Bell Pair Injection and Growing on the Rotated Surface Code

Maxim Sirotin*
Massachusetts Institute of Technology, Cambridge, MA 02138, USA
(Dated: May 9, 2025)

We present a full-circuit analysis of Bell pair injection and growing on the rotated surface code. We quantify the trade-off between the logical error rate and the discard rate as a function of the initial code distance d_{in}, at which the Bell pair is injected, when grown to a final code distance d_f. The logical error rate scales approximately as $\propto (d_f - d_{\text{in}})^{0.4}$, while the discard rate scales as $\propto d_{\text{in}}^2$ for two-qubit physical error rates below 10^{-4}. For two-qubit error rates on the order of 10^{-3}, the discard rate scales linearly with d_{in} and starts saturating around $d_{\text{in}} \approx 15$. These results enable the identification of optimal strategies for Bell pair distillation and encoding, and are relevant for distributed quantum computing in the low-rate Bell pair distribution regime, as well as for fault-tolerant nonlocal quantum sensing.

INTRODUCTION

Useful quantum computing will require millions of physical qubits. However, regardless of the physical platform used to realize a quantum computer, scaling the number of qubits while maintaining control and reliable interconnectivity presents a substantial technical challenge [1]. Distributed quantum computing (DQC) offers a promising pathway to overcome this scalability bottleneck [1–5]. By interconnecting multiple quantum processing modules into a network, DQC enables the execution of large quantum circuits without compromising performance or qubit connectivity.

A key enabler of DQC is the use of shared Bell pairs, where each qubit of the pair is located at a separate node. These Bell pairs provide non-local entanglement, facilitating quantum information transfer via local operations and classical communication. However, because large-scale quantum algorithms typically require error rates in the range of 10^{-10} to 10^{-12} [6, 7], physical Bell pairs must be further encoded into logical Bell pairs using quantum error correction (QEC) codes, such as the surface code [8].

There are several methods for establishing logical Bell pairs between two logical surface code qubits. In the distributed transversal gate approach [9, 10], each data qubit in one node is connected to the corresponding data qubit in another node via a physical Bell pair. In contrast, distributed lattice surgery [4] requires connecting only the boundary of the surface code patch, reducing the number of Bell pairs needed per QEC cycle. Nevertheless, both approaches demand hundreds to thousands of physical Bell pairs to produce a single logical Bell pair [4, 9], a requirement that is particularly difficult to meet when nodes are separated by long distances. For example, state-of-the-art quantum network demonstrations currently achieve entangling rates of only a few hundred Bell pairs per second [11–13].

It is therefore highly desirable to develop methods that make more efficient use of the communication link.

One particularly promising approach with minimal communication overhead is the injection-growing procedure [14–16] (Fig. 1). In this method, a physical Bell pair is first injected into a distance-d surface code, with all faulty events discarded. The code is then grown to larger distances, optionally interleaving or following with distillation. This procedure potentially requires far fewer physical Bell pairs per logical Bell pair, making it especially attractive for long-distance DQC where communication overhead is a major constraint. Previous work showed how a very high physical to logical distillation rate could be achieved [16]. This significantly lowers the networking rate requirements but comes at the expense of a very large local qubit overhead for the distillation procedure. Reducing this overhead is crucial due to the limited number of qubits in current quantum processors. A promising approach for this is to study and optimize the injection and growing step of such protocols, which has not been done in past work.

* sirotin@mit.edu

In this paper, we perform a full-circuit analysis of the Bell pair injection and growth procedure. We quantify the trade-off between logical error rate and discard rate for various injection distances, and use step-by-step growing procedure to the final distances up to $d = 23$.

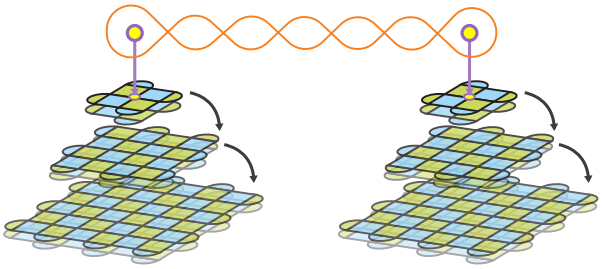

FIG. 1. Concept of the Bell-pair injection-growing procedure. Physical Bell pair (top) is injected into small-distance surface code. Growing of the code distance is then performed.

PROCEDURE FOR BELL PAIR INJECTION AND CODE GROWING

We adapt the injection-growing procedure originally developed for magic states [14, 15], to the case of Bell pairs. For clarity, we refer to *Bell pairs* as distributed Bell pairs and to *EPR pairs* as local Bell pairs between data qubits used during the growing procedure.

(1) Initialization

We begin by preparing two rotated surface code patches of the same distance d, using either the *corner* or *middle* injection pattern (Fig. 3(a), inset) [15]. A CNOT gate is then applied between the target data qubits of the two codes (highlighted with purple circles), creating a physical Bell pair in the entangled state $|00\rangle + |11\rangle$ (Fig. 2(a)).

(2) Stabilizer Measurement and Post-selection

Next, we perform a full round of stabilizer measurements across the lattice. Some stabilizers are deterministic (e.g., X-type stabilizers surrounded by $|+\rangle$ states and Z-type stabilizers surrounded by $|0\rangle$ states), while others produce random outcomes. After the first round, we place detectors on the deterministic stabilizers to catch any errors at this step.

We then perform a second round of stabilizer measurements and compare the results to those from the first round using parity detectors. If any detector signals an error, the attempt is discarded. If no errors are detected, we proceed to the growing phase. From this point on, no further post-selection is performed; all subsequent measurements are used strictly for error correction.

(3) Step-by-Step Growing Using EPR Pairs

To grow the code, we prepare data qubits around the boundary in the entangled $|00\rangle + |11\rangle$ state (Fig. 5(a)). We first merge these EPR pairs near the external stabilizers of the existing code, then measure the stabilizers of the intermediate (grown) code for two rounds:

- After the first round, outcomes should match those of the previous (pre-growth) code. We place detectors to enforce this consistency.
- After the second round, outcomes should match those from the first round. Parity detectors are added to track potential errors.

We then continue to merge the next layer of EPR pairs and finally measure the stabilizers of the full $d + 2$ code. As before:

- The first round of stabilizer measurements should match those of the previous intermediate code.
- Outcomes of subsequent rounds should be consistent with each other.
- This stabilizer measurement cycle is repeated d times.

If necessary, the procedure can be continued to further increase the code distance. If the final code distance has been reached, we measure the logical operator of the final code (of the final logical Bell pair).

Note: The $d \to d+2$ growth process causes the logical operators to rotate. For example, if the logical $\bar{Z}$ operator is the central horizontal line in the original code, it becomes the central vertical line in the $d + 2$ code, and so on in subsequent growth steps.

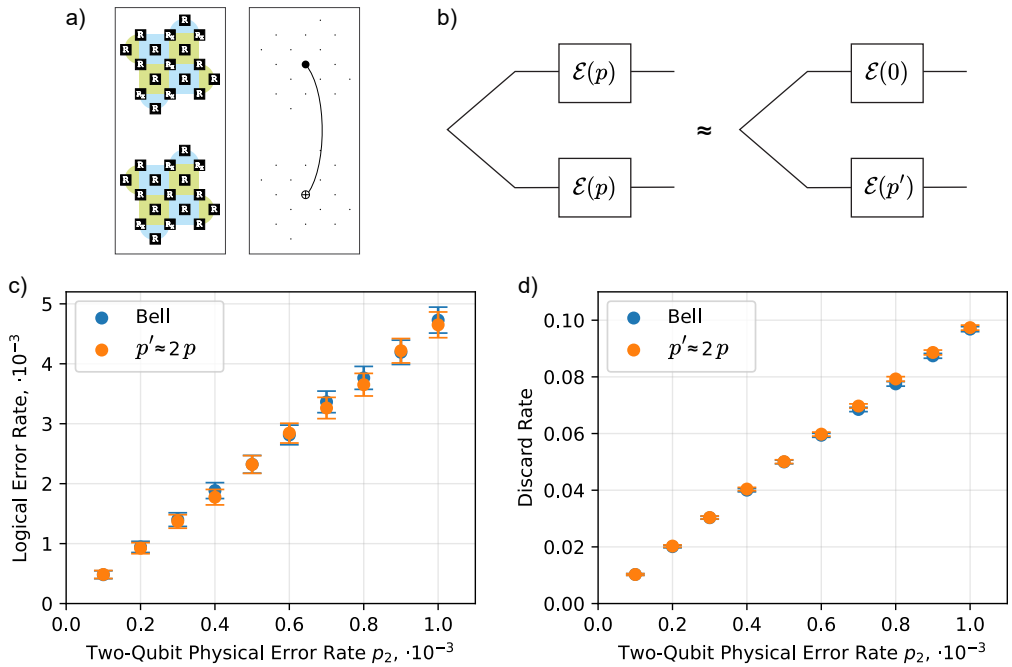

FIG. 2. a) Simulation of the logical Bell pair formation in Stim by initializing $d = 3$ surface codes (left) and performing CNOT between central data qubits (right), followed by the stabilizer measurements on two codes (not shown). b) Approximating simulation of the full Bell pair by simulating only one side of the Bell pair with the effective physical error rate p'. Logical error rate (c) and discard rate (d) for the full Bell pair simulation (blue) and one-side simulation (orange) with the effective physical error rate $p' \approx 2p$.

We emphasize that this step-by-step growth method introduces significant logical errors as the number of steps increases, making it suboptimal for large-scale applications. A more fault-tolerant alternative is to grow the code by merging with pre-prepared patches to the target distance, such as via the $d \to 3d - 4$ scheme [17].

Simulation details are provided in the Appendix A.

RESULTS AND DISCUSSION

We begin by comparing two simulation strategies: (1) full simulation of the logical Bell pair (Fig. 2 (a) and (b, left)), and (2) simulation of only one half of the Bell pair using an effective physical error rate (Fig. 2 (b, right)) [18]. Here, we use middle injection into $d = 3$ rotated surface code. As shown in Fig. 2, the one-sided simulation (orange) exhibits good agreement with the full Bell pair simulation (blue) for the range of physical error rates considered in this work (10^{-4} to 10^{-3}). Therefore, we adopt the one-sided simulation method for subsequent analysis due to its reduced computational cost and simplicity.

We note, however, that for physical error rates exceeding 10^{-3}, the one-sided approximation may no longer be accurate, and full Bell pair simulations may be required to capture correlated error effects.

Next, we compare two injection strategies originally developed for magic state injection in the surface code: *corner injection* and *middle injection* (Fig. 3 (a, inset)). Data qubits in solid blocks are initialized in the $|+\rangle$ state, while those in dashed blocks are initialized in the $|0\rangle$ state.

We follow the injection procedure described earlier and discard all faulty events through post-selection. The resulting logical error rate (LER) and discard rate (DR) are shown in Fig. 3.

We observe that middle injection yields a lower LER compared to corner injection, while the discard rates remain comparable. This is similar to results shown for the magic state injection [15]. Therefore, we adopt middle injection in the remainder of this work as the more optimal strategy.

The performance of the injection procedure also depends on the code distance. We initialize the data qubits

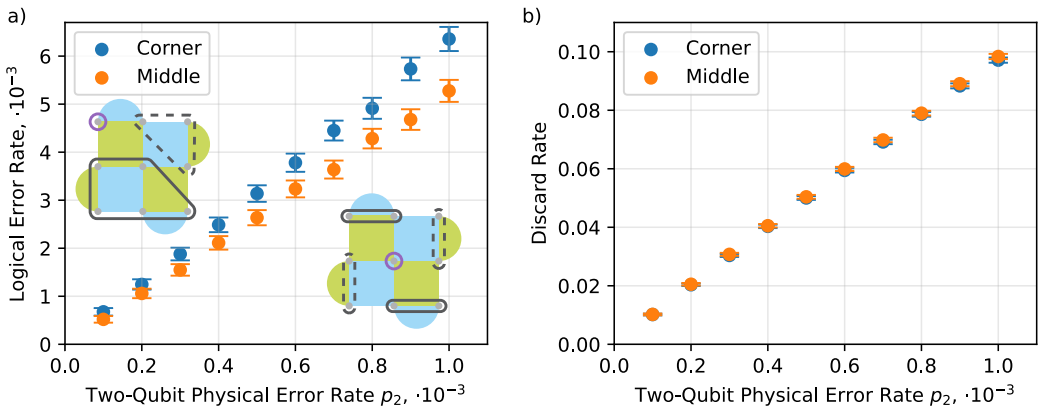

FIG. 3. Comparison of the corner (blue) and middle (orange) Bell pair injection. a) Logical error rate for two approaches. Inset shows the initialization pattern: data qubits circled with solid (dashed) lines are initialized in $|+\rangle$ ($|0\rangle$) state. b) Discard rate for the corner (blue) and middle (orange) Bell pair injection.

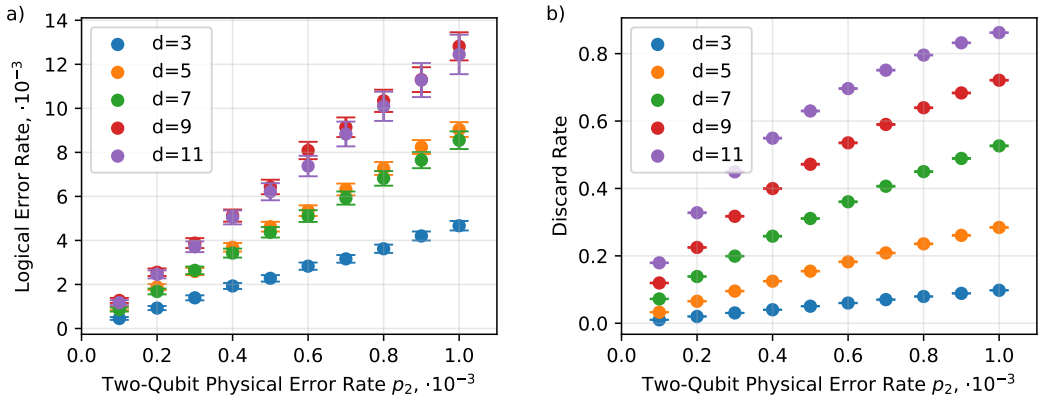

FIG. 4. Comparison of the logical error rate (a) and discard rate (b) for the Bell-pair injection in codes of different distance d.

using a fixed pattern: the top-left and bottom-right quadrants of the data qubits of the code are initialized in the $|+\rangle$ state, while the top-right and bottom-left quadrants are initialized in the $|0\rangle$ state (Fig. 3 (a)).

Fig. 4 (a) shows that LER exhibits a non-trivial dependence on the code distance. Although it generally increases with distance, we observe plateaus: LER values are similar for $d = 5, 7$ and again for $d = 9, 11$. This behavior may be attributed to the qubit initialization pattern, which is fixed across distances for simplicity. It is possible that certain distances allow for more optimal initialization patterns.

In Fig. 4 (b) we see that DR, on the other hand, consistently increases with code distance. At low physical error rates, the DR scales approximately as d_{in}^2, reflecting the number of potentially faulty locations in the code of distance d_{in}. However, as the physical error rate approaches 10^{-4}–10^{-3}, this quadratic dependence saturates and transitions to a linear scaling with distance, as illustrated more clearly in Fig. 5.

In fault-tolerant quantum computation, high code distances are typically required to suppress logical errors. However, this comes at a cost: (1) a higher discard rate for Bell pairs, and (2) increased local qubit overhead during logical Bell pair distillation. To mitigate these overheads, it may be advantageous to inject the Bell pair into a small-distance code and subsequently grow the code to the desired distance, probably performing distillation between growing stages.

We quantify the trade-off between LER and DR by injecting a physical Bell pair into a surface code of distance d_{in}, and then growing it to a final code distance of $d_f = 23$. The growth procedure follows the method described earlier and is illustrated in Fig. 5 (a).

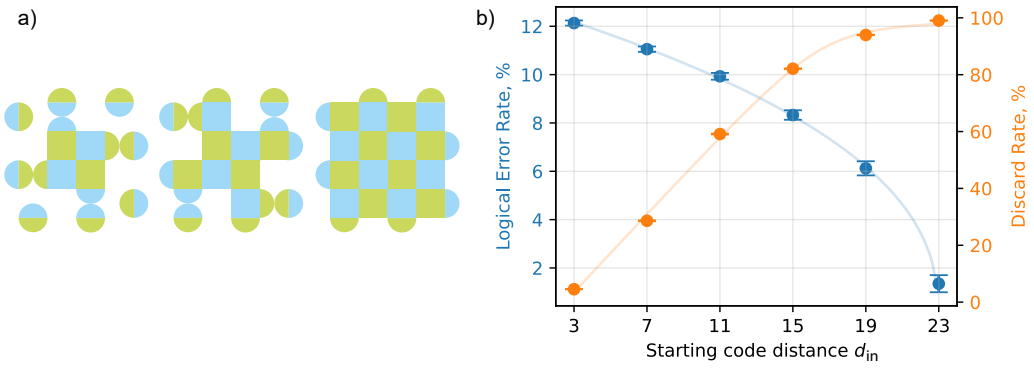

FIG. 5. a) Surface code growing procedure. Distance-3 code is first surrounded by the prepared EPR pairs (left). First, intermediate code is formed (middle); second, final distance-5 code is formed (right). b) Trade-off between logical error rate and discard rate when injecting into code with distance d_{in} and then growing to the code with distance 23.

Figure 5 (b) presents the simulation results (dots) along with their corresponding fits (lines). For this simulation the two-qubit physical error rate is fixed at $p_2 = 0.5 \cdot 10^{-3}$. Details of the fitting procedure are provided in the Appendix B. To model the LER, we use a polynomial dependence on the number of growth steps, as these steps contribute most significantly to logical errors. From the fit, we find that the LER scales approximately as $\text{LER} \propto (d_f - d_{\text{in}})^{0.42 \pm 0.07}$.

As discussed earlier, the DR exhibits quadratic scaling with d_{in} in the low physical error regime, transitioning to a linear dependence with saturation near 1 as the physical error rate increases. To model this behavior, we use a smooth saturation function with a smoothing parameter $\varepsilon = 0.05 \pm 0.03$ and a crossover distance of $d_{\text{in}}^* \approx 15$.

These results enable the identification of optimal strategies for preparing distributed logical Bell pairs on the surface code, given a specific distillation protocol, Bell pair distribution rate, and the available number of local qubits at each node. While more efficient growth procedures may be employed, the presented discard rate scaling is independent of the growth path and can be used directly. Although alternative approaches based on quantum low-density parity-check (qLDPC) codes are being actively developed [18], the injection-growing method presented here offers minimal communication overhead for surface codes, making it a promising candidate for implementation on near-term quantum hardware.

CONCLUSION AND OUTLOOK

In this work, we study Bell pair injection and growth on the rotated surface code. We compare two injection strategies—corner injection and middle injection—and demonstrate the superior performance of the latter. We also analyze the trade-off between the final logical error rate and the discard rate when growing a code from an initial distance d_{in} to a final distance of $d_f = 23$. These results are particularly relevant for distributed quantum computing and nonlocal quantum sensing.

Our approach employs a step-by-step code growth procedure, increasing the code distance via $d \to d + 2$ merges with surrounding EPR pairs. This method, while straightforward, imposes a high logical error rate on the final state as the number of growing steps increases. A more optimal strategy would involve rapid distance growth by merging with pre-prepared patches; one such protocol follows a $d \to 3d - 4$ scheme [17]. Notably, the discard rate is independent of the growing path and depends only on the injection code distance d_{in}. Interleaving with distillation steps [16], one can as well perform cultivation-like preparation of logical Bell pairs [19].

One particularly interesting application of Bell pair injection and growth is in nonlocal quantum sensing, such as quantum-enhanced telescopes [20, 21]. In such settings, an arbitrary-phase $|01\rangle + e^{i\varphi}|10\rangle$ state can be logically encoded to enable fault-tolerant implementation of phase estimation algorithms. We also see that the discard rate and logical error rate are independent of the phase of the injected Bell pair, making this approach well suited to encoding Bell pairs with unknown phase in distributed

quantum sensing scenarios.

An interesting direction for future work is the logical encoding of more complex quantum states. For example, encoding a statistical mixture of two faint astronomical sources (such as a star and its exoplanet) may enable novel fault-tolerant quantum data processing techniques. Similarly, generalizing this protocol to W-like entangled states distributed across multiple quantum sensors could further broaden the scope of fault-tolerant quantum computing-enhanced quantum sensing [22].

Appendix A: Simulation Details

We use the Python-based stabilizer circuit simulator `Stim` [23] to simulate the injection-growing procedure. Owing to the fourfold symmetry of the middle injection approach, we adopt a custom qubit coordinate grid for scalable and structured code construction. Each qubit is assigned a triplet of integers:

- r: radial distance from the center (even values for data qubits, odd for ancilla qubits),
- a: side index ($a \in \{0, 1, 2, 3\}$ corresponding to top, right, bottom, and left),
- b: position index along the side.

The qubit index is then computed as $q = 2(r-1)r + 1 + ra + b$, allowing for step-by-step qubit initialization and seamless scalability of the growing procedure.

Post-selection is performed using the post-selection mask functionality of the `Sinter` package, which enables efficient Monte Carlo sampling of QEC circuits. We select specific detectors to post-select on. For decoding, `Sinter` employs the Minimum Weight Perfect Matching decoder implemented in the `PyMatching` library [24].

The error model used is as follows:

- Each single-qubit gate is followed by a depolarizing error: X, Y, or Z with probability $p_1/3$.
- Each two-qubit gate is followed by a non-identity Pauli product from $\{I, X, Y, Z\}^{\otimes 2} \backslash \{II\}$ with probability $p_2/15$.
- Each qubit initialization and measurement is flipped with probability p_{in} and p_{m}, respectively.

In this work, we use the parameter setting $p_1 = p_{\text{in}} = p_{\text{m}} = p_2/10$. Here we also assume that the distributed CNOT also has error probability p_2, and for the one-part simulation the injected qubit initialization is followed by a depolarizing error with probability $2p_2$, as well as all the errors are scaled as $p' = 2p$.

Simulation results are processed assuming a binomial distribution. Logical error rate (LER) means are calculated as

$$\text{LER} = \frac{\text{errors}}{\text{shots} - \text{discards}},$$

and discard rate (DR) means are computed as

$$\text{DR} = \frac{\text{discards}}{\text{shots}}.$$

Error bars represent 3σ confidence intervals, where $\sigma = \sqrt{\frac{1}{n}p(1-p)}$ is the approximate standard error of a binomial distribution (here $p =$ LER or DR). Source code is available online: [25].

Appendix B: Fitting parameters

LER is fitted with a function $g(d_{\text{in}}) = \alpha(d_f - d_{\text{in}})^r$, showing polynomial scaling with the amount of steps of growing. DR is fitted with the general smooth saturation function $f(d_{\text{in}}) = \frac{1}{2}(\sqrt{\varepsilon + (x'+1)^2} - \sqrt{\varepsilon + (x'-1)^2})$, where $x' = \frac{d_{\text{in}} - x_0}{b}$, which has linear scaling before saturating and asymptotically approaching 1. Fitting parameters are shown in the Table I below.

α	r	ε	x_0	b
0.035±0.006	0.42±0.07	0.05±0.03	2.5±0.3	14.2±0.8

TABLE I. Fitting parameters.

ACKNOWLEDGMENTS

We wish to acknowledge the support of Johannes Borregaard, Frederik Marqversen, Gefen Baranes, and Juan Pablo Bonilla Ataides throughout the project.

[1] D. Main, P. Drmota, D. Nadlinger, E. Ainley, A. Agrawal, B. Nichol, R. Srinivas, G. Araneda, and D. Lucas, Distributed quantum computing across an optical network link, Nature , 1 (2025).

[2] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects, Physical Review A **89**, 022317 (2014).

[3] S. de Bone, P. Möller, C. E. Bradley, T. H. Taminiau, and D. Elkouss, Thresholds for the distributed surface code in the presence of memory decoherence, AVS Quantum Science **6** (2024).

[4] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, Fault-tolerant connection of error-corrected qubits with noisy links, npj Quantum Information **10**, 58 (2024).

[5] D. Barral, F. J. Cardama, G. Díaz-Camacho, D. Faílde, I. F. Llovo, M. Mussa-Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, *et al.*, Review of distributed quantum computing: from single QPU to high performance quantum computing, Computer Science Review **57**, 100747 (2025).

[6] J. Lee, D. W. Berry, C. Gidney, W. J. Huggins, J. R. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, PRX Quantum **2**, 030305 (2021).

[7] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, Assessing requirements to scale to practical quantum advantage, arXiv preprint arXiv:2211.07629 (2022).

[8] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, *et al.*, Quantum error correction below the surface code threshold, Nature (2024).

[9] A. G. Fowler, D. S. Wang, C. D. Hill, T. D. Ladd, R. Van Meter, and L. C. Hollenberg, Surface code quantum communication, Physical Review Letters **104**, 180503 (2010).

[10] J. Stack, M. Wang, and F. Mueller, Assessing teleportation of logical qubits in a distributed quantum architecture under error correction, arXiv preprint arXiv:2504.05611 (2025).

[11] S. Meesala, D. Lake, S. Wood, P. Chiappina, C. Zhong, A. D. Beyer, M. D. Shaw, L. Jiang, and O. Painter, Quantum entanglement between optical and microwave photonic qubits, Physical Review X **14**, 031055 (2024).

[12] C. Knaut, A. Suleymanzade, Y.-C. Wei, D. Assumpcao, P.-J. Stas, Y. Huan, B. Machielse, E. Knall, M. Sutula, G. Baranes, *et al.*, Entanglement of nanophotonic quantum memory nodes in a telecom network, Nature **629**, 573 (2024).

[13] Y. Wang, A. N. Craddock, J. M. Mendoza, R. Sekelsky, M. Flament, and M. Namazi, High-fidelity entanglement between a telecom photon and a room-temperature quantum memory, arXiv preprint arXiv:2503.11564 (2025).

[14] Y. Li, A magic state's fidelity can be superior to the operations that created it, New Journal of Physics **17**, 023037 (2015).

[15] L. Lao and B. Criger, Magic state injection on the rotated surface code, in *Proceedings of the 19th ACM International Conference on Computing Frontiers* (2022) pp. 113–120.

[16] C. A. Pattison, G. Baranes, J. Ataides, M. D. Lukin, and H. Zhou, Fast quantum interconnects via constant-rate entanglement distillation, arXiv preprint arXiv:2408.15936 (2024).

[17] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, Code deformation and lattice surgery are gauge fixing, New Journal of Physics **21**, 033028 (2019).

[18] J. Ataides, H. Zhou, Q. Xu, G. Baranes, B. Li, M. D. Lukin, and L. Jiang, Constant-overhead fault-tolerant bell-pair distillation using high-rate codes, arXiv preprint arXiv:2502.09542 (2025).

[19] C. Gidney, N. Shutty, and C. Jones, Magic state cultivation: growing T states as cheap as CNOT gates, arXiv preprint arXiv:2409.17595 (2024).

[20] D. Gottesman, T. Jennewein, and S. Croke, Longer-baseline telescopes using quantum repeaters, Physical Review Letters **109**, 070503 (2012).

[21] E. T. Khabiboulline, J. Borregaard, K. De Greve, and M. D. Lukin, Optical interferometry with quantum networks, Physical Review Letters **123**, 070504 (2019).

[22] R. R. Allen, F. Machado, I. L. Chuang, H.-Y. Huang, and S. Choi, Quantum computing enhanced sensing, arXiv preprint arXiv:2501.07625 (2025).

[23] C. Gidney, Stim: a fast stabilizer circuit simulator, Quantum **5**, 497 (2021).

[24] O. Higgott, Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching, ACM Transactions on Quantum Computing **3**, 1 (2022).

[25] M. Sirotin, `https://github.com/MaximSirotin/BellPairs` (2025).