A Compact Fermion to Qubit Mapping

Joel Klassen

QIP 2021

PHASECRAFT

About Phasecraft

Phasecraft is a quantum software company.

Our focus is designing algorithms that incorporate a detailed scientific understanding of all the moving parts – from the particular application to the specific hardware platform.

Our main goal is to design viable applications for near-term quantum computers.

We have partnerships with leading quantum hardware companies as well as academic institutions and other industry leaders.

We are hiring full-time post-doc level researchers and PhD level (or earlier) summer interns.

Come and work with us! **www.phasecraft.io**

PHASECRAFT

Joint work (arXiv:2003.06939 + arXiv:2101.10735)

Charles Derby

Phasecraft Ltd. and UCL

Joel Klassen

Phasecraft Ltd.

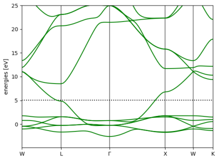
PHASECRAFT

Motivation

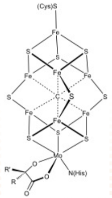
Fermion to qubit mappings are important!

They tell us how to represent fermions on quantum computers.

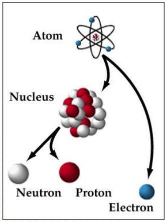
Many promising applications of quantum computers involve simulating fermions



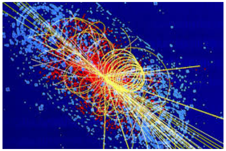
Material Science



Chemistry



Atomic Physics



High Energy Physics

The choice of mapping informs the algorithm design and cost.

PHASECRAFT

Punchline

We introduce a new fermion to qubit mapping – the Compact Encoding – that is very resource efficient and can be tailored to various lattices.

Agenda

- fermion to qubit mappings
- general framework for local encodings
- the compact encoding on the square lattice
- relationship to toric code
- other nice lattices
- the cubic lattice

Notation

Fock space on m modes $F = \bigoplus_{n=0}^{m} \bigwedge^{n} (\mathbb{C}^2)^m$

$$dim(F) = 2^m$$

Fermionic creation and annihilation operators: $a_i^\dagger, a_i \in L(F)$

Majorana operators: $\gamma_{2j} = a_j + a_j^\dagger, \gamma_{2j+1} = (a_j - a_j^\dagger)/i$

$$\gamma_j^2 = 1, \{\gamma_j, \gamma_k\} = 2\delta_{jk}$$

A Fermion By Any Other Name

A fermion to qubit mapping is a correspondence between fermion and qubit states – or equivalently between operators.

e.g. Jordan-Wigner transform:

$$a_i^\dagger \to \frac{1}{2} Z_1 Z_2 \ldots Z_{i-1} (X_i - Y_i)$$

JW is a one-to-one mapping: $|010\rangle \leftrightarrow |0\rangle\,|1\rangle\,|0\rangle$

JW operators grow with the size of the system!

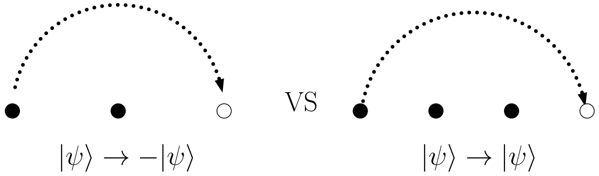
Higher weight interactions lead to longer depth circuits.

The choice of mapping can have significant consequences for the overheads of your algorithm.

PHASECRAFT

Why the JW strings?

Fermionic states have non-local structure


VS
$|\psi\rangle \to -|\psi\rangle$
$|\psi\rangle \to |\psi\rangle$

Nature preserves locality (no-signaling) by **parity superselection**.

Observables in nature never mix even and odd fermion states.

$$a^\dagger a \ ✓ \ , \ a^\dagger \ ✗$$

Nature is local in the **even fermionic algebra** : complex sums of even products of creation/annihilation operators (even products of Majoranas).

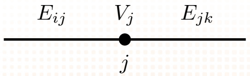
Even Fermionic Algebra

Even fermionic algebra is generated by “edge” and “vertex” operators: $E_{jk} = -i\gamma_{2j}\gamma_{2k}$ and $V_j = -i\gamma_{2j}\gamma_{2j+1}$

e.g. Hopping: $a_i^\dagger a_j + a_j^\dagger a_i \propto V_i E_{ij} + E_{ij} V_j$

Hermitian, traceless, self-inverse and $E_{jk} = -E_{kj}$.

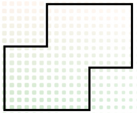
Incident Edge and vertex ops anti-commute. All the rest commute.



$\{E_{ij}, V_j\} = 0$

$\{E_{ij}, E_{jk}\} = 0$

Products of edges in a cycle c yield the identity: $\prod_{E\in c} i\,E = 1$



$= 1$

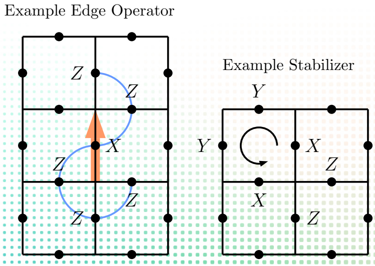
Local Fermionic Encoding

Nature is local in the even fermionic algebra – let's do the same.

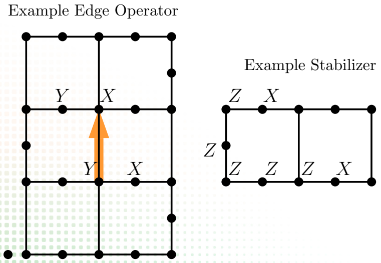
Local Fermionic Encodings map **privileged elements** of the even fermionic algebra to local qubit operators.

Fermionic states mapped into a stabilized code space.
Non-locality is manifest in the entanglement.

Bravyi-Kitaev Super-Fast



Verstraete-Cirac-Ball



State of the Art

No prior local encoding uses fewer than **2 qubits per mode** .

Verstraete-Cirac-Ball has smallest edge ops: worst case **weight-4 edge ops** on square lattice

Any improvement on these numbers reduces the cost of representing fermions on quantum computers.

The compact encoding improves on these numbers – using 1.5 qubits per mode and weight-3 edge ops

General Recipe for Local Encoding I

Start with an undirected connected graph $G = (\mathcal{V}, \mathcal{E})$ – denoting locality.

- Take the set of vertices and directed edges of G + phases:

$$s_G := \{e_{jk}, e_{kj}, v_j, \pm i \ : \ \forall\{j,k\} \in \mathcal{E} \ \forall j \in \mathcal{V}\}$$

- Impose all edge and vertex relations of the even fermionic algebra **except the cycle condition** – yielding a finitely presented group:

$$M_G := \langle s_G \mid \text{even algebra - cycle condition} \rangle$$

- Cycles $\left(\prod_{E \in c} i\, E\right)$ form an abelian normal subgroup $C_G \triangleleft M_G$ and $M_G/C_G = \langle s_G C_G \mid \text{even algebra} \rangle$

- $\mathbb{C}[M_G/C_G]$ is thus the even fermionic algebra ($e_{jk}C_G = E_{jk}$)

PHASECRAFT

General Recipe for Local Encoding II

- Now find a representation $\sigma : M_G \to L(H)$ where H is a multi-qubit system.

- A good choice of σ will map the edges and vertices in G to low weight ops.

- If $\sigma(C_G)$ has a common $+1$ eigenspace $\mathcal{U}$, then we may define a representation of M_G/C_G:

$$\tau(mC_G) := \mathrm{Proj}_{\mathcal{U}} \circ \sigma(m)$$

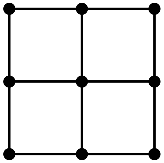
- Thus τ constitutes a representation of the even fermionic algebra in the code space stabilized by $\sigma(C_G)$ – with logical ops given by $\sigma(M_G \setminus C_G)$

All the local encodings we've looked at work this way.

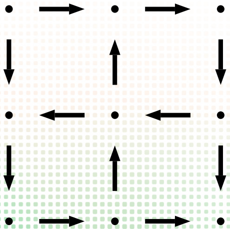
PHASECRAFT

Compact Encoding - Square Lattice

Here we use a square lattice to show how σ is constructed in the compact encoding

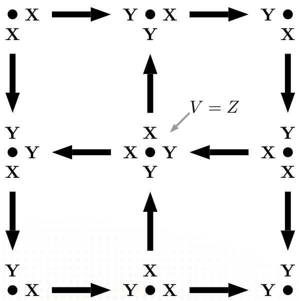


Assign a qubit to every vertex, and give every edge in the graph a carefully chosen orientation.



Start with an ansatz definition of edge and vertex ops:

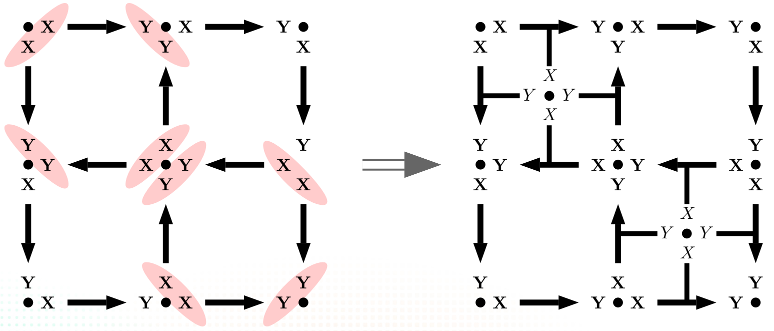
$$\sigma(v_i) = Z_i \ , \ \sigma(e_{ij}) = \begin{cases} X_i Y_j & \text{if } i \text{ points to } j \\ -X_j Y_i & \text{if } j \text{ points to } i \end{cases}$$



All edges anti-commute with incident vertices.

Incident edges anti-commute when they touch head to tail.

When incident edges don't touch head to tail – and thus commute – we add ancillary qubits to resolve the missing anti-commutation relation.

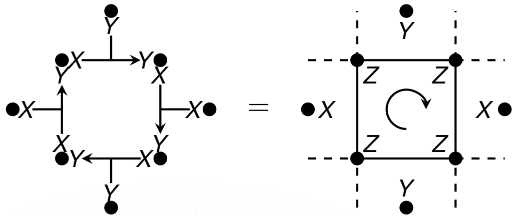


The choice of edge orientation allows us to resolve 4 missing anti-commutation relations with only one ancillary qubit!

1.5 qubits per mode, weight-3 edge ops!

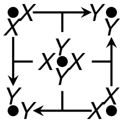
PHASECRAFT

What are the stabilizers? ops of form $\prod_{E \in c} i\,\sigma(E)$. On a planar graph cycles are generated by boundaries of faces:



Non-trivial stabilizers around “even” faces (no qubit on the face)

Problem on “odd” faces (faces with a qubit)



$$= -\mathbb{I}$$

This is not allowed since $\sigma(C_G)$ must have a common $+1$ eigenspace!

Consistently flipping a sign on one edge fixes this ($-\mathbb{I} \to \mathbb{I}$).

The compact encoding is the only local encoding we know for which $C_G \cap \ker(\sigma) \neq 0$.

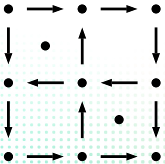
What is the dimension of the stabilized code space $\mathcal{U}$?

$$\begin{aligned}\log_2(\dim(\mathcal{U})) &= \text{qubits} & &- \text{stabilizer generators}\\ &= \text{modes} + \text{odd faces} & &- \text{even faces}\end{aligned}$$

The "disparity" between code space and Fock space:

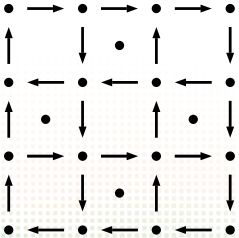
$$\Delta := \log_2(\dim(\mathcal{U})) - \log_2(\dim(F)) = \text{odd faces} - \text{even faces}$$

$\Delta = 0$



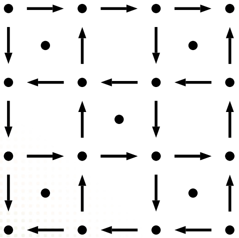
Fock space

$\Delta = -1$



even or odd Fock space

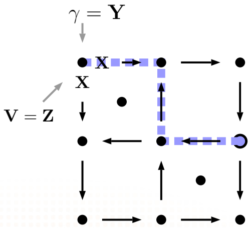
$\Delta = +1$



Fock space + a qubit??

When $\Delta = 0$ the whole fermionic algebra is represented.

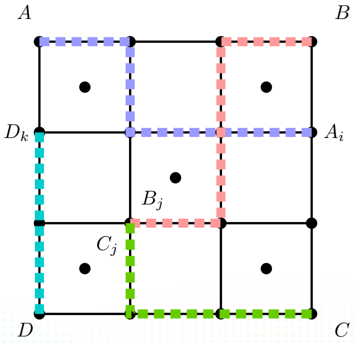
Majorana ops must admit a representation
(γ_{2i} anti-commutes w/ incident edge and vertex ops).



Inject γ_{2i} at one "odd" corner and generate others by multiplying by edge and vertex ops.

Opposite corner is a Majorana hole: $h_{2k} := \gamma_{2k} \prod_j V_j$

When $\Delta = +1$ there are four places (A, B, C, D) to "inject" Majoranas!



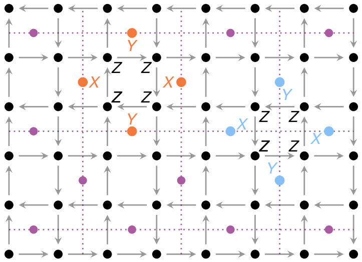
$$A_i = \gamma_{2i} \otimes \mathbb{I}$$
$$B_i = h_{2i} \otimes \tilde{X}$$
$$C_i = h_{2i} \otimes \tilde{Y}$$
$$D_i = h_{2i} \otimes \tilde{Z}$$

Pauli ops on the extra qubit are formed by annihilating holes from different corners.

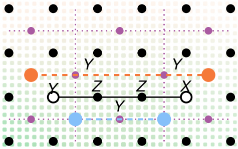
Extra logical qubit is topologically protected...

Surprise toric code!

Stabilizers are toric code stabilizers on face qubits coupled to a parity check on vertex qubits.

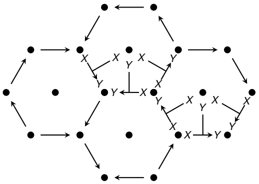


Paired magnetic (*m*) and electric (*e*) excitations of toric code act like fermions. The compact encoding de-penalizes the energy cost of these pairs – pulling them into the code space.

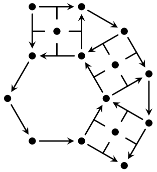


Other Planar Graphs

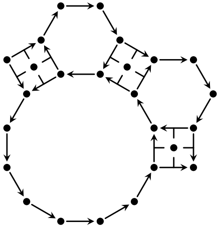
The design strategy of the compact encoding can be applied to other graphs.



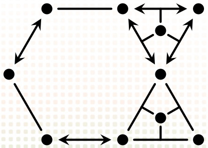
$\Delta = 0$



$\Delta = 0$ (in the bulk)

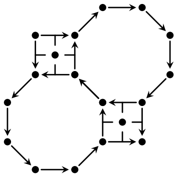


$\Delta = 0$ (in the bulk)

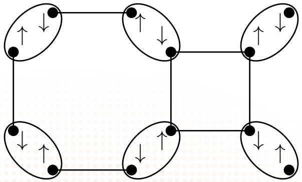


$\Delta > 0$ in the bulk. Disparity grows with system size.

One nice lattice is the 4.8.8 regular tiling.



Can be used to represent spinful-fermions on a square lattice

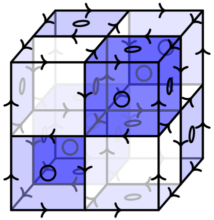


weight 4 hopping terms and fewer than 1.25 qubits per mode

fits naturally on planar hardware layout

3D Cubic Encoding

We can extend our square encoding into the third dimension – every 2d slice looks like the square encoding.



$$\Delta = \text{odd corners}/2 - 1$$

Proof is non-trivial because $\ker(\sigma)$ and C_G are more complicated.

Uses 2.5 qubits per mode and edges are weight 4.

Compare to VC: 3 qubits per mode and weight 4 edge ops.

Some questions:

- Is our construction the best one can do by these metrics?
- The toric code is hidden in the ancillary qubits of the 2d encoding – how should we understand its counterpart in the 3D version?
- Can we encode other types of particles in a similar fashion – by condensing the excitations of existing codes back into the code-space.
- Is there a larger framework for constructing local encodings that subsumes the general framework outlined here?

PHASECRAFT

Thanks!

Please come and say hi to me after the talk :)

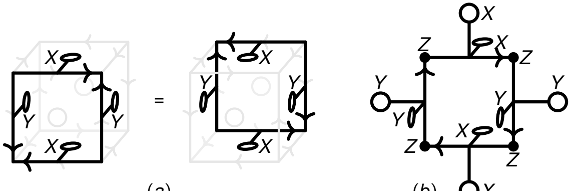
(arXiv:2003.06939 + arXiv:2101.10735)

www.phasecraft.io

PHASECRAFT

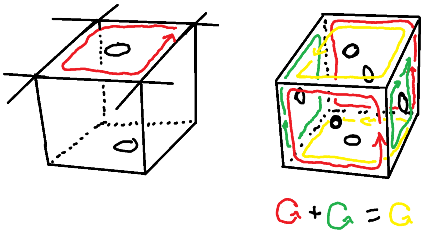
More details about the cubic encoding

Non-trivial stabilizers of cubic encoding



(a) (b)

Cycles in the kernel:



$$\text{rank(stab)} = \text{rank}(C_G) - \text{rank}(C_G \cap \ker(\sigma))$$

$$\text{rank}(\ker(\sigma) \cap C_G) = \text{Isolated Odd Faces} + 2\text{Odd Cells}$$

$$\text{rank}(C_G) = \text{edges - vertices} + 1 = \text{faces} - \text{cells}$$

$$\Delta = (\text{qubits} - \text{rank(stab)}) - \text{vertices}$$

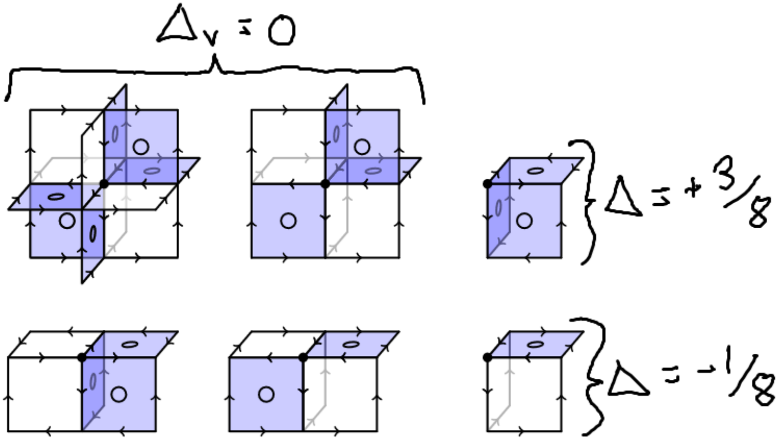
$$\text{iso. odd faces} = \text{odd faces} - 6 * \text{odd cells}$$

$$\text{cells} = \text{odd cells} + \text{even cells}$$

$$\Delta = \text{odd faces} - \text{even faces} - 3 * \text{odd cells} + \text{even cells}$$

Per vertex disparity:

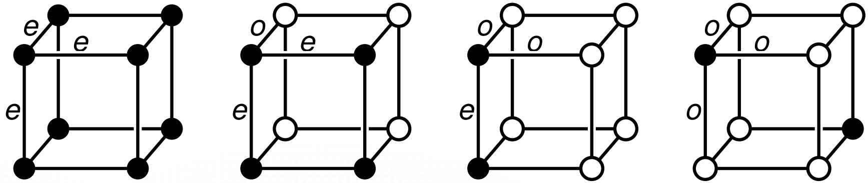
$$\Delta_v = OF_v/4 - EF_v/4 - 3 * OC_v/8 + EC_v/8$$



Only certain possible corner configurations

● odd corner

○ even corner



$$\therefore \Delta = \text{odd corners}/2 - 1$$

Clearly a topological feature – a better proof is likely possible.