j \phi_j\left(E_{R_j}\right)$$

This model includes many probabilistic graphical models, such as Bayesian Networks and Markov/Gibbs Random Fields.

Correlated case

By construction, we can extend to the correlated case by changing $\sigma \in \mathcal{P}_i$ to $\sigma \in \mathcal{P}_{R_j}$:

$$H_E(\vec{s}) := -\sum_j \sum_{\sigma \in \mathcal{P}_{R_j}} J_j(\sigma) \llbracket \sigma, E \rrbracket \prod_{k : \llbracket \sigma, S_k \rrbracket = -1} s_k$$

Nishimori condition: $$\beta J_j(\sigma) = \frac{1}{|\mathcal{P}_{R_j}|} \sum_{\tau \in \mathcal{P}_{R_j}} \log \phi_j(\tau) \llbracket \sigma, \tau \rrbracket ,$$

As before we get that $Z_E = \Pr(\overline{E})$, and so the threshold manifests as a phase transition.

Correlated case

By construction, we can extend to the correlated case by changing $\sigma \in \mathcal{P}_i$ to $\sigma \in \mathcal{P}_{R_j}$:

$$H_E(\vec{s}) := -\sum_j \sum_{\sigma \in \mathcal{P}_{R_j}} J_j(\sigma) \llbracket \sigma, E \rrbracket \prod_{k : \llbracket \sigma, S_k \rrbracket = -1} s_k$$

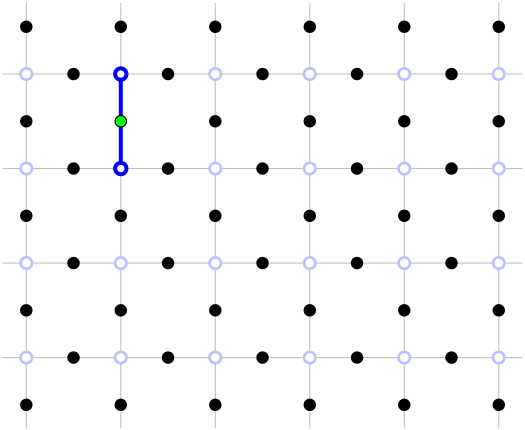
Nishimori condition: $$\beta J_j(\sigma) = \frac{1}{|\mathcal{P}_{R_j}|} \sum_{\tau \in \mathcal{P}_{R_j}} \log \phi_j(\tau) \llbracket \sigma, \tau \rrbracket \,,$$

As before we get that $Z_E = \Pr(\overline{E})$, and so the threshold manifests as a phase transition.

Correlated example

Toric code with correlated bit-flips

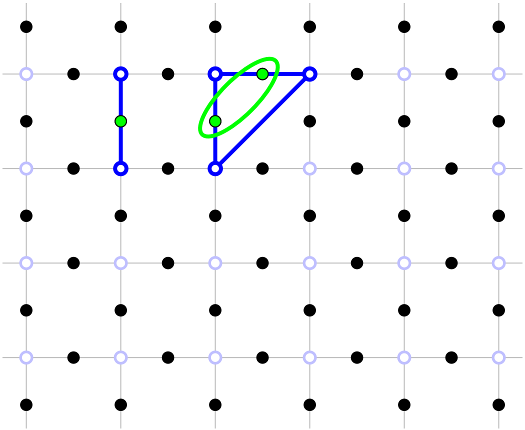
Correlations induce longer-range interactions



Correlated example

Toric code with correlated bit-flips

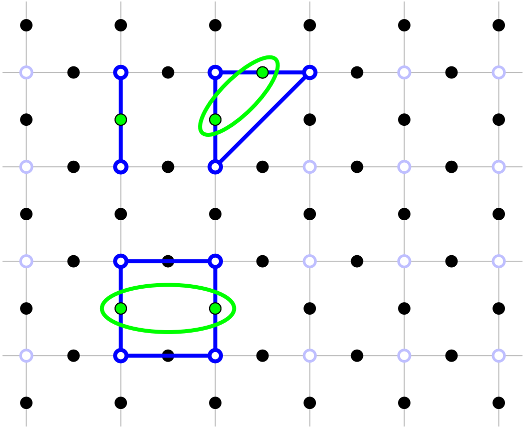
Correlations induce longer-range interactions



Correlated example

Toric code with correlated bit-flips

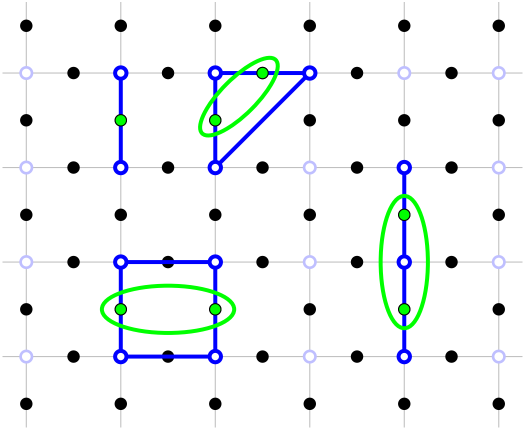
Correlations induce longer-range interactions



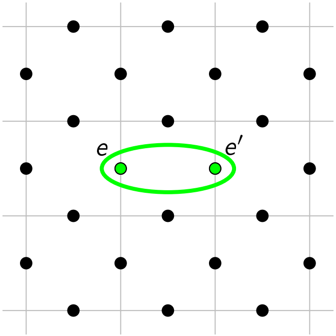
Correlated example

Toric code with correlated bit-flips

Correlations induce longer-range interactions



‘Across plaquette’ correlated bit-flips



Model specified by

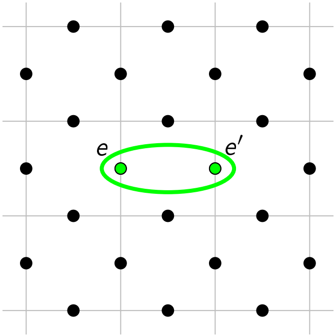
$$\Pr(I_e|I_{e'}) \qquad \Pr(I_e|X_{e'})$$
$$\Pr(X_e|I_{e'}) \qquad \Pr(X_e|X_{e'})$$

for all neighbouring edges e and e'.

Convenient parameterisation:

$$p := \Pr(X_e), \quad \eta := \frac{\Pr(X_e|X_{e'})}{\Pr(X_e|I_{e'})}.$$

‘Across plaquette’ correlated bit-flips



Model specified by

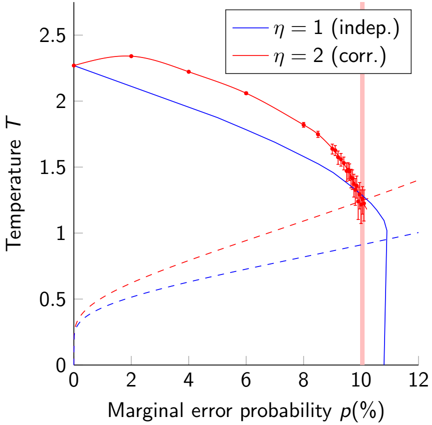
$$\begin{array}{ll} \Pr(I_e|I_{e'}) & \Pr(I_e|X_{e'}) \\ \Pr(X_e|I_{e'}) & \Pr(X_e|X_{e'}) \end{array}$$

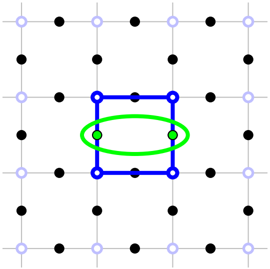
for all neighbouring edges e and e'.

Convenient parameterisation:

$$p := \Pr(X_e), \quad \eta := \frac{\Pr(X_e|X_{e'})}{\Pr(X_e|I_{e'})}.$$

Monte Carlo simulations





Thresholds

Indep.: $p_{\text{th}} = 10.917(3)\%$[1,2]
Corr.: $p_{\text{th}} = 10.04(6)\%$

[1] Dennis et.al., JMP 2002, doi:10/cs2mtf, arXiv:quant-ph/0110143
[2] Toldin et.al., JSP 2009, doi:10/c3r2kc, arXiv:0811.2101

Decoding

Can the stat mech model give us a decoder?

If an error E occurs, a decoder needs to select one of the degenerate logical error classes

$$\overline{E} \qquad \overline{EL_1} \qquad \overline{EL_2} \qquad \overline{EL_3} \qquad \ldots$$

The optimal (maximum likelihood) decoder selects the most likely class

$$D_{\mathrm{ML}} = \overline{EL_l} \qquad \text{where} \quad l = \arg\max_l \Pr\left(\overline{EL_l}\right).$$

Decoding

Can the stat mech model give us a decoder?

If an error E occurs, a decoder needs to select one of the degenerate logical error classes

$$\overline{E} \qquad \overline{EL_1} \qquad \overline{EL_2} \qquad \overline{EL_3} \qquad \ldots$$

The optimal (maximum likelihood) decoder selects the most likely class

$$D_{\mathrm{ML}} = \overline{EL_l} \qquad \text{where} \quad l = \arg\max_l \Pr\left(\overline{EL_l}\right).$$

Decoding

Can the stat mech model give us a decoder?

If an error E occurs, a decoder needs to select one of the degenerate logical error classes

$$\overline{E} \qquad \overline{EL_1} \qquad \overline{EL_2} \qquad \overline{EL_3} \qquad \ldots$$

The optimal (maximum likelihood) decoder selects the most likely class

$$D_{\mathrm{ML}} = \overline{EL_l} \qquad \text{where} \quad l = \arg\max_l \Pr\left(\overline{EL_l}\right).$$

Decoding from partition functions

Along the Nishimori line, the maximum likelihood condition corresponds to maximising the partition function

$$l = \arg\max_{l} Z_{EL_l}.$$

Approximating Z_{EL_l} therefore allows us to approximate the ML decoder.

- Step 1: Measure the syndrome s
- Step 2: Construct an arbitrary error C_s which has syndrome s
- Step 3: Approximate $Z_{C_s L_l} = \Pr(\overline{C_s L_l})$ for each logical l
- Step 4: Find the l such that $Z_{C_s L_l}$ is maximised
- Step 5: Apply $(C_s L_l)^{-1}$

Decoding from partition functions

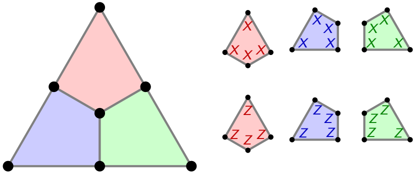
Along the Nishimori line, the maximum likelihood condition corresponds to maximising the partition function

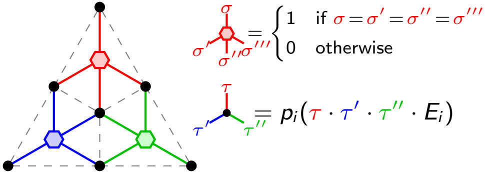
$$l = \arg\max_{l} Z_{EL_l}.$$

Approximating Z_{EL_l} therefore allows us to approximate the ML decoder.

- Step 1: Measure the syndrome s
- Step 2: Construct an arbitrary error C_s which has syndrome s
- Step 3: Approximate $Z_{C_s L_l} = \Pr(\overline{C_s L_l})$ for each logical l
- Step 4: Find the l such that $Z_{C_s L_l}$ is maximised
- Step 5: Apply $(C_s L_l)^{-1}$

ML decoding as a tensor network



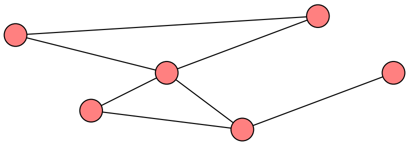


- Partition functions, and thus error class probabilities, can be expressed as tensor networks[2,3].
- There is a tensor for each qubit, and for each stabiliser
- By approximating these class probabilities we can find the ML error

[2]Verstraete et. al., PRL 2006, doi:10/dfgcz8, arXiv:quant-ph/0601075
[3]Bridgeman and Chubb, JPA 2017, doi:10/cv7m, arXiv:1603.03039

Sweepline contraction algorithm

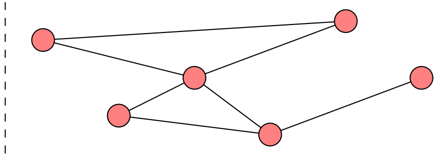
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

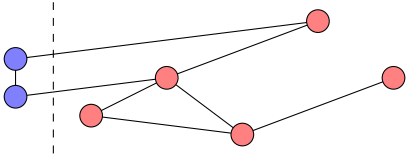
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

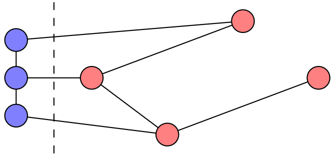
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

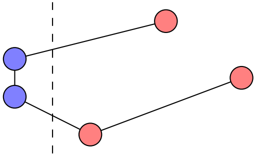
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

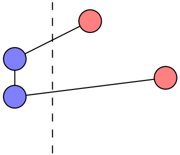
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

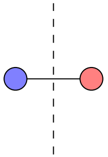
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

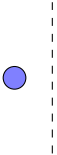
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

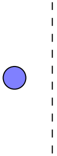
- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Sweepline contraction algorithm

- We sweep along the network, using lossy compression of Matrix Product States
- To do this for general graphs we need to use a sweepline approach



I'm going to release a general purpose Julia implementation of this soon.

Surface codes on different lattices

How does the surface code perform on different lattices/graphs?

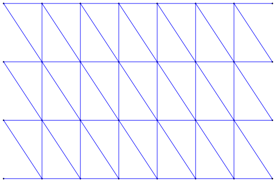
Raise or lower the connectivity

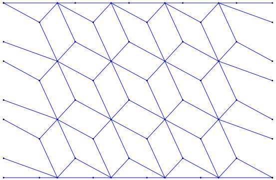
Irregular graphs

Surface codes on different lattices

How does the surface code perform on different lattices/graphs?

Raise or lower the connectivity



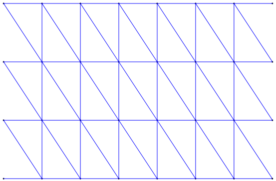


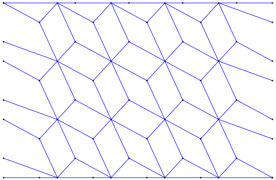
Irregular graphs

Surface codes on different lattices

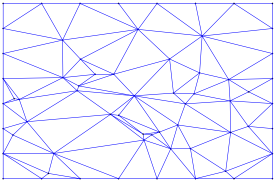
How does the surface code perform on different lattices/graphs?

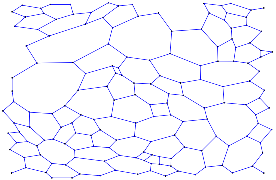
Raise or lower the connectivity





Irregular graphs



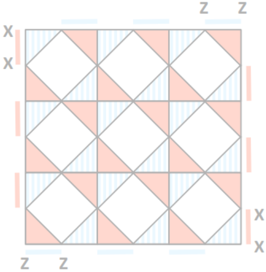


Non-SC topological models

We also want to capture 2D models that aren't (ostensibly) surface codes.

Colour code

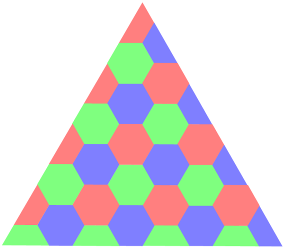
Subsystem surface code



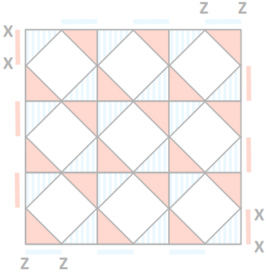
Non-SC topological models

We also want to capture 2D models that aren't (ostensibly) surface codes.

Colour code



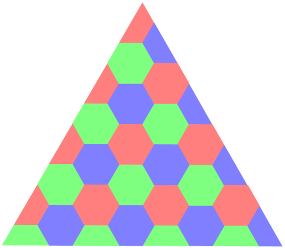
Subsystem surface code



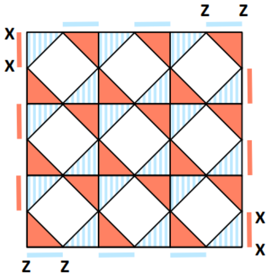
Non-SC topological models

We also want to capture 2D models that aren't (ostensibly) surface codes.

Colour code



Subsystem surface code



Thresholds

	Bit-flip		Phase-flip		Depolarising	
	Observed	Upper bound	Observed	Upper bound	Observed	Upper bound
Surface code (reg.)						
Square	10.917(5)%	10.9187%	10.917(5)%	10.9187%	18.81(3)%	18.9(3)%
Tri./Hex.	16.341(7)%	16.4015%	6.748(5)%	6.7407%	13.81(7)%	?
Kag./Rho.	9.875(5)%	?	11.910(6)%	?	18.09(4)%	?
T.H./Asa.	4.297(7)%	?	20.701(13)%	?	9.07(8)%	?
Surace code (irr.)						
Rand. Tri.	17.128(15)%	?	6.237(9)%	?	12.85(3)%	?
Rand. Quad.	12.195(12)%	?	9.715(11)%	?	18.05(3)%	?
Subsystem SC						
Square	6.705(13)%	6.7407%	6.705(13)%	6.7407%	11.23(3)%	?
Colour Code						
Hexagonal	10.910(5)%	10.9(2)%	10.910(5)%	10.9(2)%	18.68(2)%	18.9(3)%

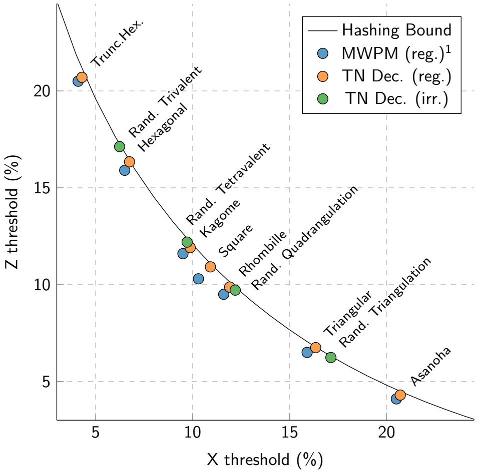
Surface code saturating the Hashing Bound

We find a trade-off between the X and Z thresholds.

Thresholds saturate the hashing bound

$$h(\tau_x) + h(\tau_z) \leq 1$$

Pair matching studied earlier by Fujii et.al.[4]



[4] Fujii et.al., `doi.org/d5sb`, `arXiv:1202.2743`

Surface code saturating the Hashing Bound

We find a trade-off between the X and Z thresholds.

Thresholds saturate the hashing bound

$$h(\tau_x) + h(\tau_z) \leq 1$$

Pair matching studied earlier by Fujii et.al.[4]

	Entropy	
	MWPM[1]	**TN**
Regular		
Square	0.957	0.9948(3)
Tri./Hex.	0.979	0.9989(3)
Kag./Rho.	0.971	0.9918(3)
T.H./Asa.	0.979	0.9915(6)
Irregular		
Rand. Tri.	?	0.9974(7)
Rand. Quad.	?	0.9948(7)

[4] Fujii et.al., doi.org/d5sb, arXiv:1202.2743

Future work

- Stat mech mapping beyond Pauli codes
- Using the stat mech mapping for more than the threshold
- TN decoding of correlated noise
- TN decoding in 3D (including noisy 2D) and beyond, LDPCs, etc.

Thank you!

Stat mech mapping: arXiv:1809.10704, to appear in AIHPD
Tensor network decoding: arXiv:2101.04125
Tensor contracting package: Coming soon!

me@christopherchubb.com
christopherchubb.com
@QuantumChubb