Local quantum memories and early fault-tolerant algorithms

Louis Paletta

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Mémoires quantiques locales et premiers algorithmes tolérants aux fautes

Local quantum memories and early fault-tolerant algorithms

Soutenue par

Louis Paletta

Le 17 Octobre 2025

École doctorale n°564

Physique en Île de France

Spécialité

Physique Quantique

Composition du jury :

Benoît Douçot Directeur de Recherche, CNRS	*Président*
Damian Markham Directeur de Recherche, CNRS	*Rapporteur*
Robert König Associate Professor, Technische Universität München	*Rapporteur*
Irène Marcovici Professeure, Université de Rouen Normandie	*Examinatrice*
Mazyar Mirrahimi Directeur de recherche, Inria	*Directeur de thèse*
Anthony Leverrier Chercheur, Inria	*Codirecteur de thèse*

ENS | PSL UNIVERSITÉ PARIS

Inria

Remerciements

Cela fait maintenant un peu plus de trois ans que j'ai commencé ce doctorat : je me rappelle de mes premières lectures d'articles de recherche, laborieuses, mais fascinées par le monde nébuleux de l'information quantique. Des nuages, il en restera sans doute toujours, mais quelle joie d'y voir tout de même un peu plus clair ! J'ai beaucoup appris pendant ces années, et je le dois avant tout aux personnes avec lesquelles je les ai partagées ; je veux maintenant les en remercier chaleureusement.

Merci d'abord Mazyar, de m'avoir accueilli dans l'équipe et de m'avoir fait confiance. Merci de m'avoir donné le goût de la recherche, merci pour tes questions, tes conseils et ton aide pour cette thèse et la suite de ma vie de chercheur. Merci Anthony, Christophe et Alain pour toutes les discussions passionnantes auxquelles j'ai eu la chance de participer, merci pour votre temps, vos remarques et vos explications qui ont tellement enrichi ce travail. Surtout, merci à tous les quatre pour qu'au cours de ce doctorat, faire de la recherche soit toujours resté un plaisir.

Mais aussi passionnante que soit la physique, la vie de laboratoire serait bien triste sans de précieux collègues et amis avec qui la partager. Merci Michiel, Christian et Ronan de m'avoir accueilli dans l'équipe quand j'étais encore un jeune stagiaire, et de m'avoir donné envie de poursuivre en doctorat ; je me rends compte maintenant comme le temps peut être précieux lors d'une dernière année de doctorat, merci d'avoir gardé un peu du vôtre pour moi. Merci Diego pour ta curiosité insatiable, nos pauses café et chocolat chaud passées à discuter de science parfois, et de tout le reste souvent, me manqueront. Merci Angela pour avoir supporté mon italien approximatif avec bienveillance, merci Sophie pour ta bonne humeur communicative, et merci Thomas d'avoir toujours été capable de me redonner le sourire. Merci à tous les autres qui ont partagé un bout de ce doctorat avec moi. Je ne vous oublie pas.

Je ne peux pas écrire ces lignes sans noter, avec une pointe de mélancolie, qu'elles marquent aussi la fin de mes années à Paris ; et je tenais à remercier

celles et ceux qui les ont rendues si belles — et avec qui j'ai hâte de vivre les suivantes. Merci Anthony pour être dans la majorité de mes plus beaux souvenirs, depuis tout petit jusqu'à ma soutenance de thèse ; tu le sais, je n'aurais sans doute jamais eu l'inspiration de commencer ce doctorat sans toi. Merci Vincent pour les mercredis soirs micro en main, les fêtes de l'huma inoubliables et suffisamment de fous rires pour toute une vie ; je manquerais d'un peu de folie sans t'avoir rencontré. Merci Léon pour les verres à la butte, et les aventures toujours plus improbables et toujours moins planifiées ; nos soirées à la colocation me manqueront. Merci Martine de m'avoir gardé près de toi depuis le Lycée, pour avoir été là quand ça allait, et quand ça n'allait pas. Merci Jonnathan pour ton calme et ta sérénité, pour pouvoir toujours compter sur toi pour partager les mêmes questions autour d'un dernier verre. Merci Federico pour le soleil que tu apportes dans la grisaille parisienne, ton sourire et tes histoires. Merci Bachar pour les intrigues du comité, les sessions surfs et pour toujours être capable de s'inventer une nouvelle vie. Merci Marie-Fleur pour m'avoir donné un peu de temps et de simplicité quand j'en avais si peu ; ces derniers mois auraient été moins doux sans toi. Merci à tous les autres — Andrea, Mathilde, Lucas, Lou, Capucine, Gaïa, Valentin, Héloïse, Alexis, Lara, Chloë, Elsa, Anna, Clément — je sais la chance que j'ai de vous avoir à mes côtés.

Enfin, merci à ma famille d'avoir toujours su m'offrir une oasis de calme et de tranquillité, préservée de l'excitation du monde de la recherche ; et de m'avoir appris ce qui sera toujours plus important que le contenu de n'importe quel manuscrit de thèse. Merci à mes grands-mères pour continuer de m'inspirer par leur courage, et à mon grand-père pour m'avoir transmis un peu de sa curiosité. Merci grand frère pour toujours rester mon exemple, et merci petite sœur pour toujours nous garder tous ensemble. Finalement, à mes parents, pour tout l'amour que vous m'avez toujours donné, et que j'espère être capable de vous rendre un jour — merci papa, merci maman.

« Et chaque jour, pour l'ouvrier, qui commence
à bâtir le monde, le monde commence. »

Antoine de Saint-Exupéry, Courrier Sud

Résumé

Introduction. On peut commencer par une simple remarque, pourtant fondamentale : l'information ne peut exister indépendamment d'un support, qu'il s'agisse, par exemple, des pages d'un livre, ou des états électroniques d'un transistor. De ce constat on peut tirer une conséquence immédiate, la manipulation de l'information, c'est-à-dire la resolution de problèmes, obéit aux lois du monde physique [97]. Mais si les technologies de l'information contemporaines peuvent être décrites par les lois de la physique dite classique, on sait depuis le début du XX^{e} siècle que ces lois émergent de principes plus fondamentaux, ceux de la mécanique quantique. Se pose alors naturellement la question de la construction d'une machine plus générale, capable de manipuler l'information sous sa forme la plus fondamentale : un ordinateur quantique.

Un calcul quantique peut se concevoir comme l'évolution d'un ensemble de bits quantiques, ou qubits, selon les lois de la mécanique quantique — d'un état initial encodant l'entrée d'un problème, à sa solution représentée par l'état final. L'intérêt pour ce modèle de calcul date de la découverte d'algorithmes quantiques résolvant des problèmes auparavant intractables pour les ordinateurs classiques. Par exemple, la décomposition d'un entier naturel en produits de nombres premiers, dont la difficulté est à la base des techniques de cryptographies contemporaines, peut par exemple être résolu efficacement sur un ordinateur quantique via l'algorithme de factorisation introduit par Peter Shor en 1994 [138]. L'application d'un algorithme quantique nécessite encore de disposer d'un ordinateur quantique, qu'il reste à construire alors que la fabrication et le contrôle des interactions d'un grand nombre de qubits de manière totalement fiable reste hors de portée en l'état actuel des connaissances. L'approche établie pour construire un ordinateur quantique fiable à partir de briques élémentaires imparfaites, c'est à dire des qubits sujets aux erreurs, consiste à combiner l'ajout de redondance dans le calcul, à un système capable de détecter et de corriger d'éventuelles erreurs avant qu'elles ne s'accumulent ou ne se propagent au fil du calcul. La conception d'une telle machine est l'objet de la théorie du *calcul*

quantique tolérant aux fautes, qui offre en principe une solution au problème des erreurs via le *théorème de seuil* [3, 116, 86, 85], mais au prix d'une complexité expérimentale additionnelle. Il s'agit donc de concevoir des architectures de tolérance aux fautes plus simples à mettre en place et plus performantes, rendue à portée des plateformes expérimentales contemporaines. Ce manuscrit se propose d'aborder ce problème, et d'y apporter des pistes de solutions potentielles.

Dans une première partie, on commence par rappeler les bases de l'information quantique, avant d'introduire les concepts de correction d'erreur quantique et de tolérance aux fautes, en se concentrant sur les codes stabilisateurs [65]. Dans un code stabilisateur, k qubits logiques sont encodés dans l'espace de Hilbert de n qubits physiques en imposant $n-k$ contraintes indépendantes. Celles-ci sont définies par des projections sur les sous-espaces propres d'un ensemble d'opérateurs de Pauli qui commutent entre eux, appelés *stabilisateurs*. La réalisation d'un ordinateur quantique tolérant aux fautes à partir d'un code stabilisateur nécessite encore de résoudre deux problèmes : (*i*) être capable de détecter et de corriger les erreurs à partir du résultat des mesures des stabilisateurs, et (*ii*) s'assurer que la propagation des erreurs reste limitée lors de la réalisation d'opérations logiques élémentaires. Dans les deux cas, ces problèmes ne peuvent être abordés indépendamment des contraintes propres aux différentes plateformes physiques. Le cadre de résolution du premier problème, dit problème de décodage, définit dans une large mesure le type de surcouche architecturale nécessaire à la correction d'erreur. En particulier, dans l'approche usuelle, une unité de calcul classique centralisée est chargée de proposer en temps réel une correction à partir des résultats de mesure des stabilisateurs, ce qui nécessite une connectivité accrue et une latence et rapidité de calcul maitrisée. En opposition avec cette vision, dans une architecture basée sur un *décodeur local*, le calcul classique est distribué pour obtenir un haut niveau de parallélisation et potentiellement pouvoir être implémenté au plus proche du système quantique, pour une latence et des besoins en connectivité réduits. Si cette approche inspirée des automates cellulaires simplifie la réalisation d'une mémoire quantique fiable, c'est cependant souvent au prix d'une perte notable de performance, dont la réduction sera l'objet des Chapitres 2 et 3, basés sur le travail publié dans [119]. Une mémoire quantique fiable est une condition sans doute nécessaire à la construction d'un ordinateur quantique, mais elle n'est bien sûr pas suffisante puisqu'un calcul nécessite en plus d'être capable de modifier l'état de la dite mémoire, c'est à dire d'appliquer des opérations

Figure 1: (a) Représentation du code de répétition quantique unidimensionnel. On considère un réseau unidimensionnelle périodique, c'est à dire formant un cycle, et dont on assigne un qubit à chaque arrête, et un stabilisateur de type Z (mesure de parité, en bleu) à la paire d'arrête adjacente à chaque sommet. (b) Les erreurs d'inversion de bit (en rouge) sont détectées à leur frontière par les mesures de parité, dont les résultats -1 sont appelés des défauts représentés en bleu foncé. La tâche de décodage consiste à apparier ces défauts à l'aide d'une correction (en bleu clair) de poids minimal.

logiques. En particulier, la protection de l'information quantique par sa délocalisation sur plusieurs qubits physiques nécessite l'indépendance des erreurs entre les différents qubits. Si cette hypothèse peut être raisonnable dans certains cas, elle cesse en général de l'être lorsque les qubits interagissent entre eux : dans ce cas, les erreurs se propagent d'un qubit à l'autre et ne peuvent plus être considérées comme indépendantes. Il existe des constructions théoriques générales assurant que la propagation des erreurs lors du calcul reste limitée, mais ces propositions restent encore hors de portée des plateformes expérimentales actuelles, malgré des progrès impressionnants au cours des dernières années [17, 5, 131, 1]. En considérant des algorithmes spécifiques, il est cependant possible de simplifier l'architecture et de l'adapter aux contraintes et possibilités de plateformes particulières. Dans cette optique, le Chapitre 4, basé sur le travail publié dans [120], se propose de construire un protocole d'avantage quantique, c'est à dire intractable pour un ordinateur classique, tolérant aux fautes et proche des capacités actuelles des plateformes d'atomes neutres [16].

Mémoires quantiques locales. Un exemple parlant d'architecture local de correction d'erreur, dans le cas classique, est la *règle de Toom*, du nom de son inventeur. Un bit d'information, encodé dans les états 0^n et 1^n d'une grille binaire bidimensionnelle de largeur $\sqrt{n}$, peut être mémorisé pour un temps exponentiel en $\sqrt{n}$, malgré des erreurs indépendantes et identiquement distribuées dans l'espace et dans le temps, via l'application uniforme et répétée d'un simple vote de majorité entre chaque bit et ses voisins « Est » et « Sud ». Il suffit ensuite de réécrire la règle locale à partir des mesures des parité entre voisins, pour adapter la construction à la protection d'un qubit contre les erreurs d'inversion de bit. Le problème est aue cette protection est sous-optimale par rapport à une

décodeur	dim.	seuil	principe
règle de Toom [146]	2D	7.7%	Un îlot d'erreur est corrigé successivement par son coin Sud-Est
Gács [61, 60]	1D	existe mais inconnu	Structure hierarchique avec auto-simulation à surcoût constant
Tsirelson [44, 10]	1D*	1.4%	Un bloc d'erreur est récursivement séparé en blocs plus petits, corrigeables localement
Harrington [74]	1D*	2.0%	Implémentation locale d'un décodeur par groupe de renormalisation
médiation de champ [77, 78]	1D*	✗	Les variables locales du décodeur encode un champ de potentiel médiant une attraction entre les défauts
règles à signaux [ce travail]	1D*	6.6%	Une attraction entre les défauts est transmise par un échange de signaux binaires

Table 1: Décodeurs locaux du code de répétition. Le label 1D* indique que chaque site du code de répétition 1D est équipé d'une mémoire de taille logarithmique. La valeur du seuil est donnée pour un modèle d'erreur indépendant et identiquement distribué dans l'espace (qubits de data et résultats de mesure) et dans le temps.

architecture basée sur un *décodeur global*, où l'augmentation du temps de vie peut être exponentielle en n, en utilisant seulement les mesures de parité entre voisins dans une chaîne unidimensionnelle de qubits. Le Chapitre 2 propose une solution pour chacune de ces lacunes : (i) une modification de la règle de Toom basée sur l'incorporation de permutations des qubits pour retrouver un temps de vie exponentiel en n, et (ii) une nouvelle règle intitulée la *règle de cisaillement* également basée sur des votes de majorité locaux et des permutations de qubit, mais cette fois applicable sur un arrangement unidimensionnel de qubits, tout en offrant de meilleures performances pour des systèmes de petites tailles ($\lesssim 30$ qubits). La règle de cisaillement peut être comprise comme une version quantique d'automates cellulaires existants — en particulier les automates GKL (Gács–Kurdyumov–Levin) et TLV (*Two-Lines Voting*) — et souffre, comme eux, d'une saturation du temps de vie de l'information encodée au-delà d'une certaine taille de système. Ce problème est résolu dans le Chapitre 3, en dotant les sites de mesure d'une petite mémoire classique, de l'ordre de la dizaine de bits, qu'on peut raisonnablement considérer comme fiable.

Le Chapitre 3 se concentre sur des architectures ou le décodage local est

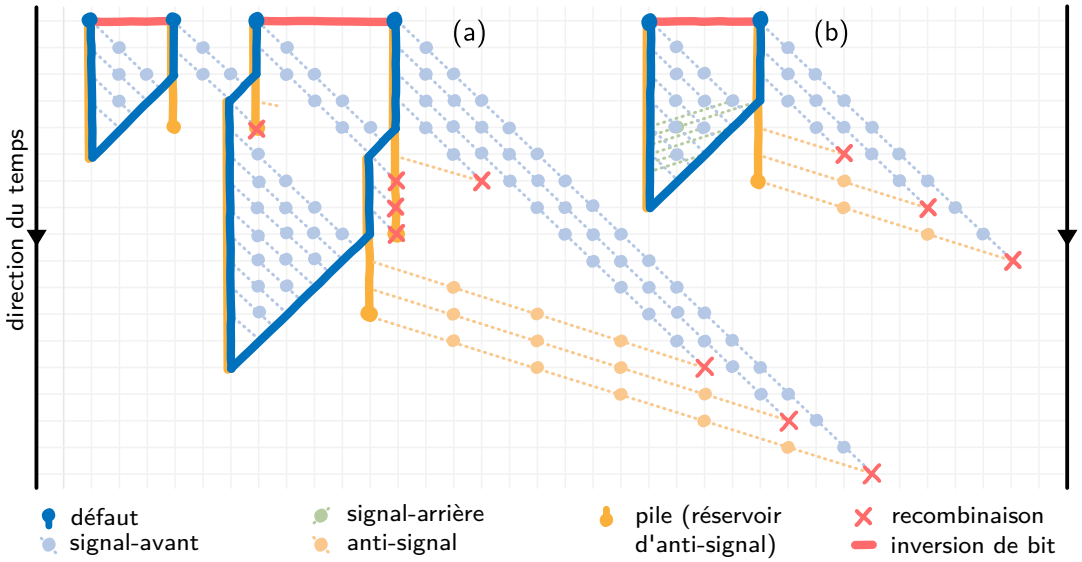

Figure 2: (a) Représentation espace-temps simplifiée de la correction d'un groupe d'erreurs complexe, où les anti-signaux ne sont représentés qu'au moment de leur création lorsque celui-ci se situe à gauche du syndrome le plus à droite, et où les signaux-arrière ne sont pas représentés. (b) Représentation espace-temps de l'effacement d'un groupe d'erreurs simple, le temps s'écoulant vers le bas. Chaque défaut envoie des signaux-avant à sa droite jusqu'à ce que les signaux-avant envoyés par le défaut de gauche atteignent le défaut de droite qu'ils déplacent vers la gauche, jusqu'à son appariement avec le défaut de gauche. Les signaux-avant ayant provoqué un déplacement du défaut se transforment en signaux-arrière, qui se propagent dans la direction opposée pour se recombiner avec la pile de gauche. Les signaux-avant envoyés par le défaut de droite se recombinent avec des anti-signaux plus rapides, créés lors de la décrémentation du réservoir d'anti-signaux de droite lorsqu'elle ne coïncide plus avec un défaut.

réalisé par l'intermédiaire de petites unités de calcul distribuées le long des sites de stabilisateurs du code quantique, ici du code de répétition unidimensionnel illustré dans la Figure 1. Puisqu'en une dimension, une mesure de parité impaire, appelée un *défaut*, correspond à une frontière d'un bloc d'erreurs, le problème de décodage consiste à apparier les mesures de parité impaire selon une mesure de proximité. On introduit une famille de décodeurs locaux, appelés *règles à signaux*, pour lesquels cet appariement est réalisé en simulant une attraction entre les défauts interprétés comme des particules ponctuelles. L'attraction est médiée par la création et la propagation de signaux binaires encodés dans les mémoires classiques du système. La plus simple des deux instances de règles à signaux introduites est illustrée par la Figure 2 et fonctionne de la façon suivante : à chaque itération, un défaut émet un *signal-avant*, qui se propage vers la droite jusqu'à rencontrer un autre défaut. Dans ce cas, le défaut rencontré se déplace d'un site vers la gauche et le signal-avant devient un *signal-arrière* se déplaçant rapidement vers la gauche à chaque itération. Pour garder la trace des signaux-avant qui ont été émis, un réservoir local d'*anti-signaux* appelée *pile* est incrémenté à chaque émission d'un signal-avant par le même site, puis est décrémenté à la réception d'un signal-arrière. Enfin, lorsqu'une pile non-nulle n'est plus associée à un défaut (par exemple parce que celui-ci s'est déplacé vers la gauche), elle libère des anti-signaux qui se déplacent rapidement vers la droite et dont le but est de se recombiner avec les signaux-avant ou signaux-arrière restants, afin que le système classique retourne à sa configuration initiale (la configuration zéro — i.e. absence de signaux) à la fin du processus de correction. Pour de meilleures performances, il est possible d'utiliser une variante obtenue par symétrisation de la première règle, c'est à dire par l'émission de signaux-avant à la fois vers la gauche et vers la droite, au prix de doubler la quantité de mémoire classique nécessaire sur chaque site. Une étude numérique de ces règles démontre que les décodeurs locaux sont capables d'approcher les performances optimales du code de répétition, notamment en termes de seuil. Pour la première règle, dont l'intérêt est avant tout son accessibilité à l'analyse formelle de par sa simplicité, il est possible de prouver un théorème de seuil dans un modèle d'erreur simplifié ou les erreurs sont restreintes à l'état initial du système. Formellement, en dessous d'une certaine probabilité constante d'inversion de qubit lors de l'initialisation du système, l'erreur initiale est corrigée et le système classique retourne à sa configuration zéro avec une probabilité exponentiellement proche de 1. La preuve est obtenue par la combinaison de la

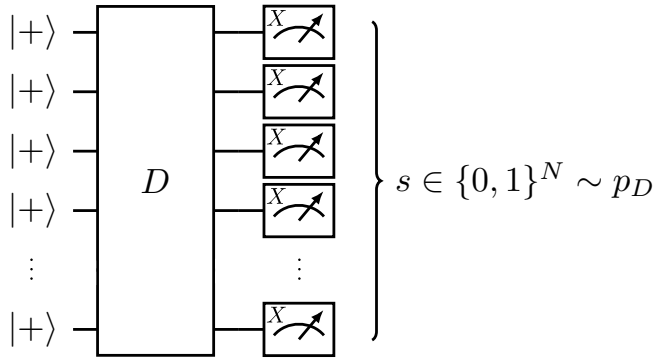

Figure 3: Les circuits IQP sur N qubits sont définis par l'application d'un unitaire D diagonal dans la base computationnelle, et où les qubits sont initialisés et mesurés dans la base X. Pour les circuits sIQP, D est un circuit de profondeur logarithmique (en espérance) généré à partir de portes T et CS.

capacité de la règle à effacer un îlot d'erreur en temps proportionnel à sa taille, et une technique générale de décomposition hiérarchique d'une configuration d'erreurs en blocs isolés de tailles croissantes [29, 74, 60, 93].

Premiers algorithmes tolérants aux fautes. Une démonstration expérimentale de l'avantage du calcul quantique sur le calcul classique ne peut avoir de sens physique sans prendre en compte l'effet du bruit et des erreurs sur le calcul. Ainsi, certains problèmes d'échantillonnage, où l'on cherche à tirer une chaîne binaire selon la distribution de probabilité associée aux mesures finales d'un circuit quantique [106, 114, 73], sont intractables pour un ordinateur classique dans leur version exacte, mais deviennent tractables dès qu'un certain niveau d'approximation est autorisé [4, 136]. Cet exemple suggère qu'il soit nécessaire d'intégrer des éléments de correction d'erreur à une démonstration d'avantage quantique pour qu'elle soit significative. Tout calcul quantique peut-être rendu tolérant aux fautes d'après le théorème de seuil [3, 116, 86, 85], mais si le surcoût asymptotique en nombre de qubits est modéré (typiquement un facteur polylogarithmique), le préfacteur constant peut-être important (~ 1000 pour l'algorithme de factorisation [62]) et sa réalisation implique une complexité expérimentale additionnelle. La démonstration d'un avantage quantique dans un avenir proche nécessite donc d'adapter les techniques usuelles de correction d'erreur à la fois à l'algorithme envisagé, et à la plateforme physique considérée. C'est ce à quoi se propose le Chapitre 4 de ce manuscrit.

Dans ce chapitre, on étudie plus particulièrement le problème d'échantillonnage de circuits sIQP (*sparse Instantaneous Quantum Polynomial-Time* [33]), illus-

problème	surcoût spatial	profondeur	avantage	hypothèses
factorisation [63]	polylog	poly, adapt.	superpoly	RSA sûr
état graphe [109]	poly	$\mathcal{O}(1)$, adapt.	superpoly	PH $= \infty$ & ACH
sIQP [ce travail]	polylog	$\mathcal{O}(1)$, adapt.	superpoly	PH $= \infty$ & ACH
carré magique [27]	polylog	$\mathcal{O}(1)$	quasi-log	inconditionnel

Table 2: Candidats potentiels pour la démonstration d'un avantage quantique tolérant aux fautes. L'algorithme de factorisation offre un avantage superpolynomial, si l'on suppose que le protocole de cryptographie RSA est effectivement sûr, mais nécessite tout l'arsenal de la tolérance aux fautes. L'échantillonnage d'état graphe ou de circuit IQP permet également un avantage important, sous des hypothèses plus fortes (PH $= \infty$: non-effondrement de la hiérarchie polynomiale, i.e. une généralisation de P $\neq$ NP, et ACH : une conjecture de difficulté d'approximation des probabilités de sortie dans le cas moyen) et peut-être réalisé par un circuit adaptatif de profondeur constante. Finalement, le problème du carré magique offre un avantage inconditionnel via un circuit non-adaptatif de profondeur constante, mais cet avantage n'est que logarithmique.

tré par la Figure 3. Les circuits sIQP sont des circuits diagonaux dans la base computationnelle, générés aléatoirement à partir de portes logiques T et CS, et où les qubits sont initialisés et mesurés dans la base X. Puisque l'avantage quantique pour ce problème d'échantillonnage est perdue lorsqu'on considère un modèle de bruit réaliste [129], et il faut inclure à une démonstration d'avantage quantique basée sur cette construction une forme de correction d'erreur. Dans cette optique, les circuits sIQP présentent l'intérêt que les portes logiques utilisées ne forment pas un ensemble universel, c'est à dire ne permettant pas l'approximation de n'importe quel calcul quantique. La non-universalité de l'ensemble de porte logique utilisé est importante car elle implique que les circuits sIQP sortent du cadre du théorème d'Eastin-Knill [52], qui pose des restrictions importantes à toute architecture de calcul quantique universel tolérant aux fautes. Il reste à définir dans quelle mesure il est possible de simplifier l'adaptation tolérante aux fautes de ces circuits. Le résultat principal de ce chapitre est, étant donné un certain circuit sIQP D, la construction d'une famille de circuits de profondeur constante permettant une approximation arbitrairement précise de l'échantillonnage selon la distribution de probabilité p_D associée à D. Par rapport aux propositions comparables préexistantes mentionnées dans le tableau 2, le nombre total de qubits est multiplié par un facteur seulement polylogarithmique en l'inverse de la précision visée (par rapport à un facteur polynomial). La construction est basée sur un nouveau code correcteur d'erreur

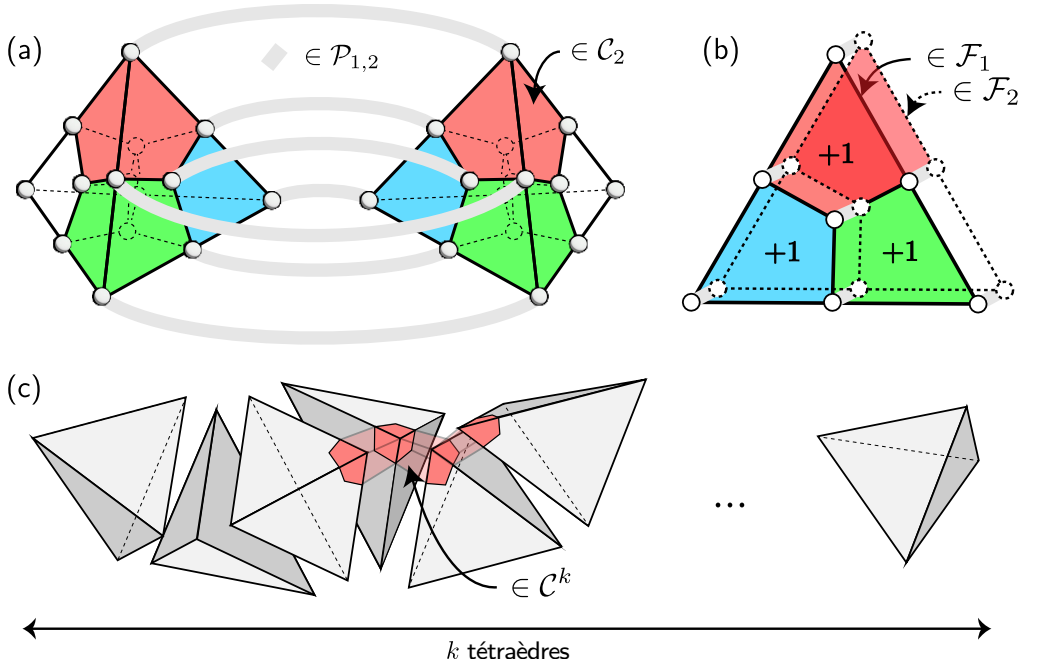

Figure 4: (a) Deux codes tétraédriques adjacents (ici avec $L = 3$) sont fusionnés en mesurant des pairs de qubits de $\mathcal{P}_{1,2}$ qui deviennent des stabilisateurs Z du nouveau code. La relation de commutation des stabilisateurs entre eux impose de fusionner les paires de stabilisateurs X (cellules) indiqués par la même couleur de part et d'autre. (b) Alors que la première mesure des nouveaux stabilisateurs Z est aléatoire, la préparation d'un état encodé impose de fixer leur valeur à $+1$. Les nouveaux stabilisateurs Z ne sont pas indépendants, puisque leur produit sur le support de deux plaquettes Z est égal à $+1$ en l'absence d'erreurs. (c) Fusionner des tétraédres supplémentaires permet de former une chaîne de tétraédre de longueur k. La configuration optimale est hélicoïdale dans le sens où elle minimise le nombre de stabilisateurs X fusionnés en un seul (de support dans $\mathcal{C}^k$), ce nombre est égal à quatre dans cet agencement.

quantique, susceptible d'être utile de manière indépendante, intitulé le *code tétrahelix*, dont les états $|+\rangle$ peuvent être préparés en profondeur constante et sur lequel des portes logiques encodées T et CS peuvent être appliquées simultanément et de manière transversale. Ces deux propriétés rendent la construction particulièrement adaptée aux plateformes d'atomes neutres, pour lesques les portes transversales entre blocs de code peuvent être implémentées nativement, et où les pertes d'atomes au cours du calcul tendent à privilégier des profondeurs de circuit réduite dans un premier temps.

La construction du code tétrahelix est illustrée par la Figure 4 et vise à combiner deux codes : le code couleur 3D pour lequel les portes T et CS sont

transversales, et le code de répétition pour sa capacité de parallélisation des portes de phases (diagonales en Z — dont T et CS). Le code tétrahelix est obtenu par fusion, en langage de chirurgie de réseaux, de codes couleurs 3D, plus particulièrement de codes tétraédriques, sur leur face. On montre que cette opération permet de conserver les propriétés de parallélisation et de transversalité désirées, et que les états de code $|+\rangle$ peuvent être préparés en profondeur constante ce qui donne directement une implémentation en profondeur constante de n'importe quel circuit sIQP. La mesure des qubits dans la base X à la fin du circuit suffit alors à approcher l'échantillonnage de la distribution de probabilité initiale en augmentant la distance du code, via le décodage du code tétrahelix.

Conclusion et perspectives. Encore récemment, une grande partie de la communauté physicienne ne croyait pas à la possibilité du calcul quantique. Pourtant, en l'espace de quelques années, l'état de l'art expérimental a évolué du contrôle de quelques dizaines de qubits bruités pour des circuits simples [8] aux premières démonstrations de correction d'erreurs [1, 131] via la mise en place de circuits complexes. Alors que les considérations récentes se limitaient généralement au régime régime NISQ (*Noisy Intermediate-Scale Quantum* [125]) sans correction d'erreur, ou alors directement au calcul quantique universel tolérant aux fautes, il apparaît désormais clairement que la communauté entre dans un régime intermédiaire, dans lequel les propositions théoriques ne peuvent pas encore s'abstraire des réalités expérimentales.

La question qui en découle naturellement est de savoir dans quelle mesure les approches théoriques proposées dans ce travail peuvent présenter un avantage expérimental. Pour les architectures reposant sur un décodeur local, il reste incertain de savoir s'il est possible d'intégrer une quantité minimale de calcul classique en parallèle du système quantique. Si l'utilisation de technologies classiques courantes (transistors, etc.) risque de générer trop de chaleur pour être effectivement intégrée, il est possible d'imaginer des architectures reposant sur des systèmes quantiques capables d'encoder et de manipuler de manière fiable un bit classique, à l'imagine du qubit de chat [112, 127]. En ce qui concerne la compilation de circuits sIQP sur des codes tétrahelix, bien que la proposition présentée semble qualitativement proche des capacités actuelles des plateformes à atomes neutres, sa réalisation expérimentale nécessiterait probablement encore un travail d'optimisation et de rationalisation. Une approche à court terme consisterait à tenter d'équilibrer les distances X et Z, mais une autre approche,

plus ambitieuse, serait de concevoir des codes à taux d'encodage plus élevé, tout en préservant les propriétés logiques de transversalité et de parallélisation des portes de phase considérées.

Il reste encore beaucoup à comprendre et à découvrir, et l'on espère que les avancées résumées ci-avant, et détaillées ci-après, contribueront à motiver la poursuite de l'exploration de la fascinante théorie du calcul quantique tolérant aux fautes, de ses premières démonstrations expérimentales jusqu'au calcul quantique universel. Ces progrès ne seront possibles sans une meilleure compréhension du monde physique et de ce qui distingue la mécanique quantique de la physique classique, sans doute la plus grande promesse de la théorie de l'information quantique.

Abstract

A quantum computation can be described by the evolution of a collection of quantum bits, or qubits, according to the laws of quantum mechanics, from an initial state encoding the input of a problem, to its solution represented by the final state. A machine capable of performing this task, a quantum computer, must be robust against errors caused by the inherent imperfections in the fabrication and control of such complex systems. This thesis examines the theory of fault-tolerant quantum computation, tailored to the physical constraints of today's leading platforms, most notably superconducting circuits and neutral-atom arrays.

A fault-tolerant quantum computation requires to correct errors in real time, at the cost of a dedicated architectural layer whose integration may introduce additional noise. A possible simplification uses a quantum memory model where information is encoded in a pair of configurations of a multi-qubit system, stabilized by the uniform application of a simple local rule fitting within the theory of cellular automata. We propose a short-term solution, the *shearing-rule* which protects a one-dimensional repetition code against errors with good performance but lacks a threshold. This shortcoming is addressed by adding small local classical memories, used to define the *signal-rules* decoders, where error correction relies on exchanging binary signals. Combined with biased-noise qubits, such as cat qubits, these local decoders define a fully local one-dimensional quantum memory.

The final part of the manuscript addresses the advantage of quantum machines over their classical counterparts. We emphasize that any such demonstration must account for the presence of noise to be meaningful — for example, while exact sampling from the output distribution of circuits drawn from certain families is believed to be hard for classical computers, the task may become tractable if one allows for an approximate solution that reflects the presence of noise. In this context, we propose a constant-depth quantum circuit that solves a sampling problem in a fault-tolerant manner. Because our approach is particularly well-suited to the atom rearrangement capabilities of neutral-atom array platforms, we hope it will contribute to a future experimental demonstration of a fault-tolerant quantum advantage.

Contents

List of Figures

List of Tables

List of Abbreviations

k-LTR k-Local Toom's Rule. 68

ASR Asymmetric Signal-Rule. 89, 93, 95–99, 101, 102, 111, 120–122

CSS Calderbank-Steane-Shor (code). xiv, 25, 129, 133, 139–141

GKL Gács, Kurdyumov, and Levin (automaton). 36

GTR Global Toom's Rule. 68

IQP Instantaneous Quantum Polynomial-time (circuit). xvi, xxviii, 55–60, 62, 125–130, 132, 133, 135, 140, 147, 148

LDPC Low-Density Parity-Check (code). 27, 38, 146

ML Maximum-Likelihood (decoder). 20, 51

MWPM Minimum-Weight Perfect Matching (decoder). xiii, xiv, 22, 23, 51, 53

QEC Quantum Error Correction. 12

sIQP sparse Instantaneous Quantum Polynomial-time (circuit). 127

SSR Symmetric Signal-Rule. 89, 90, 97, 98, 101

TLV Two-Line Voting (automaton). 80

Chapter 1

Introduction

In this introductory chapter, we review the original motivations for quantum computation, along with the theory of quantum error correction and fault-tolerant quantum computation, before highlighting the challenges faced by modern physical platforms in achieving fault tolerance.

Contents

1.1 Information is physical

Whether communicated by controlled smoke emission, written with droplets of inks on the pages of a book, or encoded in the pair of states of a semiconductor device, information does not exist independently from a physical carrier. '*Information is physical*', said the German-American physicist Rolf Landauer [97], and its manipulation must necessarily obey the laws of the physical world. Information and its manipulation however lie at the foundation of modern human society, built on the unparalled progress of *computer science* across the twentieth century and into the modern era. The fundamental idea of computation is that, if information can be represented in the state of a physical device, then its manipulation can be performed by a machine. Tools for calculating particular numerical quantities, such as the abacus, have existed since the antiquity, but it is the mastering of micro-electronics, with perhaps one of the most important discoveries of the past century, the transistor, that has driven the rise of digital computer science. Despite its size of a dozen nanometers, roughly a hundred atoms, from the point of view of computation a transistor can be understood as a switch that is either 'off' or 'on', something that can be described by the same laws of physics that we experience in our everyday lives, the ones of Newtonian mechanics and Maxwell's electromagnetism, commonly refered to as the laws of *classical physics*.

But if information is physical, then, necessarily, its manipulation, and by extension the ability to solve problems, depend on the relevant set of physical laws. In particular, the difficulty of a problem should depend on the laws of the physical world from which one looks at it. We however know, since the begining of the twentieth century, that the laws of classical physics do not suffice to describe nature, calling for more fundamental theories. A faithful description of the movement of planets and justification for the existence of black holes need the laws of *general relativity*, while the behavior of electrons, atoms and photons follow the laws of *quantum mechanics*, where physical properties are quantized to discrete values attached to particular states. This was the idea of Max Planck and Albert Einstein that suggested that light is made of discrete quanta of energy, or photons, each carrying an energy $E = \hbar\nu$, directly proportional to the light frequency [54]. The field of quantum mechanics has kept evolving since then, with foundational concepts such as superposition, the uncertainty principle, and quantum entanglement. The conceptual break

with classical physics may have culminated with John Bell's Theorem '*On the Einstein-Podolsky-Rosen paradox*' [12] that formulated a statistical test able to certify that quantum entanglement yields correlations incompatible with classical theories, a test that was later experimentally successively verified by John Clauser, Alain Aspect and Anton Zeilinger [57, 9, 153]. Those experiments are the evidence that information does not follow the same rules in the quantum world as in the classical one, and that there could be more to what can be efficiently computed in our world than what can be efficiently computed on classical computers, or stated differently, that problems known to be hard from a classical point of view could become easy when given access to a quantum machine: this is the promise of quantum computation.

The idea of quantum computation dates back to the early 80s, when digital computers were becoming increasingly powerful, but on which physicists faced an exponential increase in overhead when simulating quantum mechanics. At that time, digital computers were generally believed to be an efficient universal computing machine, meaning that any computing machine that could be fabricated, can also be efficiently simulated on a digital computer. But if simulation of physics consists of imitating it, and if a computer follows the same laws of physics that the physical system it aims to imitate, then it should be possible to make the number of elementary elements of the computer proportional to the space-time volume of the system it simulates. Those were the rules of simulation wished for by Richard Feynman, who, assessing that the physical world is quantum mechanical, famously conjectured that hardware based on quantum phenomena should be more efficient to simulate physical systems. The achievement of quantum computer science that first drew attention to the field, was however not targeted at simulating physical systems, but rather at solving a seemingly very specific problem, nevertheless at the core of most secure data transmission protocols accross the world, the problem of writing a number as a product of prime factors. Peter Shor formulated his factoring algorithm in 1995 [140], quickly followed by Lov Grover in 1996 with a proposal for unstructured search, in time proportional to the square root of the number of candidates [69]. Since then, the field of quantum algorithms has continued to expand, with several notable contributions. In 2008, Harrow, A., Hassidim, A., and Lloyd, S. proposed an algorithm that provides an exponential speedup for obtaining certain information about the solution to a system of linear equations [75]. More recently, in 2024, Jordan et al. introduced an algorithm to find good approxi-

mate solutions to hard optimization problems [82]. The challenge of quantum computation is however not only the one of imagining new quantum algorithms, it is also the immense one of achieving a reliable control of systems operating in the quantum regime, yielding a question that remains largely unanswered. Will there actually be a quantum machine reliable enough and of a sufficient scale to implement those existing proposals and the ones to come, and achieve the biggest promise of quantum computers that may remain their original one: the promise of an unprecedented understanding of quantum mechanical systems, and, by extension, of nature?

1.2 Fundamentals of quantum computation

1.2.1 Quantum bits and measurements

The fundamental building block of quantum computation is the *quantum bit*, or *qubit* in short. In contrast to the *classical bit*, after which it is named and that takes the value 0 or 1, the qubit is a quantum two-level system, meaning that the information is encoded in the space generated from two quantum states. In practice, any physical two-level system that can be operated in the quantum regime can serve as a qubit, e.g. the spin of an electron that can be 'up' or 'down', the 'ground' and 'excited' states of an atom or of the electric field in a cavity, etc. To abstract away of the physical implementation of the system, we will denote those two *computational basis states* by $|0\rangle$ and $|1\rangle$.

Following the laws of quantum mechanics, the state of a qubit is in general described by a *superposition* of the computational states, that we formally write as a linear combination of vectors $|0\rangle$ and $|1\rangle$,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1.1}$$

where the coefficients α and β are two complex numbers that satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Stated differently, the state of a qubit is a vector in a two-dimensional *Hilbert space* $\mathcal{H}$ over the complex numbers. The Hilbert space $\mathcal{H}$ is endowed with the usual Hermitian inner-product between states $|\psi\rangle$ and $|\phi\rangle$ denoted by $\langle\phi|\psi\rangle \in \mathbb{C}$, so that the relation between coefficients α and β fixes the norm to $|\,\langle\psi|\psi\rangle\,|^2 = 1$. Because of this, we may rewrite (1.1) up to an unobservable global phase as

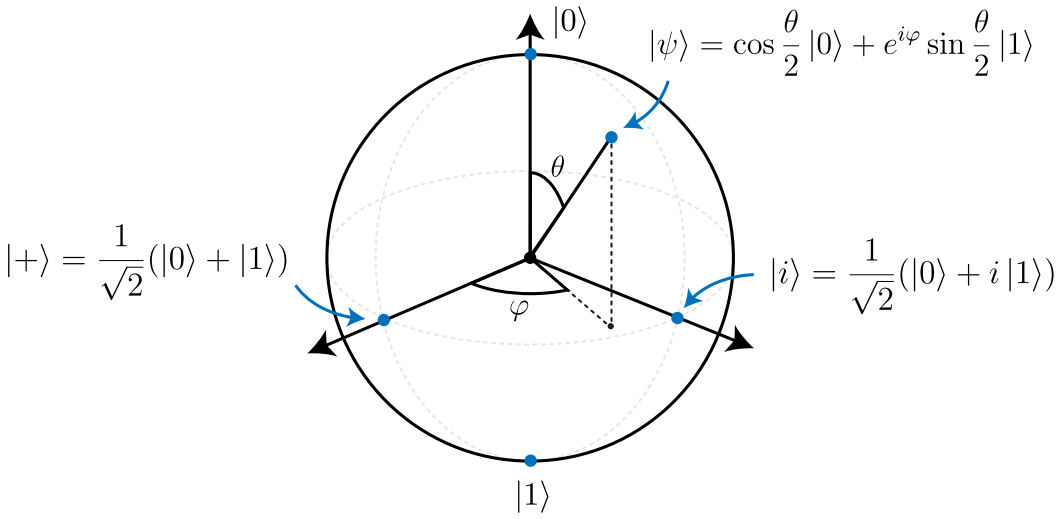

Figure 1.1: Bloch sphere representation of a qubit.

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\,, \tag{1.2}$$

where θ and φ are real numbers. The state of a qubit can then be conveniently visualized as a point on the so-called *Bloch sphere*, as shown in Figure 1.1.

While there are an infinite number of points on the unit sphere, that is to say an infinite number of possible states, a qubit does not store an infinite amount of information. To understand this, one must describe the behavior of a qubit when observed. When we measure a qubit to determine in which computational states it lies, we only get a most one bit of information: the outcome 0 with probability $|\alpha|^2$, or 1 with probability $|\beta|^2$. Crucially, according to one of the postulates of quantum mechanics, the observation *collapses* the superposition to the state that is consistent with the measurement outcome: the qubit is left in state $|0\rangle$ if the measurement outcome is 0, or state 1 if the measurement outcome is 1. In both cases, most of the information contained in the complex numbers α and β is lost.

Observables and measurements. In quantum mechanics, a physical quantity that can be measured is an *observable*, represented by a Hermitian operator O acting on a Hilbert space $\mathcal{H}$, meaning that $O^\dagger = O$, where $\dagger$ denotes the conjugate transpose.

Since O is Hermitian, by the spectral theorem it has a real spectrum and

can be diagonalized with an orthonormal basis of eigenvectors $\{|\lambda_i\rangle\}$ with associated projectors $P_i = |\lambda_i\rangle\langle\lambda_i|$. The eigenvalues λ_i correspond to the possible outcomes of measuring the observable. The *projective measurement* of O on a pure state $|\psi\rangle$ follows the *Born rule*: the outcome is λ_i and the state is projected onto the eigenstate $|\lambda_i\rangle$, with probability

$$\mathbb{P}(\text{outcome } \lambda_i) = \langle\psi| P_i |\psi\rangle , \qquad |\lambda_i\rangle = \frac{P_i |\psi\rangle}{\| P_i |\psi\rangle \|}, \tag{1.3}$$

In particular, measuring a qubit in the computational basis corresponds to taking the Pauli Z operator as the observable.

A meaningful computation must involve multiple qubits, just as a classical computer operates on multiple bits. The drastic difference here will be the evolution of the description of the system with its number of constituent. For two classical bits, there are only four possible states: 00, 01, 10 and 11. For two qubits however, similarly to the one-qubit case, the system will be in a superposition of the corresponding four computational basis states

$$|\psi\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle , \tag{1.4}$$

so that the quantum state is represented by a vector in a Hilbert space of dimension four, formally the tensor product of two single-qubit Hilbert space so that for example $|00\rangle := |0\rangle \otimes |0\rangle$. Once again, when measuring the two qubits in the computational basis, one obtains the word $w \in \{00, 01, 10, 11\}$ with probability $|\alpha_w|^2$, in which case the system is left in state $|w\rangle$. Already at this stage, some intrinsic property of quantum mechanics begins to manifest: consider the following 2-qubit quantum state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{1.5}$$

It is clear that this state, famously known as the *Bell pair*, cannot be written as a *product state*, that is to say it cannot be written as $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_i\rangle$ describes the state of qubit $i \in \{1, 2\}$. This means that if one measures the first qubit and finds it to be 0, the second qubit will definitely be 0 as well. Symmetrically, if the first is 1, the second is also 1. Importantly, this correlation does not correspond to a lack of information on the system: neither

qubit has a definite value before measurement and we say that the two qubits are *entangled*. Note that in general, we may consider a system of N qubits, in which case the Hilbert space is of dimension 2^N with the set of quantum states $\{|w\rangle\,,\, w \in \{0,1\}^N\}$ as an orthonormal basis, where $|w_1 w_2 \dots w_N\rangle = \otimes_{i=1}^{N} |w_i\rangle$. States described until now are also called *pure states*, because they exactly and completely describes the state of the system.

Because of the probabilistic and projective nature of measurements, it is often useful to describe the state of the system as a statistical mixture of pure states. For instance, we may only know the state of the system up to a probability distribution over a set of possible quantum states — that is, a set $\{(p_i, |\psi_i\rangle)\}$, where p_i denotes the probability of the system being in state $|\psi_i\rangle$, hence with the condition $\sum_i p_i = 1$. This type of state is refered to as a *mixed state*, which can be described in the *density matrix* formalism by the matrix ρ defined as a weighted sum of pure states $|\psi_i\rangle\langle\psi_i|$,

$$\rho = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \,, \tag{1.6}$$

where $|\psi\rangle\langle\phi|$ denotes the outer-product of $|\psi\rangle$ and $|\phi\rangle$ in $\mathcal{H}$. Note that the normalization condition on probabilities can simply be translated here as $\mathrm{Tr}(\rho) = 1$, and because probabilities are necessarily positive the density operator eigenvalues can be shown to be non-negative. A measurement can then be described using the same notation as in the pure state setting: the outcome is λ_i and the post-measurement state is ρ_i, occurring with probability

$$\mathbb{P}(\text{outcome } \lambda_i) = \mathrm{Tr}(P_i \rho P_i^\dagger), \qquad \rho_i = \frac{P_i \rho P_i}{\mathrm{Tr}(P_i \rho P_i^\dagger)}. \tag{1.7}$$

Note that more general notions of measurement exist, but these will not be considered in the scope of this thesis. Mixed states can correspond to the state after measurement in the event where one has not read the measurement outcome, or alternatively, and importantly, to the remaining part of an entangled state after one has discarded the other part. For example, discarding the second qubit of a Bell pair results in the state

$$\rho = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|). \tag{1.8}$$

Formally, the operation of discarding a part of the system corresponds to taking

the partial trace of the density matrix. The associated effect — the decay of off-diagonal elements in the partial trace of the joint system, for any basis of the combined system, corresponding to the loss of quantum superposition — is commonly referred to as decoherence.

1.2.2 Circuit model of quantum computation

Because we aim to perform a computation, we would like to manipulate the quantum state. Since in quantum mechanics, an isolated system evolves according to the Schrödinger equation, the evolution will correspond to the application of unitary operators.

Time evolution. The time evolution of a quantum state $|\psi(t)\rangle$ isolated from the environment, is governed by the Schrödinger equation,

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle\,, \tag{1.9}$$

where H is the Hamiltonian operator, representing the total energy of the system and $\hbar$ is the reduced Planck constant. The formal solution is

$$|\psi(t)\rangle = e^{-iHt/\hbar}|\psi(0)\rangle\,, \tag{1.10}$$

describing a unitary evolution generated by H. For a system of N qubits, the evolution is an element of $\mathrm{SU}(2^N)$ up to a global phase, where $\mathrm{SU}(2^N)$ is the group of $2^N \times 2^N$ unitary matrices with determinant 1.

By definition, a unitary operator U satisfies $UU^\dagger = I$. An equivalent property of unitaries is that it preserves the norm, so that this type of evolution preserves the normalization condition. A single-qubit unitary, or single-qubit gate, corresponds to a 2×2 unitary matrix U, examples of which are the four 2×2 *Pauli gates*

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.11}$$

The Pauli X gate acts as a quantum equivalent of the NOT gate, mapping

$|0\rangle = \binom{1}{0}$ to $|1\rangle = \binom{0}{1}$ and vice versa, so that in general, by linearity, $X(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|1\rangle + \beta\,|0\rangle$. We can define the N-qubit Pauli group.

Pauli group. The Pauli group over N qubits $\mathcal{P}_N$, defined by all N-fold tensor products of the Pauli matrices X, Y and Z and the identity I, multiplied by an overall phase $\{\pm 1, \pm i\}$:

$$\mathcal{P}_N = \{\alpha\, P_1 \otimes P_2 \otimes \cdots \otimes P_N \mid \alpha \in \{\pm 1, \pm i\},\ P_j \in \{I, X, Y, Z\}\}\,, \quad (1.12)$$

where the phase factors ensure that the set is closed upon multiplication so that it is a group.

Observe that if the qubit is initialized in the computational basis, we cannot create a superposition in the same basis by applying only Pauli gates. To achieve this, we need for instance the so-called *Hadamard gate* that maps $|0\rangle$ to $|+\rangle$, with matrix representation

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (1.13)$$

Note that the Hadamard gate is denoted H similarly as the Hamiltonian, but it should be clear from the context which one is being referred to.

Meaningful computation, however, requires qubits to interact with one another. This is done by gates acting on at least two qubits. A 2-qubit gate is defined by a 4×4 unitary matrix, or equivalently, by its action on the four computational basis states. For example, the CNOT gate (sometimes noted CX) flips the second qubit conditionned on the fact that the first is set to 1. In equations, $\mathrm{C}X\,|00\rangle = |00\rangle\,,\ \mathrm{C}X\,|01\rangle = |01\rangle\,,\ \mathrm{C}X\,|10\rangle = |11\rangle\,,\ \mathrm{C}X\,|11\rangle = |10\rangle\,.$

Recall that a system of N qubits is described in an Hilbert space of dimension 2^N, with the set of quantum states $\{|w\rangle\,,\ w \in \{0,1\}^N\}$ as an orthonormal basis. The fact that the dimension of the Hilbert space grows exponentially with the number of individual components lies at the heart of both the difficulty of simulating quantum mechanics on a classical computer, and the promise of the enormous computational power of quantum computers. A quantum computation can, without loss of generality, be described as the application of a $2^N \times 2^N$ unitary matrix to an N-qubit quantum state initialized in $\left|0^N\right\rangle$, followed by a

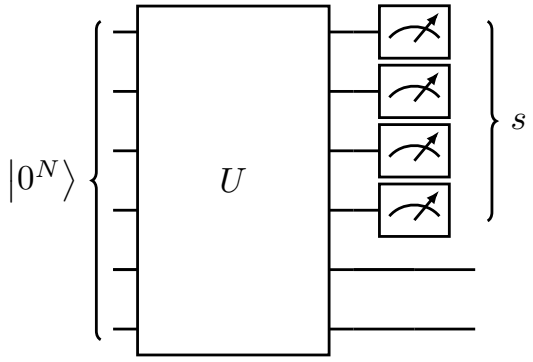

Figure 1.2: Circuit representation of a quantum computation on N qubits, with output a bitstring s obtained from measurement outcomes. Qubits are represented by wires on which is applied a unitary operator U, before measurement of a subset of qubit at the end.

measurement of a subset of the qubits in the computational basis. A quantum computation is illustrated in Figure 1.2 in the so-called *quantum circuit* model, where qubits are represented by wires, on which unitaries are applied with time going to the right.

It is however not reasonable to expect a realistic physical system to be able to implement any arbitrary unitary U in a single step, so that we would prefer to decompose U into a sequence of elementary few qubit unitaries, that we call *quantum logic gates*, similarly to classical computation where an arbitrary logical operation can be performed from a finite set of logic gates, e.g. AND, OR and NOT. The problem is however more complicated in the quantum case since the set of unitaries is continuous hence infinite. Fortunately, the problem can be simplified by realizing that arbitrary single-qubit gates and CNOT gates taken together are universal, meaning that any unitary U can be synthesized into a circuit using only these gates only. Because there are an uncountable number of possible unitaries, while a finite gate set only generates a countable number of unitaries, the synthesis of an arbitrary U from a finite set of gates can only be achieved in an approximate sense. The *Solovay–Kitaev Theorem* ensures that any $U \in \mathrm{SU}(2)$ can be approximated to arbitrary accuracy using a finite set of elementary single-qubit gates, with the number of gates scaling only polylogarithmically with the inverse of the error.

Theorem 1 (Solovay-Kitaev [84])**.** *Let $\mathcal{G} \subset \mathrm{SU}(2)$ be a finite gate set that is closed under inversion and such that the group it generates is dense in* $\mathrm{SU}(2)$*, and some precision target $\delta > 0$. There exists a constant $c > 0$ such that for*

any unitary $U \in \mathrm{SU}(2)$, there exists a sequence $V = V_1 \cdots V_m$ of gates from $\mathcal{G}$, of length $m = \mathcal{O}(\log^c(1/\delta))$ such that $\|U - V\| \leq \delta$ in operator norm.

Importantly, there exists an efficient algorithm to find such a sequence. A gate set that satisfies those conditions is said to be universal. Universal gate sets for single-qubit gates include $\mathcal{G} = \{H, T\}$ where the T gate is defined as the fourth root of the Pauli Z gate, that it to say it applies a $\pi/4$ phase to the $|1\rangle$ state. This directly implies that the gate set $\{H, T, \mathrm{CNOT}\}$ is universal for quantum computation. Note that the CNOT and T gates can be replaced by the *Toffoli gate*, or controlled-controlled-NOT (sometimes noted CCX) that flips the state of the third qubit if and only if the first two qubits are in the $|1\rangle$ state. Interestingly, because the Toffoli gate is already universal for classical reversible computation, this shows that one only needs to add the Hadamard gate to make a 'classical' set of gates universal for quantum computation. The Toffoli gate and the T gate are examples of non-Clifford gates, a notion that we define now and that will prove useful later on.

Clifford hierarchy. The Clifford hierarchy is a recursive hierarchy of unitary operators that plays a central role in fault-tolerant quantum computation. It is defined recursively as follows:

- The first level of the hierarchy is the Pauli group: $\mathcal{C}_1 := \mathcal{P}_N$.
- The k-th level, for $k \geq 2$, is defined as

$$\mathcal{C}_k := \left\{ U \;\middle|\; UPU^\dagger \in \mathcal{C}_{k-1} \text{ for all } P \in \mathcal{P}_N \right\}. \tag{1.14}$$

Operators in level $k = 2$ (resp $k \geq 3$) are said to be Clifford (resp. non-Clifford). The presence of non-Clifford gates is a necessary condition for a quantum computation to be intractable for classical simulation, and is required to achieve universal quantum computation.

The challenge of quantum computation however not only lies in understanding the intricacies of quantum information and designing powerful quantum algorithms, but also in engineering macroscopic systems that maintain quantum properties. The peculiarity of quantum physics is that, although it accurately describes the laws governing the microscopic world from which our macroscopic

reality emerges, quantum coherences are lost in the process, causing us to experience an effectively classical world — where, famously, a cat can only be dead or alive, but never both. For the same reason, qubits are inherently fragile and highly susceptible to errors due to decoherence, imperfect gate operations, and environmental noise — preventing meaningful quantum computation that involves coherent superpositions and entanglement. The fact that it is theoretically possible to detect and correct errors without directly measuring and collapsing the quantum state is captured by the theory of Quantum Error Correction (QEC), one of the defining miracles of quantum computation.

1.3 Quantum computation in a noisy world

Quantum computers function in an imperfect, noisy world, where quantum logic gates are subject to errors. To give a rough estimate, depending on the physical platform, typical 2-qubits gates are erroneous with probability $0.1\% - 1\%$ [1]. While this may seem to be a small quantity, this has to be compared to the number of gates within a meaningful quantum computation. This is not encouraging: state-of-the-art constructions estimate the number of Toffoli gates needed to factor 2048-bit RSA to more than a billion [42]. Such a computation would demand gates that are orders of magnitude more reliable to obtain a meaningful result: it seems hopeless to bridge this gap by hardware improvements alone, and other ideas are needed.

1.3.1 A classical intuition

The problem of protecting information from errors long predates quantum computation. Even in classical settings, communication is often imperfect. Imagine two fictional characters, Alice and Bob, attempting to exchange messages over a noisy channel represented in Figure 1.3. We will abstract away the exact physical nature of the channel, but for instance, suppose Alice sends a 0 or a 1 via a telegraph, each bit represented by a particular electronic pulse propagating along the wire. The system transmits the message correctly with probability $1 - \varepsilon$, but with probability ε the signal is perturbed, say due to corrosion affecting the wire, so that a signal corresponding to a 0 is reshaped to correspond to a 1, and vice versa. This type of channel is known as the *binary symmetric channel*.

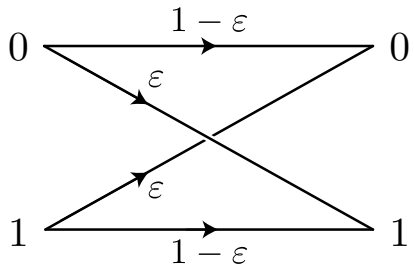

Figure 1.3: Binary symmetric channel.

A simple but insightful way for Alice to increase the probability that the information is succesfully received by Bob, is to repeat the message, for example sending three bits instead of one. More formally, we say that Alice encodes her initial message in a so-called 3-bit *repetition code*: instead of sending a 1, Alice sends the bitstring 111, and instead of sending a 0, she sends 000:

$$0 \to 000, \qquad 1 \to 111. \tag{1.15}$$

Bob can then simply perform a *majority vote* on the three bits of information he receives to infer the most likely logical value that was initially sent. In particular, for Bob to misidentify the logical value — e.g., interpret a logical 1 when Alice actually sent 0 (that is, 000) — at least two errors must occur, so that the received bits are 110, 101, 011, or 111, all of which he would decode as 111. If error events arise independently on each bit, we can easily compute the probability of this event to be $3\varepsilon^2(1-\varepsilon) + \varepsilon^3 = \mathcal{O}(\varepsilon^2)$. This is less than the initial error ε whenever $\varepsilon < 1/2$, in which case the error probability is suppressed quadratically. The reduction in error probability corresponds to the fact that more than half of the bits must be flipped for a logical error to occur — in the case of a 3-bit repetition code, this means at least $(3+1)/2 = 2$ bit flips. A straightforward generalization is the n-bits repetition code, where the *logical error* probability can be shown to be exponentially suppressed with the code size, more precisely with $(n+1)/2$, allowing one to achieve arbitrarily reliable communication at relatively little cost, more precisely only logarithmic in the inverse desired error probability.

Perhaps not surprisingly, simply repeating the initial information is not the most efficient way of encoding information. Nevertheless, the concept of *encoding* initial logical information into the *code space* of some *error correcting code*— i.e. some subspace of a larger space—which *decoding* maps back to the original

information, has far-reaching implications. Its generalization to more efficient codes from more involved mathematical structures has led to the tremendous success of the theory of *classical error correction*, used across a wide range of classical communication systems — e.g., *Hamming codes* in computer memory, *Reed–Solomon codes* in CDs and DVDs, *convolutional codes* in satellite communication, and *low-density parity-check (LDPC) codes* in modern data transmission standards.

Despite the achievements of classical error correction, extending these ideas to what would become the field of quantum error correction is not straightforward. This is due to fundamental differences between the manipulation of classical and quantum information, so that new ideas are required. At first sight, there appear to be three main challenges in protecting quantum information from noise. First, quantum mechanics forbids copying an arbitrary quantum state, preventing naive encoding. Second, unlike in classical computation, noise can act continuously on the quantum state. Third, measurement generally destroys the quantum state under observation, necessitating a new decoding scheme. We discuss these three points in more detail in the following.

Encoding. As in the classical scenario with Alice and Bob, we would like to introduce redundancy into our quantum information to protect it from noise; this is however not so straightforward at first sight. Simply copying the initial logical information by encoding any logical qubit $|\psi\rangle$ into a product state of the form $|\psi\rangle \otimes \cdots \otimes |\psi\rangle$ is not possible, due to the famous *no-cloning theorem* that states that given some state $|\phi\rangle$, no single unitary produces the mapping

$$U(|\psi\rangle \otimes |\phi\rangle) = |\psi\rangle \otimes |\psi\rangle\,, \tag{1.16}$$

for all states $|\psi\rangle$. This can easily be proven by contradiction: assume such a unitary U exists: then let $|\psi_1\rangle$ and $|\psi_2\rangle$ be two orthogonal states, applying U to the product state $\frac{1}{\sqrt{2}}(|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi\rangle$ leads to a contradiction,

$$U[\tfrac{1}{\sqrt{2}}(|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi\rangle] = \tfrac{1}{\sqrt{2}}(U[|\psi_1\rangle \otimes |\phi\rangle] + U[|\psi_2\rangle \otimes |\phi\rangle]) \tag{1.17}$$

$$= \tfrac{1}{\sqrt{2}}(|\psi_1\rangle \otimes |\psi_1\rangle + |\psi_2\rangle \otimes |\psi_2\rangle) \tag{1.18}$$

$$\neq \tfrac{1}{\sqrt{2}}(|\psi_1\rangle + |\psi_2\rangle) \otimes \tfrac{1}{\sqrt{2}}(|\psi_1\rangle + |\psi_2\rangle). \tag{1.19}$$

Here the first line is obtained by linearity and the last one by the contradictory hypothesis. An important subtlety, thankfully, is that the no-cloning theorem rules out the possibility of existence of an unitarying copying any input state $|\psi\rangle$, but not the one of an unitary copying a particular state. In this context, a logical qubit may be encoded into the larger Hilbert space spanned by n physical qubits $\mathcal{H}_{\text{physical}} = (\mathbb{C}^2)^{\otimes n}$, with encoding map the isometry

$$\mathcal{E} : \mathcal{H}_{\text{logical}} \to \mathcal{H}_{\text{physical}}, \tag{1.20}$$

that corresponds to an unitary evolution U in the larger dimensional physical Hilbert space

$$\mathcal{E}(|\psi\rangle) = U(|\psi\rangle \left|0^{n-1}\right\rangle). \tag{1.21}$$

Importantly, we will see that this can be done efficiently for good choices of encoding. Note that in the following of the thesis, we will refer to encoded states/operators with an overline so that for example $\left|\overline{\psi}\right\rangle = \mathcal{E}(|\psi\rangle)$.

Quantum noise. The encoding must be such that it protects quantum information from noise. The noise processes affecting a quantum computer are however highly more complex than those typically considered in the field of classical communication, in which case errors are typically restricted to *bit-flips* where a 0 is flipped to a 1 and vice versa.

First, quantum bits are never perfect two-level systems, and the quantum state can *leak* into other states, such as $|2\rangle$, which lie outside the computational subspace spanned by $|0\rangle$ and $|1\rangle$. For example, in typical superconducting qubits like the transmon, quantum information is stored in the two lowest energy levels of an anharmonic oscillator, but there remains a risk that the state starts to have non-zero composants in higher energy levels. These *leakage errors* are especially problematic because, if they go undetected, they can accumulate over time and significantly disrupt the computation.

Even without leakage errors, there is no specific reason for noise acting on a qubit, say in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, to be limited to bit-flip noise, sometimes refered to as X-type noise because it corresponds to applying a X-type Pauli gate,

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle. \tag{1.22}$$

For example, noise could take the form of an unwanted Z-type Pauli gate,

usually refered to as a *phase-flip*,

$$Z(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|0\rangle - \beta\,|1\rangle\,. \tag{1.23}$$

Even worse, noise can in principle take the form of all possible evolution described until now, including non-unitary evolution like the ones introduced to describe projective measurements, see the frame below, or unitary evolution close from the identity. For example, because the Hilbert space is continuous, noise could correspond to a rotation around the Z axis by a small angle θ,

$$Z(\theta)(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|0\rangle + e^{i\theta}\beta\,|1\rangle\,, \tag{1.24}$$

where $Z(\theta) = \exp\big(-i\frac{\theta}{2}Z\big)$, and hence $Z(\pi) = Z$ up to a global phase. In general continuous noise processes can be particularly detrimental to computation, because below some detection level, they can lead to errors that accumulate in space and time, until it is too late to correct them.

Kraus formalism. Consider some quantum system of interest representing e.g. the set of qubits, described by a quantum state in a Hilbert space of dimension d. Quantum noise arises from undesired coupling between the system and its environment. In this case, the joint evolution of the system and environment can be described by a unitary operator acting on the combined system 'system + environment'.

The associated dynamics induced on the reduced system is obtained from tracing out the environment and is in general non-unitary, it can be described in the Kraus formalism by the channel

$$\mathcal{N}(\rho) = \sum_{k=1}^{M} E_k \rho E_k^\dagger. \tag{1.25}$$

Here $M \leq d^2$ and the set of operators $\{E_k\}$, known as *Kraus operators*, satisfies a completeness relation

$$\sum_{k=1}^{M} E_k^\dagger E_k = I, \tag{1.26}$$

so that $\mathcal{N}$ is trace-preserving. Examples of typical quantum noise channel

include depolarizing noise

$$\mathcal{N}(\rho) = (1-\varepsilon)\rho + \frac{\varepsilon}{3}(X\rho X + Y\rho Y + Z\rho Z), \tag{1.27}$$

and bit-flip noise

$$\mathcal{N}(\rho) = (1-\varepsilon)\rho + \varepsilon X\rho X, \tag{1.28}$$

both parametrized by some error rate $\varepsilon > 0$. The noise process can be understood, for example, in the case of bit-flip noise as flipping the qubit with probability ε and leaving it unchanged otherwise.

The wide variety of possible error types makes the choice of encoding more complex than in the classical case. Surprisingly, however, we will see that it is possible to transform the inherently analog problem of quantum noise into a digital one, much like in classical computation. This discretization of errors results in a remarkable insensitivity to small perturbations, thereby enabling the correction of arbitrary errors.

Decoding. We also need to devise a scheme that corrects errors on a given encoded state $|\overline{\psi}\rangle$ so that they do not accumulate in time. This ability seems to inevitably rely on observing the system in some way. While this poses no conceptual difficulty for classical systems, where observation does not disturb the state, the situation is fundamentally different in quantum mechanics.

By linearity, the encoded state $|\overline{\psi}\rangle$ will be a superposition of encoded computational basis states; for example, $|\overline{\psi}\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$ for some complex coefficients α and β. We now aim to observe the state to detect potential errors, but the process must leave undisturbed states invariant, as it should not interfere with an ongoing quantum computation on encoded states. The problem is that, for any non-trivial task and non-trivial encoding, the error-free state $|\overline{\psi}\rangle$ will in general also be a superposition of n-qubit computational basis states. This means that individual measurements on all physical qubits will collapse the state, losing most of the encoded quantum information.

The collapse of the state occurs because, just as in the single-qubit case, measuring all qubits individually reveals information on the encoded state, something we only want to do at the end of the computation. Fortunately, we will see that it is possible to detect and correct errors without revealing information

about the logical state, by exploiting the symmetries of the code space, thus preserving quantum superpositions.

1.3.2 Quantum error correction

The solution to protect information from errors and decoherence lies in the theory of *quantum error-correction* pioneered by Peter Shor[1], who discovered how quantum information can be stored within the highly entangled states of nine qubits [139], formulating the first *quantum error correcting code*. A quantum error correcting code is formally defined as a subspace $\mathcal{C}$ of a larger Hilbert space $\mathcal{H}$ that enables to encode quantum information in a redundant way by using the additional degrees of freedom available within the larger space. Since we are interested in quantum computation over qubits, we will focus on codes that encode k *logical qubits* in a subspace of dimension 2^k of a larger Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$ of n *physical qubits*.

Starting from general assumptions, let us suppose that the noise is described by a noise channel $\mathcal{N}$, and that the error-correction procedure corresponds to a trace-preserving quantum operation $\mathcal{R}$ (defined similarly as the noise process in the Kraus formalism). For error-correction to be successful, we require that for any state $\rho = |\psi\rangle\langle\psi|$ with $|\psi\rangle \in \mathcal{C}$

$$(\mathcal{R} \circ \mathcal{N})(\rho) = \rho. \tag{1.29}$$

This may seem to be an overwhelming task, in the sense that the set of possible errors, that is the set of possible Kraus operators, is infinite, so that one would need to be able to correct an infinite number of different error types. Fortunately, the linearity of quantum mechanics allow us to reduce the problem to the one of correcting a finite set of errors.

Theorem 2 (Discretization of errors [116])**.** *Suppose $\mathcal{C}$ is a quantum code and $\mathcal{R}$ is the error-correction operation to recover from a noise process $\mathcal{N}$ with Kraus operators $\{E_i\}$. Suppose $\mathcal{M}$ is a quantum operation with operation elements $\{F_i\}$ which are linear combinations of the E_i. Then the error-correction operation $\mathcal{R}$ also corrects for the effects of the noise process $\mathcal{M}$.*

In particular, if one is considering arbitrary errors on a subset of qubits, it is sufficient to prove that a given recovery operation corrects for a basis of uni-

[1] and further developed by Andrew Steane and others.

taries on said qubits, for example the associated Pauli group. This means that, without loss of generality, the task of correcting for arbitrary errors[2] reduces to the task of correcting arbitrary Pauli errors.

A particularly important and powerful class of quantum error-correcting codes is the one of *stabilizer codes* develeloped by Daniel Gottesman in his PhD thesis [65]. A stabilizer code $\mathcal{C}$ is defined from an abelian subgroup $\mathcal{S} \subset \mathcal{P}_n$, called the *stabilizer group*, and where in addition we require that $-I \notin \mathcal{S}$. The elements of $\mathcal{S}$ are stabilizers and the code space $\mathcal{C}$ is defined by states that are left invariant by stabilizer multiplication:

$$\mathcal{C} = \left\{ |\psi\rangle \in (\mathbb{C}^2)^{\otimes n} \;\middle|\; S|\psi\rangle = |\psi\rangle, \;\; \forall S \in \mathcal{S} \right\}. \tag{1.30}$$

Because all elements of $\mathcal{S}$ commute ($\mathcal{S}$ is abelian), they share a common basis of orthonormal eigenstates, so that if $\mathcal{S}$ is generated by $m = n - k$ independent stabilizers, then $\mathcal{C}$ is a 2^k-dimensional subspace of the Hilbert space, i.e. the code encodes k logical qubits into n physical qubits. Pauli operators that preserve the code space should commute with all stabilizers, this corresponds to the normalizer of $\mathcal{S}$ in $\mathcal{P}_n$ denoted by $\mathcal{N}(\mathcal{S})$

$$\mathcal{N}(\mathcal{S}) = \{ P \in \mathcal{P}_n \mid P\mathcal{S} = \mathcal{S}P \}. \tag{1.31}$$

We will be interested in operators that preserve the code space up to stabilizer multiplication but are not themselves stabilizers. These form the quotient group

$$\mathcal{N}(\mathcal{S})/\mathcal{S} \cong \mathcal{P}_k. \tag{1.32}$$

By a counting argument, this group has $2k$ independent generators. Moreover, since it inherits the commutation structure of $\mathcal{P}_n$, it is isomorphic to the Pauli group on k qubits. We can now fix the logical Pauli operators of $\mathcal{C}$ by picking $2k$ independent generators $\overline{X}_1, ..., \overline{X}_k, \overline{Z}_1, ..., \overline{Z}_k$ of $\mathcal{N}(\mathcal{S})/\mathcal{S}$ that follow the usual commutation relations for all i, $j \leq k$

$$[\overline{X}_i, \overline{X}_j] = 0, \qquad [\overline{X}_i, \overline{Z}_j] = 2\delta_{ij}\overline{X}_i\overline{Z}_j, \qquad [\overline{Z}_i, \overline{Z}_j] = 0. \tag{1.33}$$

We are now ready to discuss the effects of errors on the code, recall that we can restrict ourselves to Pauli errors without loss of generality. Consider a Pauli error $E \in \mathcal{P}_n$. For E to be undetected, it must commute with all stabilizers,

[2] At the notable exception of leakage.

i.e., $E \in \mathcal{N}(\mathcal{S})$. Such an E is either a stabilizer (acts trivially on the code space and hence is harmless) or a logical operator (and causes a logical error). We define the code distance as the minimal weight of a nontrivial logical operator, a value that characterizes how well information is protected within the code space

$$d = \min_{E \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}} |E|. \tag{1.34}$$

The distance d, along with the number of encoded logical qubits k and the number of physical qubits n, are the key figures of merit of a quantum error-correcting code, and in that case the code is referred to as an $[\![n, k, d]\!]$ code.

The error-correction procedure begins by measuring the m stabilizer generators. In the absence of errors, all measurements yield $+1$, and the encoded state remains unchanged. However, in the presence of errors, the measurements project the state onto one of the 2^m orthogonal subspaces associated with the measurement outcomes $s \in \{\pm 1\}^m$. This is equivalent to having applied some Pauli error $E_s \in \mathcal{P}_n$ (up to stabilizer multiplication) to the code state in the first place. To set back the system to the code space, it is sufficient to apply a Pauli correction operator $C_s \in \mathcal{P}_n$ that anti-commutes with the same set of stabilizers as E_s, and the accumulated operation is then $C_s E_s$. This procedure is successful if $C_s E_s$ belongs to the stabilizer group (i.e. corresponds to a trivial operation on the logical qubits). If instead $C_s E_s$ represents a nontrivial logical operator, the correction fails, resulting in a logical error. The task of chosing the correction operator C_s as a function of s is usually done by a classical computer that implements a *decoder* — that is, a map of the form

$$\mathcal{D} : \{\pm 1\}^m \to \mathcal{P}_n. \tag{1.35}$$

that assigns to each syndrome a Pauli correction operator. Optimal decoding, in the sense that of minimizing the probability of a logical error, corresponds to computing the most likely equivalence class of errors consistent with s according to the noise model, for example, some independent and identically distributed (i.i.d.) noise model, and is known as the *Maximum-Likelihood* decoder (ML). This, however, corresponds to summing probabilities over an exponentially large space and is intractable in general[3]. When employing a code for error correction,

[3]In fact, it is #P-complete [81], and notably harder than optimal decoding of classical linear codes, which is already NP-hard.

it is crucial to have an efficient decoder that approximates this task[4], for example by computing the most likely particular error consistent with s — i.e., the one with *minimal weight*. This would for example already be sufficient to correct all abritrary errors of weight up to $\lfloor \frac{d-1}{2} \rfloor$. Remark that for an i.i.d. noise model of parameter ε, if every error configuration leading to a logical failure contains a minimal error configuration of weight $\frac{d+1}{2}$, and if the number of such minimal configurations is at most exponential in d, then the logical error probability is bounded by some $(B\varepsilon)^{\frac{d+1}{2}}$ that goes to zero when increasing d for all $\varepsilon < \varepsilon_{\text{th}} := B^{-1}$. We say that ε_{th} is the threshold of the error-correction protocol, a value that, importantly, will depend on the noise model, and both the code and the decoder.

Here we have assumed that all measurements are perfect, this correspond to the so-called *code-capacity model*, where we wait for the decoder to finish before a next round of noise.

Definition 1 (Code-capacity model)**.** *The code-capacity noise model assumes that each round of error correction consists of the application of Pauli error on data qubits with i.i.d. probability ε, perfect syndrome extraction, and perfect recovery operations applied thereafter.*

The specific type of Pauli error is left unspecified for now, but in what follows we will typically consider depolarizing noise or bit-flip noise. In general, measurements will however also be noisy, possibly even more so than data qubits depending on the physical platform, because they are typically performed using an ancillary qubit and more noisy 2-qubit gates, so that a realistic setting would favor the use of the *phenomenological model*.

Definition 2 (Phenomenological model)**.** *The phenomenological model assumes that each round of error correction consists of the application of Pauli error on data qubits with i.i.d. probability ε_d, flipping measurement outcomes with i.i.d. probability ε_m, and perfect recovery operations applied thereafter.*

Note that in both models we have assumed that errors at different space-time locations in the circuit are independent. While this is a reasonable assumption, some level of correlation can be incorporated by considering the more general *local-stochastic noise model*, provided that errors remain exponentially

[4]On the contrary, for cryptographic purposes, one seeks codes for which decoding is hard in general but becomes easy when given some secret key [108].

suppressed with their weight, so that the probability that a space-time set of faulty locations contains a specific set of A locations is upper bounded by $\varepsilon^{|A|}$.

A large variety of quantum error-correcting codes exists, and the most comprehensive list can be found in the Error Correction Zoo maintained by Victor Albert and Philippe Faist [6]. We start by the simple repetition code that only protects against bit-flip noise, before introducing Kitaev's surface code as a concrete example to illustrate the concepts introduced above. We then discuss how to perform logical operations on an encoded state in the next section.

Quantum repetition code. The first naive encoding one can think of by taking inspiration from the simple classical repetition code, is the one of the quantum repetition code. It corresponds to the following choice of encoding

$$\left|\overline{0}\right\rangle := |0^n\rangle\,, \qquad \left|\overline{1}\right\rangle := |1^n\rangle\,, \tag{1.36}$$

where $0^n = 0\cdots 0$ is the n zero bitstring and similarly for 1. Alternatively, in stabilizer language it can be described by placing qubits on the n edges on a cycle and stabilizers $S_i := Z_{(i-1,i)}Z_{(i,i+1)}$ on its vertices $i \in \mathbb{Z}_n$ as illustrated in Figure 1.4 (a). It is easy to see that the X logical operator is $\overline{X} = X_{(0,1)} \otimes ... \otimes X_{(n-1,n)}$ and corresponds to flipping all qubits. The operator is of weight n so that similarly to the classical case, the bit-flip distance is n. The syndrome of an X-type error E defined on the edges of the cycle corresponds to the boundary of the error, that we refer to by defects which should be matched by the decoding algorithm. This task is efficiently performed by the Minimum-Weight Perfect Matching (MWPM) decoder [53, 49] that minimizes the weight (number of non-trivial Pauli components) of the correction, as illustrated in Figure 1.4 (b). Concerning phase-flip errors, it suffices to apply a single Pauli Z on any qubit to flip the phase of the encoded state, that is to say the logical Z operator is $\overline{Z} = Z_{(i,i+1)}$ for all $i \leq n$, and the distance of the code is $d = 1$.

This means that the quantum repetition code does not protect against general quantum noise. This is perhaps not surprising since the repetition code is a classical code with only Z-type stabilizers, hence unable to detect any Z error. It nevertheless remains an intuitive concept that can become useful when combined with biased-noise qubits, that it to say qubits suffering only from bit-flip noise, an example of which are cat qubits [112, 127]. Its limitations however motivate the construction of quantum codes, with for example both Z and X

Figure 1.4: (a) Representation of a quantum repetition code. The tiling has periodic boundary condition and is therefore a cycle. The qubits are placed on the edges, the Z-type stabilizers (parity check) are defined by edges incident to a vertice (blue). (b) Bit-flip errors (in red) are detected at their boundary by parity measurements, whose -1 outcomes are refered to as defects represented in dark blue. The decoding task aims at merging those defects with the light blue correction, which that can be done efficiently with e.g. MWPM.

stabilizers that can detects all type of Pauli errors.

Kitaev's toric code. The paradigmatic Kitaev's toric code [85] belongs to the family of topological codes, here defined on a two-dimensional torus. Such codes are obtained from the tessellation of a finite dimensional manifold, on which are defined local stabilizers. Consider the 2D periodic lattice $\mathcal{L} = \mathbb{Z}_d \times \mathbb{Z}_d$ for some integer d. The toric code is defined by assigning qubits to edges, Z-type stabilizers to vertices and X-type stabilizers to plaquettes in a manner illustrated in Figure 1.5.

The number of qubit is the number of edges that is $2d^2$. We have defined d^2 stabilizer of each type (X and Z) but at each time, the multiplication of all Z stabilizers assigned to vertices yields the identity, so that only $d^2 - 1$ are independent. The same relation holds for all X stabilizers, which leaves $2d^2 - 2(d^2 - 1) = 2$ degrees of freedom and the code encodes two logical qubits. It is left to determine what are the non-trivial logical operators. The $\overline{X}_1$ and $\overline{Z}_1$ logical operators of the first logical qubit can be respectively assigned to the Pauli X string performing a non-trivial 'horizontal' loop around the torus, and the Pauli Z string performing a non-trivial 'vertical' loop. The two Pauli strings can be seen to commute with stabilizers while they cannot be generated from them. Switching X and Z then gives the logical operators of the second logical qubit. Finally, it is easy to see that those operators are of weight at least d, so that increasing the size of the lattice increases the code distance and the Kitaev's surface code is a $[\![2d^2, 2, d]\!]$ code.

Kitaev's toric code can famously be adapted to fit on a strictly 2D square layout that is usually required on superconducting platforms, by taking the form of the *surface code* [56]. The surface code remains as of today the leading ex-

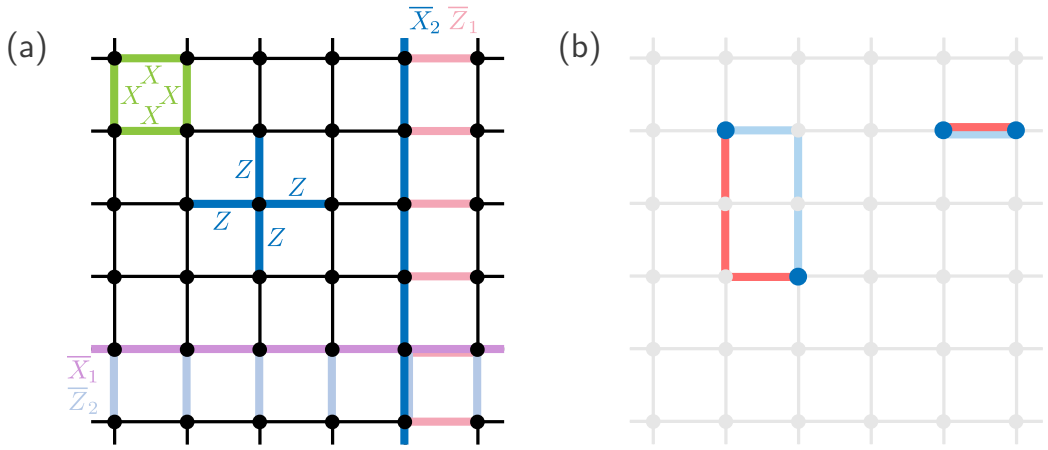

Figure 1.5: (a) Representation of a toric code. The tiling has periodic boundary condition and is therefore a torus. The qubits are placed on the edges, the Z-type stabilizers are defined by the vertices (blue) and the X-type stabilizers by the plaquettes (green). There are two logical qubits, whose operators are represented by non-contractible loops around the torus represented in brown ($\overline{X}_1$ and $\overline{Z}_1$) and grey ($\overline{X}_2$ and $\overline{Z}_2$). (b) Bit-flip errors (in red) are detected at their border by vertices measurements, whose -1 outcomes are refered to as defects represented in dark blue. The decoding task aims at merging those defects with the light blue correction, which that can be done efficiently with e.g. MWPM. In the example one of the correction does not exactly correspond to the initial error, but this difference is harmless since they only differ by a X-type stabilizer of the toric code. Phase-flip errors can be treated similarly and simultaneously in the dual lattice.

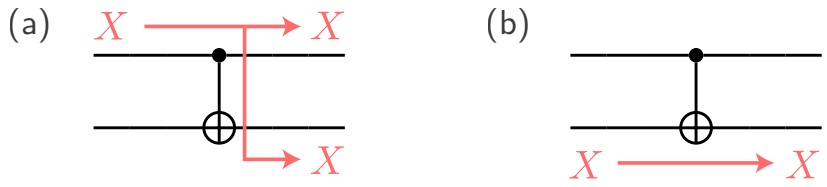

Figure 1.6: The propagation of an error through a given gate U is obtained by commuting the error through U. (a) A Pauli X error on the control qubit of a CNOT gate propagates to the target qubit so that it is equivalent to two X errors after the gate. (b) A Pauli X error on the control qubit of a CNOT gate does not propagate. Because an Hadamard gate maps the target to the control and vice versa, the situation is reversed for a Z error.

perimental platform for which the error correction experiments below threshold have recently been performed [1] for the first time.

1.3.3 Fault-tolerant quantum computation

Having a long-lived qubit may be a necessary condition for useful quantum computation, but it is not sufficient. We must also be able to perform logical operations while maintaining protection against noise — something that is far from trivial. The key assumption of quantum error correction is that noise is local, so that errors can be treated as arising independently. Logical operations however may require qubits within the same code block to interact between each other, allowing errors to propagate as illustrated in Figure 1.18, possibly leading to highly correlated errors. Preventing such propagation of errors is the goal of fault-tolerant quantum computation, culminating in the quantum fault-tolerance theorem that ensures that any computation can in principle be performed up to arbitrary precision at the cost of an only polylogarithmic space-time overhead.

Assume that we have encoded some k-qubit state $|\psi\rangle$ into a n physical qubit state $|\overline{\psi}\rangle$ with encoding map $\mathcal{E}$. We have already discussed how to implement Pauli operators in stabilizer codes. To perform arbitrary quantum computations, however, it is necessary to implement a universal gate set, such as $\{H, \mathrm{CNOT}, T\}$. This may be difficult because we need the candidate unitary U to (i) preserve the code space, so that the output state is still stabilized by $\mathcal{S}$, (ii) to implement the correct logical operation, which requires ensuring that the logical operators transform in the same way as their physical counterparts

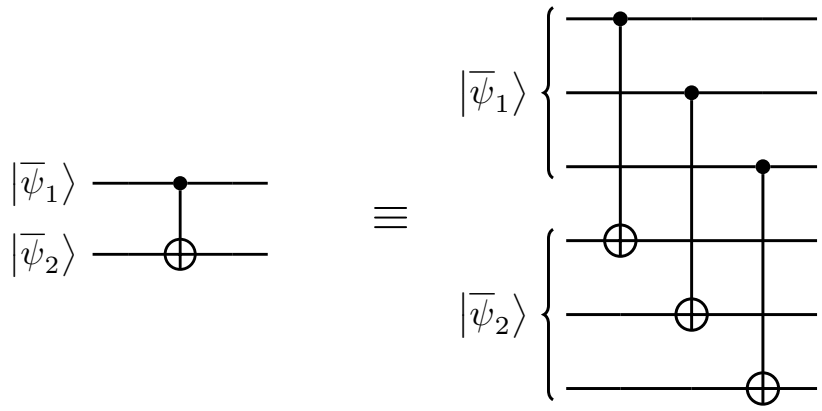

Figure 1.7: Example of a transversal CNOT gate between two CSS codes (e.g. repetition and toric codes). A transversal CNOT gate corresponds to pairing qubits from the two code blocks and applying CNOT gates in parallel between them, preventing a single error to propagate to more than two qubits.

under U, (*iii*) while ensuring that a single-error propagates to at most a constant number of qubits (including those used for measurements), so that the circuit is fault-tolerant. Note that the later condition is typically satisfied for constant depth circuits for gates acting on a constant number of qubits.

Things turn out well for the CNOT gate that is a *transversal gate* for both repetition and toric codes[5], meaning that $\overline{\text{CNOT}} = \text{CNOT}^{\otimes n}$ as illustrated in Figure 1.7. Transversal gates between code blocks are the simplest and often preferred form of fault-tolerant gates, because an error on a single qubit propagates to at most one other qubit per code block involved. We may now try to implement a logical Hadamard. It is easy to see that already for the simple repetition code, applying a Hadamard gate on each qubit does not implement a Hadamard gate, this is because $|\overline{+}\rangle = \frac{1}{\sqrt{2}}(|0^n\rangle + |1^n\rangle) \neq |+^n\rangle$. A possible solution would be to decode the quantum repetition code to map the logical information to a single qubit, apply the Hadamard gate, and later encode it back to the code space. While this unitary indeed performs a logical Hadamard, it is clearly non fault-tolerant because for example a single error on the first qubit at the begining will propagate to all, or alternatively because beween decoding and encoding, the logical information is no longer protected by the code.

[5]The CNOT gate is actually transversal on all codes encoding a single-qubit with only X and Z-types stabilizers, called CSS (Calderbank-Steane-Shor) codes, because of the way X and Z Paulis commute through a CNOT gate (see Figure 1.6).

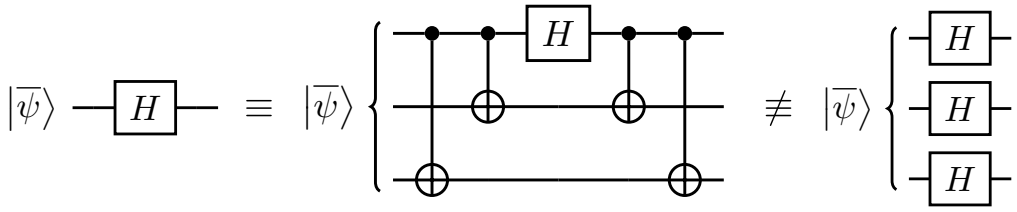

Figure 1.8: The Hadamard gate is not transversal on the repetition code. A logical Hadamard can be implemented by a decoding unitary mapping the logical information to a single qubit, applying a physical Hadamard on it and encoding back to the code space. The logical information is however left unprotected beween decoding and encoding, so that the circuit is clearly not fault-tolerant.

One might think that the difficulty lies with a particular encoding, and that other codes could support a transversal Hadamard gate. Indeed, some codes — such as self-dual codes like the 2D color code [24], where the X and Z logical operators and stabilizers have identical support — do admit a transversal Hadamard. However, the Eastin-Knill theorem [52] rules out the possibility of any quantum code supporting a universal set of gates entirely through transversal operations. In other words, while individual gates may be implemented transversally in certain codes, it is fundamentally impossible to realize a full universal gate set in this manner.

Theorem 3 (Eastin-Knill [52])**.** *No finite-dimensional quantum error correcting code can transversally implement a universal gate set.*

Nevertheless, ingenious schemes have been developed to enable the fault-tolerant implementation of a universal gate set. These include, for instance, code switching between different quantum error-correcting codes whose combined set of transversal gates is universal [118, 22], as well as gate teleportation techniques [67], which shift the complexity to the preparation of high-fidelity magic states [28]. These approaches culminate in the quantum fault-tolerance theorem, which guarantees that, below a certain constant noise threshold, any quantum computation of polynomial size can be efficiently and reliably simulated on a realistically noisy quantum device.

Theorem 4 (Quantum fault-tolerance [3, 116, 86, 85])**.** *A quantum circuit C containing G gates may be simulated with probability of error at most δ on circuit C' with*

$$G' = \mathcal{O}(G \cdot \mathrm{polylog}(G/\delta)) \tag{1.37}$$

gates, on hardware whose components fail independently with probability at most ε, provided ε is below some constant threshold, $\varepsilon < \varepsilon_{\text{th}}$.

Note that the asymptotic overhead of fault-tolerant quantum computation can be reduced from a polylog to a constant [55, 156], thanks to recent advances in quantum LDPC codes with constant encoding rate [100, 121]. While these results offer a compelling proof of principle, improved asymptotic scaling does not necessarily imply practical low-overhead fault tolerance, as it often comes with large constant factors and long-range connectivity. As a result, the most efficient way to build a scalable quantum computer remains an open question and is likely to depend heavily on the underlying physical platform. In the next subsection, we examine two of the most advanced fault-tolerant platforms to date: superconducting qubits [1] and atom arrays [16, 17].

1.3.4 Constraints from experimental platforms

A meaningful attempt at building a fault-tolerant quantum computer should start by assessing what are the contraints posed by realistic hardware. Some of those constraints are very general: multiple qubit gates become less and less reliable when increasing the number of qubits involved. This has led to the consensus that LDPC (Low-Density Parity-Check) codes should be preferred, where the stabilizers are of bounded size and each qubit is in the support of a bounded number of stabilizers. Importantly, this constraint does not forbid to achieve asymptotically good codes [100, 121], meaning with non-vanishing encoding rate and distance increasing linearly with the number of qubits. In addition to this LDPC property, it has also been generally preferable for the code to be geometrically local—meaning that qubits can be arranged in a 2D layout or 3D space with a lower-bounded qubit density and an upper-bounded stabilizer size. This has led to the paradigmatic surface code [56] that remains as of today the leading experimental platform with recent first-of-their-kind demonstrations of a logical qubit below threshold [1, 17].

The task of scaling up a machine from a single logical qubit to a universal fault-tolerant quantum computer operating $\sim$ 100–1000 logical qubits for useful applications however remains colossal. Taking the example of superconducting hardware, each qubit requires its own control equipment — such as wires and pumps — inevitably introducing additional sources of noise. Importantly, different physical platforms come with distinct challenges but also offer unique

Physical platform	Superconducting [1]	Neutral-atoms [16]
Coherence time	$\sim 100\mu$s	$\sim$ 1s
1 / 2-qubit gate time	$\lesssim$ 50ns	$\sim$ 500ns / $\sim$ 200ns
1 / 2-qubit gate fidelities	99.97% / 99.7%	99.97% / 99.5%
Mid-circuit measurement time	$\gtrsim$ 100ns	$\sim$ 10ms
Measurement fidelities	99.3%	99.7%
Connectivity	2D nearest neighbors*	Reconfigurable
Logical CNOT (for surface code)	Lattice surgery	Transversal

Table 1.1: Comparison of leading fault-tolerant quantum computing platforms — superconducting qubits and neutral-atomes — focusing on metrics relevant for fault-tolerant architecture design. *Although recent experiments point toward the development of long-range couplers.

opportunities, which will inevitably shape the choice of a fault-tolerant architecture. We will dedicate this subsection to assessing those properties for the two experimental leading platforms: superconducting qubits and neutral-atoms compared in Table 1.1.

In the superconducting *transmon* [87], one takes advantage of the non-linearity of a Josephson junction to create an anharmonic potential in which the ground and excited states are isolated from the rest of the spectrum and can be used to store quantum information. Several variants of this fundamental design exist but, in general, superconducting qubits can take advantage of fast gate operations and scalable fabrication techniques from the semiconductor industry, albeit usually limited to 2D nearest neighbor connectivity. Having a fast clock cycle can be interesting, as it can ultimately lead to machines that are several orders of magnitude faster than competing technologies, all else being equal, but this also tightens constraints on classical control, particularly in terms of wiring, communication and decoding speed. We discuss this in more details in Section 1.4 which motivates Chapters 2 and 3.

More recently, proposals for a logical processor based on *neutral-atoms* have seen tremendous progress [16, 17, 43, 131]. There, quantum information is encoded in the internal quantum states of individual atoms, trapped and manipulated with laser beams. Those experiments can take advantage of a large number of individual qubits (up to 3000 in a recent experiment [43]) that can be manipulated in a highly parallel manner in 3D space, allowing to implement transversal gates between two code blocks. The proposal leverages reconfigurable arrays of atoms with a high degree of multiplexing, to facilitate scaling

to larger numbers of qubits. The main challenges however remain the slow clock cycle mainly because of slow mid-circuit measurements, along with loss of atoms during the computation, both pointing towards a preference for parallelizable architectures. This will be the topic of Chapter 4 motivated in Section 1.5.

1.4 Local protection of a quantum memory

1.4.1 Limits of global decoders

Quantum error correcting codes protect information in a noisy quantum computer by delocalizing it across a higher-dimensional Hilbert space. Topological codes are a particularly efficient approach to realize a quantum memory where one enforces local constraints on physical qubits placed on a surface [85]. This locality property simplifies experimental implementation for certain physical platforms, particularly superconducting qubits, where the fabrication on a surface constrains qubit connectivity to nearest neighbors in two dimensions. Topological codes generally display a good resistance to noise, and the prototypical 2D surface code [56] remains today the leading experimental platform [5, 90, 1].

The locality of constraints however does not suffice to make the memory fully local, if the decoding process in itself is not. This is typically the case of known efficient decoding algorithms of the surface code, Minimum-Weight Perfect Matching [53, 49], Union-Find [48] or Renormalization Group [51] decoders, that take as input the entire error syndrome (all stabilizer generator measurement outcomes) and output a Pauli correction. The problems posed by *global decoders* fall into three categories.

First, and perhaps more importantly, they impose additional hardware constraints to enable the transmission of stabilizer measurement outcomes to a centralized classical computer, e.g. located outside the cryostat in superconducting qubit architectures.

Then, the corresponding classical communication and computation must be performed fast enough to avoid the so-called *backlog problem* [145], where the accumulation of pending decoding tasks (due to slow classical processing) forces an exponential slowdown of the overall quantum computation as errors and decoding tasks pile up faster than they can be handled. In short, we need fast decoders. Note that the decoding speed needs to be compared to the gate

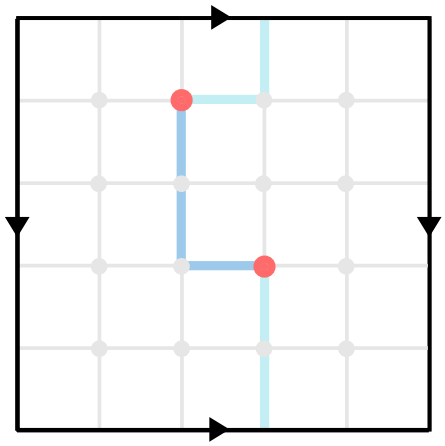

Figure 1.9: The effect of measurement errors when decoding the toric code. A pair of measurement errors is indicated in red. Two possible errors with consistent syndrome are shown in different shades of blue. The two possible corrections are of weight $\Theta(d)$, so the decoder will end up introducing an error of macroscopic weight due to a constant number of measurement errors.

and coherence time of the physical platform that is actually considered, making it a challenging problem especially for faster systems like superconducting qubits. Characteristic times for the most advanced technologies in terms of fault-tolerance are given in Table 1.1.

Finally, if the fundamental assumption of error correction is that errors are local, a non-local decoder can map local measurement errors into a non-local error on the output correction. This can be easily understood from the example of Figure 1.9 for the distance d toric code, where 2 measurement errors are interpreted as an error string of length $\mathcal{O}(d)$ connecting the two erroneous measurements. Measurement errors can typically be addressed with extra redundancy in the syndrome, either in space by exploiting higher-dimensional codes with a so-called *single-shot* property, as initially defined by Bombín [23], or in time by storing the error syndrome in memory for a duration of order $\mathcal{O}(d)$, effectively extending the decoding graph by one dimension to incorporate the temporal information [49]. Importantly, the Pauli correction needs in general to be applied before the next round of non-Clifford gates, hence this scheme is at the cost of an $\mathcal{O}(d)$ time overhead[6].

1.4.2 Self-correcting quantum memories

An attractive alternative to this active error correction strategy would be to rely on passive error correction. The utility of this concept is well illustrated in the classical setting with the well-known example of the Ising model. Consider a n-site periodic lattice of dimension D and size ℓ with each site v assigned a spin

[6] for transversal gates, the time overhead can be reduced to $\mathcal{O}(1)$ at the cost of additional decoding complexity, see [158]

$\sigma_v \in \{-1, +1\}$. Without an external magnetic field and ignoring interactions beyond nearest neighbors, the system is governed by the Ising Hamiltonian

$$H(\sigma) = -\sum_{\langle i,j \rangle} \sigma_i \sigma_j, \tag{1.38}$$

where $\langle i, j \rangle$ indicates nearest neighbor pairs. Intuitively, the $-$ sign indicates that it is energetically favorable for spins to be aligned. It is then straightforward to verify that the lowest-energy configurations correspond to all aligned spins, i.e. $\sigma_- = (-1)^n$ and $\sigma_+ = (+1)^n$. More importantly, the model famously shows a phase transition in dimension $D \geq 2$, meaning that below some critical temperature, the two stable configurations $\sigma_- = (-1)^n$ and $\sigma_+ = (+1)^n$ become robust against thermal fluctuations, with lifetime diverging with ℓ. This phenomenon allows to reliably store classical information in the macroscopic magnetization of ferromagnetic material, an essential principle behind technologies such as hard drives.

A quantum system with a similar property would be of great practical value, as it would eliminate the need for active error correction and the associated overhead of stabilizer measurements, fast classical decoding, and feed-forward operations. This is the premise of so-called *self-correcting* quantum memories [41, 157, 98, 35], where information is encoded in the degenerate ground states of a n-body geometrically local Hamiltonian. The system is assumed to interact with a Markovian thermal environment, the detailed description of which is beyond the scope of this thesis but can be found in [35]. Given a decoding map $\mathcal{D}$, we can define the memory time of information encoded in an initial state $\rho^{(0)}$, up to an error $\delta > 0$ by,

$$\tau = \sup \left\{ t > 0 \,|\, \|\mathcal{D}(\rho^{(t)}) - \rho^{(0)}\|_1 < \delta \right\}, \tag{1.39}$$

where the ℓ_1 norm of a vector is the sum of the absolute values of its components. Below some critical temperature, the quantum information should be recoverable for abritrarily long time meaning that τ diverges in the limit where $n \to \infty$. A natural candidate construction arises from an n qubit stabilizer code $\mathcal{C}$ defined from its stabilizer group $\mathcal{S} = \langle S_1, ..., S_m \rangle$, with local generators in D dimensions. Recall that the code space is

$$\mathcal{C} = \left\{ |\psi\rangle \in (\mathbb{C}^2)^{\otimes n} \;\middle|\; S|\psi\rangle = |\psi\rangle, \;\; \forall S \in \mathcal{S} \right\}. \tag{1.40}$$

A candidate Hamiltonian can be constructed from $\mathcal{C}$ by setting

$$H = -\sum_{i=1}^{m} S_i, \tag{1.41}$$

that is both local and frustration free by definition of $\mathcal{C}$. The energy of a state $|\phi\rangle$ is

$$E_\phi = \langle\phi|H|\phi\rangle, \tag{1.42}$$

so that the ground space $\mathcal{H}_0$ with ground energy E_0 is

$$\mathcal{H}_0 = \left\{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} \mid \langle\phi|H|\phi\rangle = E_0\right\}, \qquad E_0 = \min_{|\phi\rangle}\langle\phi|H|\phi\rangle. \tag{1.43}$$

Notice that in this framework, the weight of the error syndrome is analogous to the energy of the excitation, so that states with non-trivial error syndrome are energetically unfavored. It can easily be verified that code-states of $\mathcal{C}$ have minimal energy ($E_0 = -m$) and hence belong to the ground space of H — meaning it can be used to encode quantum information.

$$\mathcal{H}_0 = \mathcal{C}. \tag{1.44}$$

The existence of a degenerate ground state however does not suffice to ensure an increasing coherence time with system size at constant temperature — which is a much stronger property. It is in fact known that, in stark contrast with the classical case, quantum memories obtained from stabilizer codes in 2 dimensions or less cannot be self-correcting [30]. This can be readily explained by introducing a new idea.

A simple but useful concept in understanding thermal stability of a quantum memory is the one of the energy barrier. Let $|\psi\rangle$ and $L \in \mathcal{L}$ be respectively an encoded state and an encoded logical operator. Consider a decomposition of L into a sequence of single qubit Pauli operators $\{P_i\}_{i=1}^m$ representing the effect of local noise,

$$L = \prod_{i=1}^{m} P_i. \tag{1.45}$$

We define a path to be the sequence of quantum state $\gamma = \{|\psi_t\rangle\}_{t=1}^m$ such that for all $t \geq 0$, $|\psi_t\rangle = \prod_{i=1}^t P_i|\psi\rangle$. The set of all such paths implementing a logical operator L is denoted by $\Gamma(L)$. We are interested in the energy required

to create a logical error. The initial and final states are in the code space and thus have minimal energy. However, the intermediate states along the path may not. The energy cost $\Delta(L)$ of a logical operator L is defined as the minimum over all possible paths $\gamma \in \Gamma(L)$ of the maximum energy encountered along γ

$$\Delta(L) := \min_{\gamma \in \Gamma(L)} \max_{|\phi\rangle \in \gamma} (E_\phi - E_0). \tag{1.46}$$

The energy barrier of the code is simply given from minimizing over the energy cost of logical operators

$$\mathcal{E} := \min_{L \in \mathcal{L}} \Delta(L). \tag{1.47}$$

Although one can define a notion of protected qubit with only a constant energy gap [50], an energy barrier diverging with n is likely necessary for a thermally stable memory[7], as has been shown for a large class of 2D topological phases [83, 144, 88]. In this case it is expected that the coherence time scales with system size accordingly to Arrhenius' law

$$\tau \sim \exp(\beta \mathcal{E}), \tag{1.48}$$

with β the inverse temperature. For instance, in the 2D Ising model described earlier on a $\ell \times \ell$ lattice, considering only bit-flip type errors, the energy barrier corresponds to the minimum number of disaligned spins while going from σ_- to σ_+ by flipping one spin at a time. Since violated spins separate regions with spin up and spin down, hence the bit-flip energy barrier is $\mathcal{E}_X = 2\ell$ yielding a memory time diverging exponentially in ℓ. Large energy barrier are however more difficult to reach in low dimensions for quantum codes, a typical example of which is the case of the toric code (and surface code by extension) with following associated Hamiltonian

$$H = -\sum_p X_p - \sum_v Z_v, \tag{1.49}$$

and interactions terms X_p and Z_v are respectively assigned to plaquettes and vertices as shown in Figure 1.5. Here a single X error on edge e creates a pair of excitations — the two Z checks with support on site e. Notice that successively appending errors along a chain displaces the two violated checks,

[7] although the condition is not strictly sufficient in general, as entropic effects need to be taken into account, see [72, 111, 141]

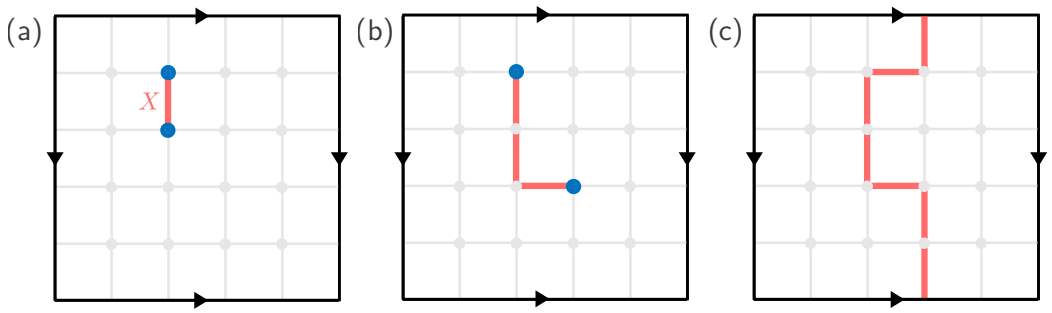

Figure 1.10: Logical error at a constant energy cost for the toric code. (a) An intial single qubit error violates the two checks with non-trivial support on said check. Violated checks are represented in blue. Successively appending errors along a chain (b) displaces the two violated checks, (c) ultimately enabling the creation of a logical error at constant energy cost.

ultimately enabling the creation of a logical error at constant energy cost, as illustrated in Figure 1.10, thus preventing the system from being self-correcting. This constant energy barrier is due to the string-like nature of logical operators in the surface code. A well-known result is that this property is in fact common to all local stabilizer codes in two dimensions or lower.

Theorem 5 (Bravyi-Terhal [30]). *Let $\mathcal{S} = \langle S_1, ..., S_m \rangle$ be a stabilizer set with local generators on a 2D lattice, such that each qubit participates in a constant number of generators. Then the associated code energy barrier is upper bounded by a constant independent of the lattice size ℓ.*

Because the coherence time is expected to scale at best according to Arrhenius' law, this result excludes the possibility of self-correcting quantum memories from stabilizer codes in two dimensions. Note that although the two-dimensional constraint is crucial, as it matches the typical layout of superconducting qubits, the three dimensional case may nevertheless be interesting for alternative platforms. If the 4D toric code is known to be self-correcting [7, 85] thanks to its membrane-like logical operators, whether self-correction is possible in three dimensions remains an outstanding open question in the field [29, 104]. We list the self-correcting properties of standard classical and quantum memories in Table 1.2.

Code	Lattice	Distance	Self-correction	Energy barrier
1D repetition code	ℓ	$d_X = \ell$	✗	$\mathcal{E}_X = 2$
2D repetition code	$\ell \times \ell$	$d_X = \ell^2$	✓	$\mathcal{E}_X = \Theta(\ell)$
2D toric code	$\ell \times \ell$	$d = \ell$	✗	$\mathcal{E} = 2$
4D toric code	$\ell \times \ell \times \ell \times \ell$	$d = \ell^2$	✓	$\mathcal{E} = \Theta(\ell)$

Table 1.2: Self-correction properties and energy barrier of various classical and quantum memories, where we use the bit-flip distance and energy barrier for classical codes. The 1D repetition code (1D Ising model) and the 2D surface/-toric code fails to be self-correcting because of their string-like logical operators, resulting in constant energy barrier. On the other hand, the 2D repetition code (2D Ising model) and the 4D toric code supports are respectively classical and quantum self-correcting quantum memories. However, it is important to note that the energy barrier in these two codes is not equal to the code distance itself, but instead scales as the square root of the distance.

1.4.3 From cellular automata to local decoders

Self-correcting quantum memories aim at adressing the challenge posed by the need for centralized classical computation by eliminating the need for classical computation altogether, at the cost of extra dimensions and important performance loss because of lifetime typically exponential in some power $\alpha < 1$ of the distance ($\alpha = 1/2$ in the examples listed in Table 1.2 while optimal decoders yield $\alpha = 1$). An intermediate approach involves designing the decoding process so that it can be implemented in a streamlined, parallel architecture situated close to the quantum hardware eliminating the need for long-range classical communication. This is the premise of local decoders, also known as cellular automaton decoders.

The interest in cellular automata originates from the early days of classical computation, pioneered by John von Neumann [151] and Edward F. Moore [113], who were motivated by the goal of reproducing the rich dynamics of biological systems and the human neural system, whose complexity emerges from local interactions. A cellular automaton is defined on a lattice $\mathcal{L}$ of size m with finite neighborhood $\Gamma \subset \mathcal{L}$, where each site i of the lattice is assigned a register u_i with value in a finite alphabet A. A configuration of the automaton is then $u = (u_1, ..., u_m) \in A^{\mathcal{L}}$. The dynamics is defined recursively by a local update rule $f : A^{\Gamma} \to A$ applied uniformly across all sites

$$\forall i \in \mathcal{L}, \quad u_i^{(t+1)} = f(u^{(t)}|_{i+\Gamma}). \tag{1.50}$$

Where we note $u|_{i+\Gamma} = (u_{i+z})_{z\in\Gamma}$. Interestingly, a rich variety of macroscopic behavior is known to emerge from simple local update rules. For example, Conway's Game of Life [13] uses a simple set of local rules on a rectangular grid of cells, each holding a single bit. The state of each cell at the next time step depends on its current state and the sum of its eight nearest neighbors. This simple setup allows for configurations that can grow indefinitely or even simulate arbitrary computations. Other examples more closely related to the subject of this thesis include the task of density classification. The goal here is to perform a majority vote one-dimensional array of bits through successive homogeneous application of a 1D local update rule.

Task C1 (Density classification)**.** *Given an initial binary configuration, update the configuration using a 1D local update rule so that the system converges to the uniform configuration of all 1s if the initial density of 1s is greater than 0.5, and to all 0s otherwise.*

From an error-correction point of view, this corresponds to decoding the 1D repetition code in the code-capacity setting, that is to say starting from an erroneous configuration but without additional errors while decoding. Note that the task is difficult because, contrary to the usual error-correction task where we assume the error rate to be small, here we consider symmetrically distributed initial configurations, i.e. corresponding to an error rate of 50%. Nethertheless, Gács, Kurdyumov, and Levin found an automaton (known as the GKL automaton), that solves this majority problem in 78% of random cases, by taking local majority votes on the neighborhoods $V_0 = \{-3, -1, 0\}$ and $V_1 = \{0, +1, +3\}$ when the cell's state is 0 and 1, respectively [61]. Interestingly, the GKL automaton is able to erase an isolated finite island of errors in time proportional to its width. Under more relaxed constraints, Henryk Fuchs proposed a hybrid automaton that solves the majority problem [59] by alternating two rules. The machine is however not strictly speaking a cellular automaton, as the switching period scales linearly with system size which cannot be implemented from the uniform application of single rule on a fixed size alphabet. In fact, it has been shown that no perfect deterministic classifier exists in 1D [95], a result later extended to the stochastic setting and in any finite dimension [37].

We are however more interested in realistic noise models, where errors are rare but arise with constant probability in time. In this case, the local update rule is considered noisy, meaning that random flips occur independently with

i.i.d. probability , and we are interested in the stability properties of the system near the fixed points of the noiseless automaton.

Task C2 (Memory retention)**.** *Given an initial configuration and a update rule subject to noise parameter ε — which determines the probability of errors in the update rule — evolve the system over time and determine whether the evolved configuration remains close to the initial configuration despite the noise.*

A system that remains close to its initial state for a long time (typically exponential in the system size) is said to be non-ergodic in cellular automaton terminology. Non-ergodicity typically requires the noise rate to be below some constant threshold $\varepsilon_{\text{th}} > 0$, analogous to a phase transition at a critical temperature $T_{\text{c}} > 0$ in more physical terms. This is a stronger task to the one of erasing finite isolated islands, since the decoding process needs to be somewhat robust to subsequent noise, which explains why for example the GKL automaton does not achieve the latter task. The first construction exhibiting non-ergodicity is due to the Russian mathematician Andrei Toom [146], with an automaton now commonly referred to as *Toom's rule*. Toom's rule enables to safely store a bit of information in the macroscopic configuration of a 2D binary grid, by emulating a digital version of the Ising model with a simple update rule that flips a bit if it disagrees with both its West and South neighbors. Perhaps more suprisingly, a one-dimensional cellular automaton was shown to exhibit a similar behavior by Peter Gács in his seminal work [61, 60] (see also Gray's reader's guide [68]), violating the so-called 'positive rate' conjecture [103], motivated by classical Hamiltonian memories that lack long-range order at any finite temperature, e.g. the 1D Ising model. While a beautiful proof-of-principle, Gács automaton however offers limited practical applications to error correction due to its high complexity [107] and suboptimal performance.

In order to avoid the hardware implementation cost of global decoders in the quantum setting, it is tempting to consider cellular automata for a streamlined and highly parallel decoding architecture of quantum error-correcting codes. As discussed in the previous section, recall that compared with classical error correction, quantum error correction poses the additional constraints that we only have access to parities of the code — e.g. stabilizer measurement outcome forming the error syndrome — and we need to decode both bit-flip and phase-flip errors. That being said, we can also relax some defining assumptions of cellular automata, where both the data and the update rule correspond to clas-

sical hardware. In modern quantum fault-tolerant architectures, the reliability of classical microelectronics — thanks to decades of accumulated progress — can be safely assumed, and thus the noise model is typically confined to the quantum part of the computation, i.e. the data and ancillary qubits. Moreover, while classical cellular automata traditionally operate with a local state alphabet of constant size, i.e. $|A| = \mathcal{O}(1)$, in quantum error correction the emphasis may be on absolute local state space size and performance for reasonably sized systems, and not only on asymptotic scaling. Consequently, we relax the assumption of a fixed-size local alphabet and instead allow it to grow slowly (typically polynomially) with the system size, so that the local state can still be represented using a logarithmic number of bits. To avoid confusion with the terminology typically used for cellular automata, an automaton that satisfies this assumption will be referred to as a *local decoder*.

Let us start by formally defining the local update rule in this context. Consider a family of n-qubit LDPC, with sites corresponding to stabilizer generators, and a constant local neighborhood Γ. Here, the local update rule is no longer restricted to a classical local state, since it also takes as input the values of the local stabilizer checks and outputs a local Pauli correction acting on the $w = \mathcal{O}(1)$ qubits in the stabilizer's support.

Definition 3 (Local update rule)**.** *The local update rule maps the value of neighboring stabilizer checks and states to a Pauli correction and an updated local state*

$$f : \{\pm 1\}^{\Gamma} \times A^{\Gamma} \longrightarrow \mathcal{P}_w \times A, \tag{1.51}$$

with the two outputs obtained via projections

$$\pi_{\mathcal{P}} \circ f : \{\pm 1\}^{\Gamma} \times A^{\Gamma} \longrightarrow \mathcal{P}_w, \qquad \pi_A \circ f : \{\pm 1\}^{\Gamma} \times A^{\Gamma} \longrightarrow A, \tag{1.52}$$

onto the Pauli and local state components, respectively.

Given a local update rule f, a *local decoder* is obtained by applying f simultaneously and in parallel across all sites of the lattice $\mathcal{L}$.

Definition 4 (Local decoder)**.** *A local decoder in configuration $u \in A^{\mathcal{L}}$ is maps a syndrome $s \in \{\pm 1\}^{\mathcal{L}}$ to a Pauli correction through the parallel application of a local update rule f*

$$\mathcal{D}_u : \{\pm 1\}^{\mathcal{L}} \longrightarrow \mathcal{P}_n, \qquad \mathcal{D}_u(s) = \bigotimes_{z \in \mathcal{L}} (\pi_{\mathcal{P}} \circ f)(s|_{z+\Gamma}, u|_{z+\Gamma}). \tag{1.53}$$

After the correction is applied, the decoder configuration is updated as

$$u' = \big((\pi_A \circ f)(s|_{z+\Gamma}, u|_{z+\Gamma})\big)_{z\in\mathcal{L}}. \tag{1.54}$$

The full local decoding process is then defined from an inital configuration u_0, by the repeated application of $\mathcal{D}_u$ together with the updates of u.

We are interested in defining quantum analogue tasks to the ones of density classification and memory retention. Let us start with the density classification task. Because in the quantum setting, the code that we will consider is not necessarily the repetition code, the goal will be in general to retrieve a logical state that was initially corrupted, in the absence of further additional noise so that we can repeat the local update rule as any times as we want. This task is known as offline decoding.

Task Q1 (Offline decoding or 'Logical classification')**.** *Let ρ be an initial encoded quantum state and $\mathcal{N}$ an error channel. Given the corrupted state $\mathcal{N}(\rho)$, recover ρ by repeated homogeneous application of the local update rule, assuming no further noise occurs during decoding.*

In general quantum error correction phrasing, this corresponds to the *code-capacity* scenario where data qubits are initially affected by an error channel $\mathcal{N}$, corresponding, for example, to independent depolarizing noise on each data qubit, but with noiseless measurements. While the simplicity of this setting facilitates formal analysis, it does not reflect a realistic error model where measurements can be faulty and errors occur continuously over time with some constant probability. We now describe a more relevant task to realistically stabilizing a quantum memory, the one of online decoding. In the *phenomenological model*, measurements are faulty and new errors arise over time. In this scenario, fully restoring the state to the code space is impossible, so the focus shifts to analyzing the stability properties around a given initial encoded state.

Task Q2 (Online decoding or 'Quantum memory retention')**.** *Let ρ be an initial encoded quantum state, $\mathcal{N}$ an error channel. Evolve the state over time by interspersing homogeneous applications of the local update rule and error channel $\mathcal{N}$, and determine whether the evolved state remains close to the initial state despite the noise.*

We chose to remain vague here with the notion of being 'close to the initial

Decoder	Dim.	Threshold	Basic idea/principle
Toom's rule [146]	2D	7.7%	Localized error cluster are removed from their south-east corner
Gács [61, 60]	1D	exists but unknown	Hierarchical structure with constant-overhead self-simulation
Tsirelson [44, 10]	1D*	1.4%	Large error clusters are recursively split into smaller, locally erasable clusters
Harrington [74]	1D*	2.0%	Local implementation of a renormalization-group-like decoder
Field-based [77, 78]	1D*	✗	Local variables encode a classical field attracting defects to each other

Table 1.3: Local decoders of the repetition code. The label 1D* indicates that the 1D repetition code is equiped with logarithmic size memory. The value of the threshold is given for the online decoding setting under phenomenological noise, where for Harrington defined initially for the toric code we define and simulate a repetition code variant for a fair comparison. All decoders except Gács' construction are known to generalize to (or have been initially defined for) the toric code, at the cost of doubling the dimension, e.g. 4D toric code for a generalization of Toom's rule, and 2D toric code equipped with local logarithmic memories for 1D* automata.

state'. In what follows, this will correspond to whether a round of perfect measurements followed by global decoding recovers the initial logical information.

1.4.4 Notable examples of local decoders

Here, we highlight notable examples of local decoders that have been previously proposed in the litterature, that are listed and briefly compared in Table 1.3 for a 1D repetition code implementation.

We begin with a description of the first cellular automaton to demonstrate phase-transition-like behavior — known as non-ergodicity in cellular automaton terminology — the well-known Toom's rule that emulates a digital version of the 2D Ising model. Since our interest in local decoders is to circumvent the stringent requirements of self-correcting quantum memories, we then turn our attention to local decoders for the 1D repetition code and the toric code, both of greater practical relevance but not self-correcting. We will treat the 1D repetition code and the toric code together in the following discussion depending on the initial proposal, as they share a similar decoding problem due to their string-like logical operators, and all local decoders we describe have been

adapted for both, at the exception of field-based decoder that have not been defined and studied for the 1D repetition code. Listed proposals for quantum decoders fall into two main categories: automata with a hierarchical structure [10, 74, 34] inspired by classical constructions [44, 60], and field-based decoders [77, 78, 110] where defects are interpreted as particles interacting with each other through a classical field simulated by the classical automaton.

We now give a brief description of each proposal below, each of those can be read independently.

1.4.4.1 Toom's rule

Introduction. A well-known example of a local scheme that achieves reliable classical memory is Toom's rule [146], which stabilizes a classical bit encoded in a two-dimensional repetition code. The construction builds on the idea that in 2D the boundary of a droplet of errors forms a loop, whose curvature can be learned from local rules.

Setup. Toom's rule acts as a decoder of the 2D repetition code illustrated in Figure 1.11 (a). Consider the 2D periodic lattice $\mathcal{L} = \mathbb{Z}_\ell \times \mathbb{Z}_\ell$ for some integer ℓ. The n-qubit 2D quantum repetition code ($n = \ell \times \ell$) is defined from assigning qubits vertices and Z-type stabilizers (parity checks) to edges (note that we adopt the opposite convention here compared to the rest of the thesis).

For a given site $i, j < \ell$, the Z stabilizers associated with horizontal and vertical edges are respectively of the form

$$S^h_{i,j} := Z_{i,j} Z_{i+1,j}, \tag{1.55}$$

$$S^v_{i,j} := Z_{i,j} Z_{i,j+1}. \tag{1.56}$$

The set $\{S^h_{i,j}, S^v_{i,j}\}$ forms an overcomplete set of stabilizer generators for the quantum repetition code, that encodes a single logical qubit as $|\overline{k}\rangle = |k\rangle^{\otimes n}$ for $k \in \{0, 1\}$.

Update rule. Let $s^h_{i,j}$ (resp. $s^v_{i,j}$) denote the measurement outcome of the stabilizer $S^h_{i,j}$ (resp. $S^v_{i,j}$). Toom's rule applies a local Pauli correction $X_{i,j}$ whenever both parities are odd.

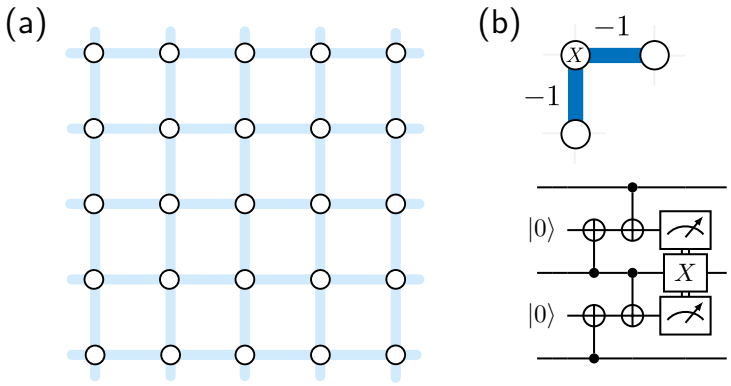

Figure 1.11: (a) Representation of the 2D quantum repetition code. Qubits are assigned to vertices and Z stabilizers are assigned to edges. (b) Each qubit is flipped when in odd parity with its southern and eastern neighbors. This can be implemented by computing parities on two ancillary qubits using a pair of CNOT gates, measuring them in the computational basis, and applying a Pauli X correction whenever both measurement outcomes are -1.

$$f(s^h_{i,j}, s^v_{i,j}) = \begin{cases} X & \text{if } (s^h_{i,j}, s^v_{i,j}) = (-1,-1), \\ I & \text{otherwise.} \end{cases} \tag{1.57}$$

Note that contrary to the more general setting described earlier, Toom's rule does not require other classical memories than required to store one round of measurement outcomes.

Properties. Toom's rule is known to have a threshold in both offline and online decoding setting. In the offline decoding setting, this is because any contiguous error cluster confined within a $\Delta \times \Delta$ box (for $\Delta < \ell$) can be completely erased after 2Δ successive applications of Toom's rule, as illustrated in Figure 1.12. More remarkable, however, is the particularly involved proof of online decoding threshold [147], in which case the memory lifetime scales exponentially in $\sqrt{n}$, similarly to the 2D Ising model. A simple explanation, although not relevant for practical performances as we shall see in Chapter 2, is that columns and rows are stable states of size $\sqrt{n}$ and that once one of those states are created it can easily propagate to the entire lattice with a $\Theta(\sqrt{n})$ additional error.

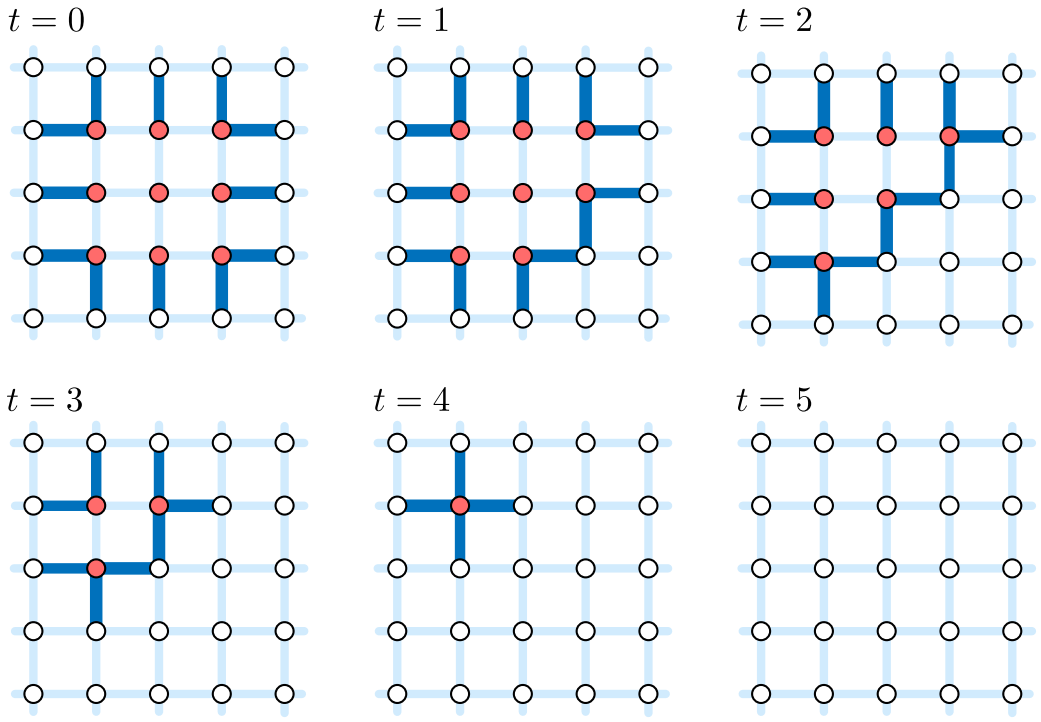

Figure 1.12: Example of the erasure of a 3×3 error cluster where $+1$ and -1 parities are respectively represented in sky and dark blue. Each iteration eliminates errors along the right-most southwest-northeast diagonal of the cluster, while faults do not propagate outside of the initial square box.

1.4.4.2 Tsirelson decoder

Introduction. Tsirelson's automaton was originally designed to protect a single bit of information encoded in a classical 1D repetition code [44]. Here we will focus on describing the classical case for the sake of simplicity, but the construction can be easily adapted to the 1D quantum repetition code, and with some more work to the 2D toric code [10].

The central idea behind Tsirelson's construction is that approximate decoding of a 3^{k+1}-bit repetition code can be achieved locally, provided that one can already decode a 3^k-bit repetition code. When global classical computation is available, this approach is equivalent to decoding a concatenated repetition code. Performing majority votes on the three blocks of 3^k bits will either result in all blocks agreeing on the same bit value, or in one block differing from the other two, for example 000111000 for $m = 2$. In the latter case, further subdividing each block into three and performing three 'transversal' majority votes across the three blocks (all three being 010 in the example) ensures that the automaton returns to a uniform configuration of either all 0s or all 1s.

At first sight, the main caveat is that to implement this procedure locally, one would need to physically swap bits around. Completing this step requires time $\mathcal{O}(3^{k+1})$, during which errors can accumulate. However, this does not prevent the scheme from having a threshold. This is because in this scenario, since majority votes are performed in parallel, k-level errors (error block outcome of a level k majority vote) accumulate only linearly in time (exponentially in k), while the error suppression is exponential in the code distance (and thus doubly exponential in k). In the following, we sketch how Tsirelson's construction implements those ideas, we adopt the definition introduced in [10], with slight modifications to the notations for consistency within this thesis.

Setup. Consider the 1D periodic lattice of length $n = 3^m$ for some integer m. The state of the automaton is described by a binary vector $u = (u_1, u_2, ..., u_n)$. Tsirelson's automaton is generated from three elementary gadgets, an *idle* gate X_0, a *swap* gate Y_0 and a *majority vote* gadget Z_0 defined as follows

$$X_0(u_1) = u_1, \qquad Y_0(u_1, u_2) = (u_2, u_1) \tag{1.58}$$

and

$$Z_0(u_1, u_2, u_3) = (\mathrm{Maj}(u_1, u_2, u_3),\ \mathrm{Maj}(u_1, u_2, u_3),\ \mathrm{Maj}(u_1, u_2, u_3)), \qquad (1.59)$$

and are represented below:

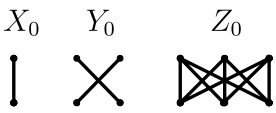

Figure 1.13: Figure taken from [10].

Thus X_0 acts as the identity, Y_0 swaps its inputs, and Z_0 performs a majority vote of its inputs and broadcasts three copies of the result. Note that, since all operations can easily be performed from parity measurements outcome, this description is compatible with the quantum setting. k-level gadgets can then be recursively generated from $(k-1)$-level gadgets, via the substitution rules.

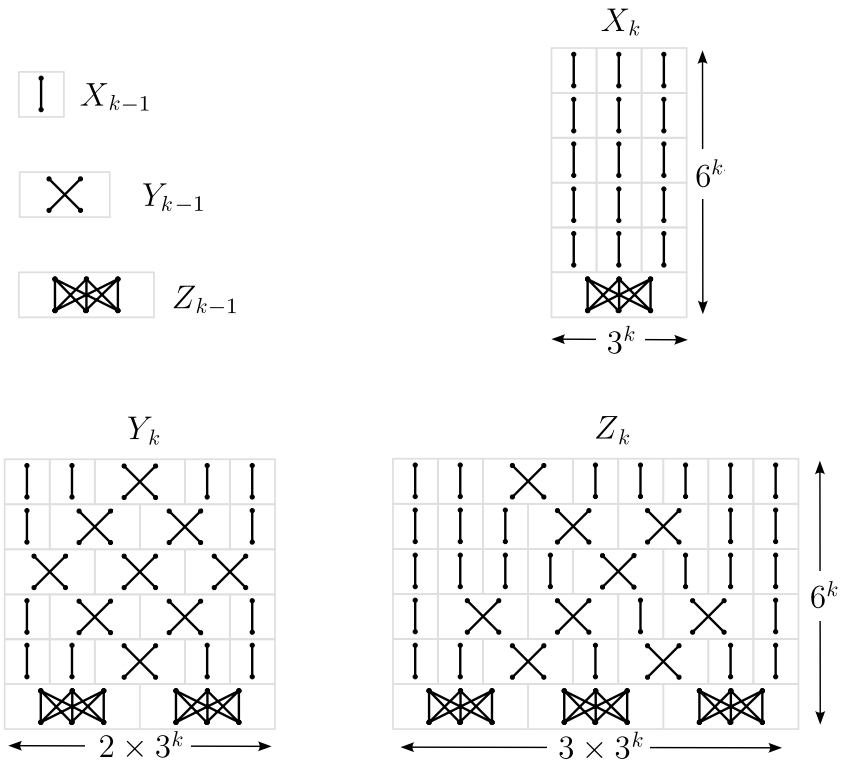

Figure 1.14: Figure taken from [10]. Time goes from the bottom to top. Note that because gadgets are repeated in time we do not need to add a layer of majority votes at the end of e.g. the definition of Z_k.

The k-level idle gate is constructed by performing a $(k-1)$-level majority

vote followed by idling. The k-level swap gate is realized through two simultaneous $(k-1)$-level majority votes followed by a transversal swap. Finally, the k-level majority vote is implemented using three parallel $(k-1)$-level majority votes, followed by a bit permutation that interleaves the initial blocks of three bits.

Definition. A version of Tsirelson automaton can then simply be defined as the repeated application of gadget X_m.

Properties. Tsirelson's automaton is known to have a threshold in both offline and online decoding settings, in the latter case with memory lifetime scaling exponentially in n^α for some $\alpha \in (0,1)$ [44]. For the slightly modified version introduced in [10], the same statement holds with $\alpha = \log_3 2 \simeq 0.63$, notably achieving better scaling than Toom's rule, despite operating in a lower-dimensional setting.

1.4.4.3 Harrington decoder

Introduction. In his PhD thesis, Harrington presented a set of local rules that handle the error syndrome processing of Kitaev's toric code building on ideas from Gács. The decoder takes inspiration from renormalization-group decoders exploiting the scale invariance of the toric code by decomposing the decoding problem into a hierarchy of smaller subproblems. In the following we give a sketch of a simplified version of Harrington's construction, where we also rename some variables for consistency within this thesis.

Setup. Consider the 2D periodic lattice $\mathcal{L} = \mathbb{Z}_d \times \mathbb{Z}_d$ with $d = L^m$ for some integers L and m. The toric code is defined by assigning qubits to edges, Z-type stabilizers to vertices and X-type stabilizers to plaquettes. Since bit-flip and phase-flip errors are decoded independently, we focus on bit-flip decoding; phase-flip decoding follows by symmetry. The error syndrome is point-like, and we refer to -1 parities as defects, which the decoding process aims to match by pairs. For further details on Kitaev's toric code, we refer the reader to the definition given in Subsection 1.3.2, illustrated in Figure 1.5.

A classical processor is placed at each vertex of the lattice, is able to communicate with its neighbours indicated by cardinal directions $\mathtt{X} \in \{\mathtt{C}, \mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$,

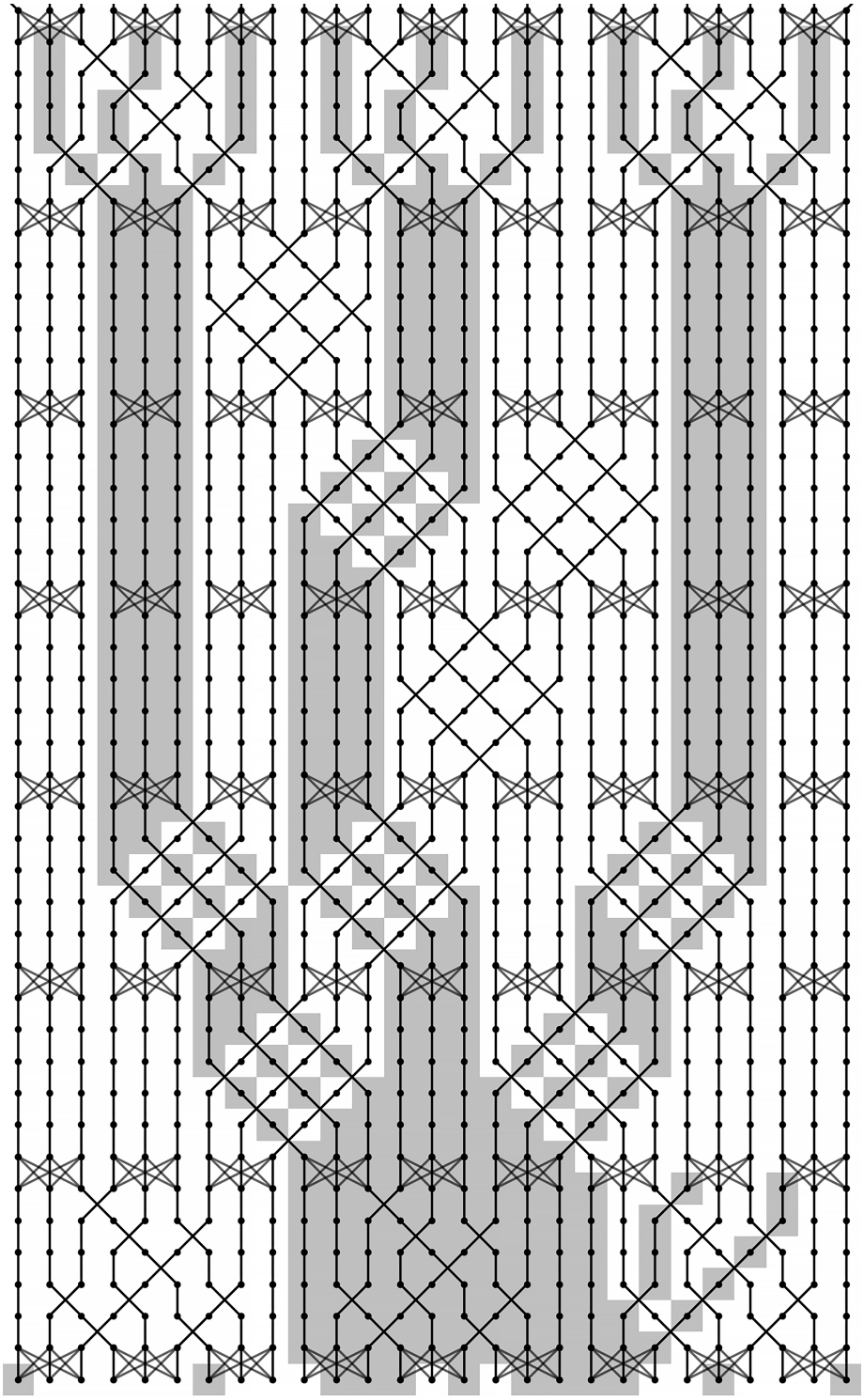

Figure 1.15: Figure taken from [10]. The evolution of an input state with randomly chosen initial error. Time goes from bottom to top and errors are shown in grey. The initial error is confined within a 2-level block resulting from the initial majority votes. This block is subsequently divided into three 1-level blocks, each of which is further subdivided into three 0-level blocks — i.e., individual errors that can then be corrected locally.

Field name	Size	Description
`Address`	$2\lceil m \log L \rceil$	Horizontal and vertical address
`Age`	$\lceil m \log \tau_f \rceil$	Local clock running from 0 to $(\tau_f^m - 1)$
`Def`	1	Local parity check outcome
`DefCount.I`	$\lceil k \log \tau_i \rceil$	Past defect historic
`DefCount.F`	$\lceil k \log \tau_i \rceil$	Coarse-grained count of past defect historic
`MacroDef`	1	Average presence of a defect on k-colony center

Table 1.4: Processor memory variables for local error correction of toric codes, excluding temporary variables used for copying local information. Each k-colony center at level k of the hierarchy is assigned a set of the final three k-dependent variables.

and is responsible for measuring the corresponding Z-type parity check and execute eventual bit-flip correction on linked edges. The processors that we call sites are then organized following a hierarchical structure. Cells are grouped into $L \times L$ colonies, which in turn form $L \times L$ super-colonies, continuing this hierarchy until it encompasses the entire lattice. At each level k of the hierarchy, cells may serve different roles and are divided into three categories: : those at k-colony centers (with `Address` $\equiv 0 \pmod{L^k}$), those positioned directly in between adjacent k-colony centers, and all others.

All cells are equipped with variables `Address`, `Age` and `Def` accounting respectively for the horizontal and vertical cordinates, the simulation time modulo some τ_f^m and the presence or not of a defect. For each upper level k of the hierarchy ($k \geq 2$) k-colony centers are equiped with a set of variables `DefCount.I`, `DefCount.F` and `MacroDef`. The average presence of a defect on a colony center is stored in `MacroDef` and determined from a coarse-grained count based on variables `DefCount.I` and `DefCount.F`, where `I` and `F` denotes two levels of averaging, for some threshold parameter $r \in (0, 1)$. More precisely, the coarse-grained count is determined to be positive if for τ_i^k successive blocks of τ_i^k iterations, the defect was present at least r times at a rate at least r. `MacroDef` is then used to perform correction in between k-colony center with defects. Initialization of the device involves setting all `Address` variables to each cell's relative position within its colony and all other variables to zero.

Update rule. The decoder operates by attempting to pair defects within the same colony. Isolated defects that cannot be matched locally are displaced to colony centers, and escalated to higher levels of the hierarchy, super-colonies

and so on. To achieve this, the local classical variables at each hierarchical level are synchronously updated at every time step across all sites, following a set of update rules. We now present the update rules separately for levels 1 and $k \geq 2$ but that are implemented in parallel at all times.

Algorithm 1: Harrington level-1 update rule

Input: `Address`, `Age`, `Def`

Output: `Address`, `Age`, `Def`

Update clock-time;
Increment `Age` by 1;
Syndrome extraction and local defect handling;
Set `Def.X` to local parity check outcomes for $\texttt{X} \in \{\texttt{C}, \texttt{N}, \texttt{E}, \texttt{S}, \texttt{W}\}$;
Merge adjacent defects, displace isolated defects by 1 toward closest colony center according to `Address`;

Cells at the centers of k-colonies are updated accordingly. We assume for now instantaneous classical communication between neighboring centers at the same level.

Algorithm 2: Harrington level-k update rule

Input: `Address`, `Age`, `Def` and k-level `DefCount` and `MacroDef`

Output: `Address`, `Age`, `Def` and k-level `DefCount` and `MacroDef`

Information from level 0;
if $\texttt{Address} \equiv 0 \pmod{L^k}$ **then**
    Increment `DefCount.I` if `Def` odd ;
Coarse-grained defect count;
if $\texttt{Age} \equiv 0 \pmod{\tau_i^k}$ **then**
    if $\texttt{DefCount.I} > r \cdot \tau_i^k$ **then**
        Increment `DefCount.F` by 1;
    Reset `DefCount.I` to 0;
k-level correction;
if $\texttt{Age} \equiv 0 \pmod{\tau_f^k}$ **then**
    if $\texttt{DefCount.F} > r \cdot \tau_f^k$ **then**
        Set `MacroDef` to 1;
        Merge adjacent macro-defects, displace isolated macro-defects by 1 toward closest $(k+1)$-colony center according to `Address`;

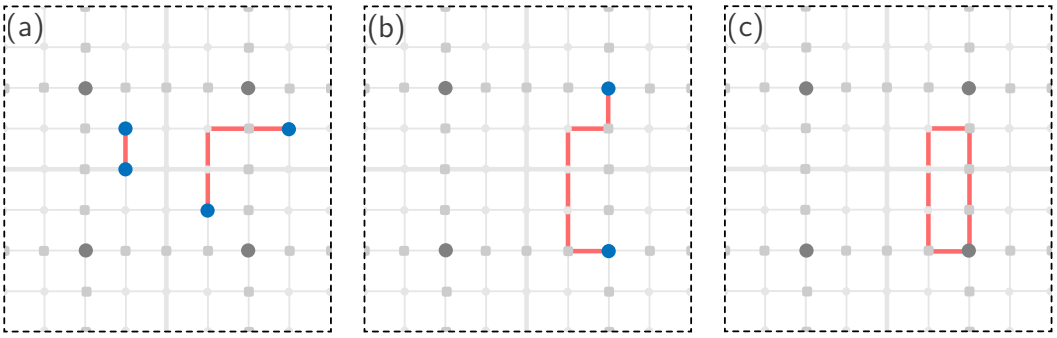

Figure 1.16: Illustration of Harrington decoder for $L = 4$ and $m = 2$ zoomed in on a part of the lattice (the entire lattice is of dimension 16×16), with initial condition (a) a set of erroneous edges represented in red with pairs of defects at the extremity. (b) Neighbouring defects are matched instantaneously, while isolated defects are moved toward the nearest colony center (shown in dark grey). At the end of the work period τ_f^2, each 2-colony center evaluates the average local defect presence and communicates with neighboring 2-colony centers via unrepresented additional variables on grey squares. (c) In the example shown, the cumulated error and correction is a contractible loop of the torus and as such a X-type stabilizer of the toric code.

Although the portion highlighted in yellow requires non-local communication between neighbouring colony centers from the same level, it is clear that this step can be executed locally, at the cost of a delay of $\mathcal{O}(L^k)$, by employing additional variables on sites directly in between colony centers. Further details on how this can be achieved are provided in [74].

Although this was not proven in the initial construction, the underlying intuition on why this delay is not detrimental is that, due to the influence of lower levels in the hierarchy, a defect that reaches a k-colony center is exponentially unlikely to escape. Therefore, even with a linear delay, $(k + 1)$-level correction is effective.

Properties. The decoder is proven to have a threshold in the offline setting [74], and in the online setting if instantaneous communication between colony center is available [47]. In addition, numerical evidence suggests that the decoder exhibits a threshold in the online setting, even when limited to local communication, with memory lifetime scaling exponentially in n^α for some $\alpha \in (0, 1)$.

1.4.4.4 Field-based decoders

Introduction. Field-based decoders were first introduced by Herold et al. [77, 78] (see also [94] for a modern variant) as a method for solving matching problems using only local information. For codes with point-like syndrome structures — such as the repetition code or the toric code — the decoding task reduces to pairing up defects (i.e., odd syndrome sites) based on some notion of proximity. While MWPM seeks the most-likely error consistent with the syndrome, it is inherently non-local. In contrast, field-based decoders aim to find sufficiently good approximate solutions by ensuring that defects close from each other recombine. The construction of [77, 78] achieves this locally by mediating an attractive interaction between defects via the propagation of a classical field represented by the automaton.

Setup. Consider the toric code setup described in the first paragraph of Harrington's decoder setup.

A classical processor is placed at each vertex of the lattice, and is responsible for measuring the corresponding Z-type parity check and execute eventual bit-flip correction on linked edges. Each vertex $v \in \mathcal{L}$ is assigned a pair of variables: the charge binary variable $q(v)$ takes value 1 in the presence of a defect, and 0 otherwise, while the fields value is stored in the real variable $\phi(v)$. A configuration of the automaton is then characterized by vectors q and ϕ. The goal of the update rule is for ϕ to represent a Coulomb-like potential field generated by the charges indicated in q, such that local corrections move defects toward neighboring sites with the highest ϕ value. The Coulomb potential is solution of Poisson equation,

$$\Delta\phi(v) = q(v). \tag{1.60}$$

Equation (1.60) can be discretized into a linear system of equation, whose resolution with the Jacobi method gives the field update rule, so that the field update rule is performed from adding the local average of the field to the charge,

$$\phi^{(t+1)}(v) = \underset{\langle\tilde{v},v\rangle}{\text{avg}}(\phi^{(t)}(\tilde{v})) + q(v), \tag{1.61}$$

where the average is made over $\tilde{v}$. And a defect update rule such that a defect is displaced to neighboring site with the highest ϕ value with probability 1/2.

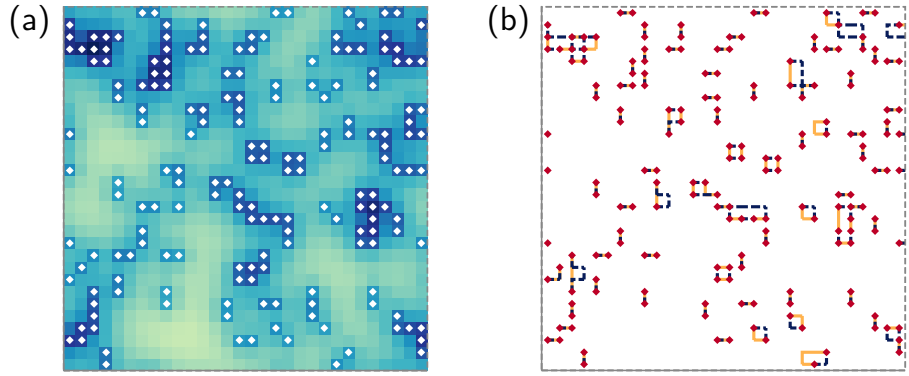

Figure 1.17: Taken from [77]. (a) The field, encoded in the cellular automaton, generated by defects of the toric code. Regions with a larger density of defects have a larger field value. The cellular automaton performs two tasks: (*i*) it updates the field in order to propagate information between distant defects, and (*ii*) it moves the defects in the direction of largest field gradient. (b) Logical information associated with one decoding run. The red diamonds are the ends of error strings, the orange lines are the actual physical error lines, and the blue dotted lines are the recovery paths dictated by the decoder.

Definition. The sequence of the automaton is defined as the repeated application of c field updates, for some positive integer c, that we will refer to as the field velocity, followed by one round of defects update.

Properties. Numerical evidence shows that field-based decoders exhibit a threshold in the offline decoding setting, at the cost of either a field velocity that increases linearly with the duration of the simulation, or the introduction of an additional spatial dimension for the field variable, with the field velocity then increasing only logarithmically with system size [77]. In the case of an additional dimension, and increasing field velocity, numerical evidence suggests that the scheme retains a threshold in the online setting [78]. Note that this increase of the field-velocity is necessary to avoid defects from interacting with themselves, a phenomenon refered to as self-interaction in [77].

1.4.5 Bridging the gap with global decoders

Let us now briefly comment on the decoder performances. In general, hierarchical constructions [10, 74, 34] works in both offline and online decoding setting, and protect information for a time $\exp(\gamma_d)$ with $\gamma_d \propto d^\alpha$ for $\alpha > 0$, but suffer from a low-error threshold for ε_d and ε_m, and from a poor *effective distance* γ_d,

corresponding to the minimal weight of error configuration leading to a logical failure. The effective distance therefore quantifies how well the local decoder performs compared to a global decoder (e.g. MWPM) that decode up to the distance of the code, that is $d = n$ for the repetition code, and $d = \sqrt{n}$ for the toric code. Field-based decoders, on the other hand, have been initially defined in the offline setting [77, 94] where they display high thresholds and good performance for a small system size. In the online setting however [78, 110, 70], the memory lifetime saturates above a certain system size, unless one keeps on increasing the communication speed. While these decoders are typically designed for the surface code, they are either directly inspired from a construction for the repetition code in 1D, or can be easily adapted to that case, with a similar expected qualitative behavior.

Overall, known constructions exhibit too far from optimal performances especially in the relevant online setting, in terms of threshold and scaling with system size, to represent a credible alternative to global decoders. This explains why interest in local decoders has so far mainly remained theoretical.

In Chapter 2, we examine the limitations of Toom's rule and propose two approaches to achieve improved performance. The first introduces a degree of long-range connectivity by shuffling qubits across the lattice, which yields a dramatic improvement when qubit swaps can be performed at low cost. The second compresses the 2D grid into two qubit layers, enabling the design of an automaton that performs well over a practical range of parameters while reducing implementation overhead. However, this latter approach lacks a threshold in the online decoding regime — a limitation we address by incorporating classical memories. This leads to a new family of local decoders, based on the exchange of point-like signals between point-like defects, described in Chapter 3, which achieve thresholds in both online and offline decoding and deliver state-of-the-art performance.

1.5 Quantum advantage before universal fault-tolerance

1.5.1 The computational power of nature

As information is necessarily carried by a physical system, its manipulation obeys the laws of the physical world. In particular, whether a problem is hard or not, should depend on a set of relevant physical laws. This raises a fundamental question: does our classical model of computation fully capture the computational capabilities of all physically realizable machines? The initial belief in the universality of classical computation is captured by the *strong Church–Turing thesis*.

The strong Church-Turing thesis — in its extended form, accounting for randomized computation — conjectures that a problem that can be solved efficiently by some physical device can also be solved efficiently on a Turing machine. In other words: every physically realizable computation can be performed efficiently on a classical computer [150, 14]. In particular, this implies that it should be impossible to construct machines that achieve exponential advantage over classical computers. The belief at the core of the development of quantum computing is that, machines based on the laws of a more fundamental theory of nature than the one of classical physics, e.g. based on the laws of quantum mechanics, can violate the strong Church-Turing thesis. In this sense, an experimental demonstration of an exponential advantage of quantum machines over their classical counterparts — known as quantum supremacy [124] — would indicate that there exists something beyond classical computation, and that quantum mechanics enables fundamentally richer computational processes than classical physics. Such a result would mark a major milestone in both worlds of computer science and physics.

A good candidate problem to demonstrate quantum supremacy should be (*i*) as simple as possible so that it can be solved by a near-term quantum machine, classicaly hard (*ii*) asymptotically[8] and (*iii*) in practice, meaning it cannot be solved by any polynomial-time classical algorithm and remains intractable even for brute-force classical methods. Finally, it should also (*iv*) be efficiently verifiable, in the sense that there exists a test to confirm that the task was suc-

[8] typically conditioned on complexity-theoretic conjectures, which should be minimal in number and highly plausible.

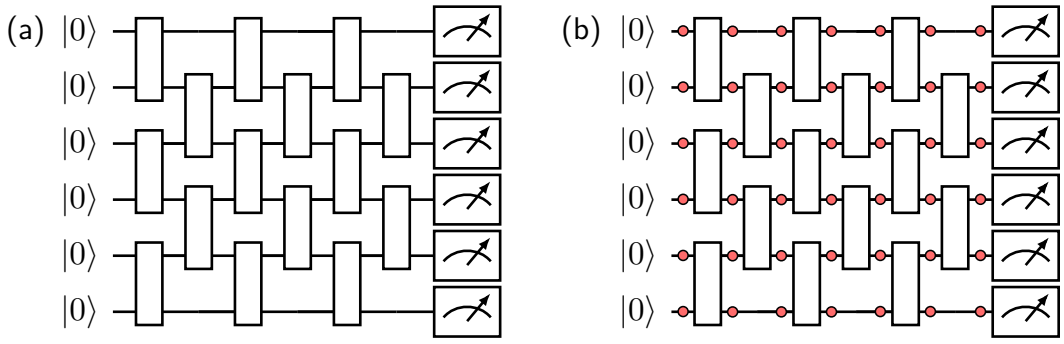

Figure 1.18: Adapted from [4]. (a) A 1D brickwork circuit structure, where random circuits are generated by independently sampling 2-qubit gates from the Haar measure. (b) A realistic noise model should at a minimum include a noise channel (e.g., i.i.d. depolarizing noise) at each space-time location of the circuit. The computational task for a classical computer to compare with the quantum machine is thus reduced from exact sampling to the possibly easier approximate sampling.

cessfully and unambiguously completed. The task of factoring integers directly satisfies (*ii-iv*), however, despite huge progress on every layer of the stack from algorithms to fault-tolerant implementation [63, 62], Shor algorithm remains out of reach of near-term machines.

In the meantime, sampling problems [106, 114, 73] — illustrated in Figure 1.18, where the task is to output sample according to some given probability distribution — appeared to be a promising avenue to demonstrate such a quantum advantage since they can be solved with reasonably small circuits. This is because, in principle, any quantum computation can be seen as a sampling problem on the outcome probabilities. More formally a random circuit sampling task can be defined as follows

Task 3 (Quantum random circuit-sampling)**.** *Given as input a problem size N and a circuit C chosen at random from a family $\mathfrak{C}_N$, sample from the output distribution p_C of the circuit applied to a reference state $|0^N\rangle$, with the probability of an outcome $s \in \{0,1\}^N$ given by $p_C(s) = |\langle s|C|0^N\rangle|^2$.*

Well-known examples of circuit families include linear optical circuits in the case of Boson Sampling [2], random quantum circuits [26] and Instantaneous Quantum Polynomial-time (IQP) circuits [137]. Classical hardness evidence are discussed in more detail below and follows from relating the hardness of comput-

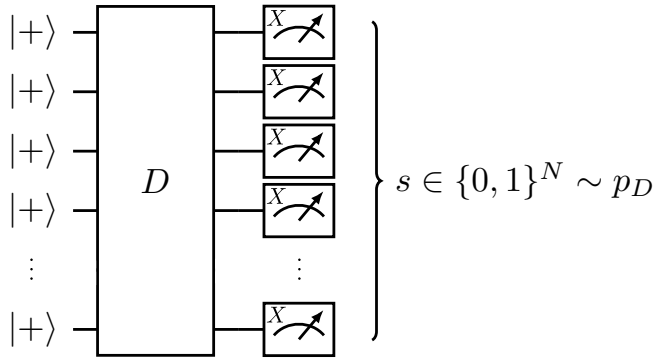

Figure 1.19: IQP circuits on N qubits are defined by a unitary D diagonal in the computational basis with state preparation and measurements performed in the Hadamard basis. For sparse IQP circuits, D is a logarithmic-depth circuit consisting of T and CS-gates.

ing outcome probabilities to the hardness of sampling[9] through Stockmeyer's algorithm [143]. Random circuit families are then designed such that their outcome probabilities correspond to known problems believed to be computationaly hard, such as computing the permanent of a matrix in Boson Sampling or the partition function of a general Ising model in IQP circuits. The condition of exact sampling can be relaxed to approximate sampling up to an additive error if, in addition, probabilities are hard to approximate in the average-case and the scheme presents some hiding property so that the output bitstring does not reveal useful information about the internal structure of the circuit. We now present evidence of computational hardness for the specific case of IQP sampling.

1.5.2 Evidence for classical hardness: the IQP example

A prominent family of random quantum sampling schemes based on restricted gate sets is defined by the so-called IQP (Instantaneous Quantum Polynomial-time) circuits [137]. An IQP circuit (see Figure 1.19) is a commuting quantum circuit that is diagonal in the Hadamard basis. Such a circuit can always be written in the form $C = H^{\otimes n} D H^{\otimes n}$, where D is a diagonal unitary operator in the computational basis. Quantum devices however are not expected to implement arbitrary unitaries directly, so to avoid compilation overhead in near-term experiments, circuits natively defined from a finite gate set are preferred.

[9] up to multiplicative error on probabilities

In a sparse variant of IQP circuits, introduced in [33], the circuit D is generated randomly from logarithmic-depth circuits with gate set $\{T, CS\}$:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \qquad CS = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}. \tag{1.62}$$

More precisely, such a circuit on N qubits is generated in the following way:

- a single-qubit gate T^k is applied to every qubit, with $k \in \{0, \ldots, 7\}$ chosen uniformly and independently for every qubit,
- for every pair of qubits, a gate CS^k with $k \in \{0, \ldots, 3\}$ chosen uniformly at random, is applied with probability $\gamma \log N/N$, for some fixed parameter $\gamma > 0$.

Let us denote by $\mathfrak{D}_N$ the family of IQP circuits generated from the gate set $\{T, CS\}$. We associate to each circuit of $\mathfrak{D}_N$ its probability of being generated by the previous random process to define a distribution over $\mathfrak{D}_N$. We call an IQP circuit picked from this distribution 'sparse' and in the following whenever we discuss about a fraction of sparse IQP circuits we mean a fraction of circuits in the sense of the probability distribution defined above.

We note that all the considered gates commute, and can therefore be applied in any order. Given that each qubit will typically be involved in a logarithmic number of 2-qubit gates, we see that sparse IQP circuits can be implemented by circuits of average depth $\Theta(\log N)$ [33]. For each sparse IQP circuit D, we denote by p_D the probability distribution on $\{0,1\}^N$ corresponding to the output distribution of the circuit. In particular, it holds that

$$p_D(0^N) = \sum_{z \in \{0,1\}^N} e^{i\pi/8\left(\sum_{i<j} w_{i,j} z_i z_j + \sum_{k=1}^N v_k z_k\right)}, \tag{1.63}$$

for some integer weights $w_{i,j}, v_k$. The key argument for hardness of (sparse) IQP sampling is that the hardness of sampling can be related to the hardness of computing probabilities, a connection we outline below. For a more detailed discussion of these hardness arguments, we refer the reader to [31, 32, 2, 73].

Hardness of exact sampling. For some families of quantum random circuits, output probabilities can be shown to be #P-hard to approximate in the worst-case. While the sole ability to sample is not sufficient to estimate probabilities in general, this is no longer the case when one has access to an NP oracle. Building on this extra computational power, Stockmeyer's approximate counting algorithm [143] achieves this task, allowing one to estimate probabilities to within any inverse-polynomial multiplicative error from the ability to sample from the distribution. Of course, access to an NP oracle does not correspond to any physically realistic model of computation, and it is not surprising that such access enables the solution of many otherwise hard problems. However, #P-hard (#P is the class of problems that count how many solutions exist for an NP problem) problems are believed to be strictly harder than those solvable with NP oracles. More formally, Stockmeyer's algorithm lies in the third level of the polynomial hierarchy — a hierarchy that extends the classes P and NP into increasingly powerful levels, and the approximation of #P-hard problems within the third level implies a collapse of the polynomial hierarchy to its third level, something that is believed to be highly unlikely by complexity theorists.

Hardness of approximate sampling. Available quantum hardware is however unlikely to be able to sample exactly from the target probability distribution, hence a robust hardness statement should allow some level of approximation, e.g. up to constant error in total variation distance. Proofs of approximate sampling hardness typically proceed by showing that, given access to an approximate sampler, one can estimate specific output probabilities. To enable this, it is necessary to ensure that the total variation distance between the approximate and ideal distributions is well-distributed across all outcomes, rather than concentrated on the specific output whose probability we aim to estimate. Thankfully, for IQP circuits, the outcome probability of a bit string $x \in \{0,1\}^N$ can be mapped to that of another bit string $y \in \{0,1\}^N$ by appending Pauli X gates $X^{x_1 \oplus y_1} \cdots X^{x_N \oplus y_N}$. As a result, applying X gates at random to the output qubits helps prevent errors from being always biased toward the particular outcome we are interested in. This hiding property, however, is a probabilistic statement with respect

to the probability distribution on circuits, so any hardness claim must rely on the assumption that output probabilities are hard to approximate in the average case. Proving the approximate average-case hardness of output probabilities remains an open problem and must therefore be conjectured in order to support such a claim.

In the case of IQP the quantity $p_D(0^N)$ corresponds to an Ising model partition function, which is proven to be hard to compute in the worst case[10] [64, 58, 89] and conjectured to be hard to compute on average. We formally recall the conjecture from [33]:

Conjecture 1 (Average Case Hardness of Ising model [33])**.** *Consider the partition function of the general Ising model,*

$$Z(\omega) = \sum_{z \in \{\pm 1\}^N} \omega^{\sum_{i<j} w_{i,j} z_i z_j + \sum_{k=1}^N v_k z_k}, \tag{1.64}$$

where the exponential sum is over the complete graph on N vertices, $w_{i,j} \in \mathbb{R}$ and $v_k \in \mathbb{R}$ are weights for edge ij and vertex k, and $\omega \in \mathbb{C}$.

If the weights are chosen uniformly at random from the set $\{0, \ldots, 7\}$, then it is #P-hard to approximate $\left|Z(e^{i\pi/8})\right|^2$ up to multiplicative error $1/4 + o(1)$ for a $1/24$ fraction of instances, over the random choice of weights.

The sparse IQP problem is as follows: pick a random $D \in \mathfrak{D}_N$ according to the random process described before, and output an N-bit string s according to a distribution q_D such that

$$\|p_D - q_D\|_{\mathrm{TV}} \leq \delta, \tag{1.65}$$

where the total variation distance between two distributions p and q is defined as

$$\|p - q\|_{\mathrm{TV}} := \frac{1}{2} \sum_{s \in \{0,1\}^N} |p(s) - q(s)|. \tag{1.66}$$

Assuming Conjecture 1, and the non-collapse of the Polynomial Hierarchy, a generalisation of the $P \neq NP$ conjecture widely considered to be true, Bremner

[10] Specifically, computing the partition function is #P-hard to approximate within an exponentially small multiplicative error.

et al. proved that there is no efficient classical algorithm for the sparse IQP problem. More precisely,

Theorem 6 (Classical hardness of sparse IQP sampling [33])**.** *Assuming Conjecture 1, there exists $\delta > 0$ independent of N such that a constant fraction of sparse IQP circuits cannot be simulated by a polynomial-time classical algorithm up to precision δ in total variation distance unless the polynomial hierarchy collapses to its third level.*

This theorem states that on average over the choice of D from the probability distribution over $\mathfrak{D}_N$ defined previously, it is hard to sample classically from a distribution close to p_D.

1.5.3 Losing classical hardness in the presence of noise

The caveat, however, is that current quantum processors are not equipped with fault-tolerance, and will instead output noisy samples, thus only solving a noisy version of the initial sampling problem. In fact, the initial proposal [33] partially addressed this issue by considering a simple noise model where the quantum circuit is assumed to be ideal, except for some independent and identically distributed noise added to the classical value of the final outcomes. Unfortunately, this model is too naive and a more realistic noise model should assume that every gate suffers from some constant level of noise ε as illustrated in Figure 1.18. In that case, because the number of gates for a typical random circuit is of order $N \log N$, it is immediate that noise will accumulate through the circuit and that the level of noise per qubit cannot be assumed to be constant, independent of N.

More formally, given as input the description of some circuit C with ideal outcome probability distribution p_C, a noisy quantum device will instead sample from a modified distribution $p_{C,\varepsilon}$, which deviates from the ideal due to the presence of noise. In general, there is no guarantee that $p_{C,\varepsilon}$ is close from p_C in terms of total variation distance, and hence, a fair comparison between the power of classical and quantum computation should be based on sampling from the noisy distribution $p_{C,\varepsilon}$ rather than the ideal distribution p_C. This could make the task easier, in particular this is the case if the target probability distribution *anti-concentrates*, meaning that it is sufficiently spread out over all

outcomes, i.e. $\sum_{s\in\{0,1\}^N} q(s)^2 = \mathcal{O}(1)\cdot 2^{-N}$. Under this condition[11], Aharonov et al. prove in a seminal work [4] that the associated noisy sampling task becomes classically tractable, in the sense that it can be solved by a polynomial-time classical algorithm.

Theorem 7 (Polynomial-time classical algorithm for noisy random-circuit sampling [4])**.** *Assuming anti-concentration, there exists a classical algorithm that, given a random circuit C on any fixed architecture, produces a sample from a distribution that is δ-close in total variation distance to the noisy output distribution $p_{C,\varepsilon}$, with constant success probability over the choice of C, in time* $\mathrm{poly}(n, 1/\delta)$.

Note that a direct corollary of the Theorem is that there does not exist an efficient statistical test that can distinguish between the output of the algorithm and the measurement outcomes of the experiment. To illustrate how noise may simplify the sampling task for a classical computer, we briefly outline the main idea of the proof: the decomposition of outcome probabilities in Pauli paths.

Pauli path We start by the general description of the noiseless unitary dynamics in terms of Pauli path. Let ρ an N-qubit density matrix, we write the decomposition of ρ in the Pauli basis

$$\rho = \sum_{P_0\in\mathcal{P}_N} \mathrm{Tr}(P_0\rho)\cdot P_0. \tag{1.67}$$

We are interested in how the coefficient changes upon unitary evolution $\rho \to U\rho U^\dagger$. The new component in $P_1 \in \mathcal{P}_N$ is

$$\mathrm{Tr}(P_1 U\rho U^\dagger) = \sum_{P_0\in\mathcal{P}_N} \mathrm{Tr}(P_1 U P_0 U^\dagger)\cdot \mathrm{Tr}(P_0\rho). \tag{1.68}$$

In general, for a quantum circuit $C = U_D U_{D-1}...U_1$ of depth D where U_i is layer of 2-qubits gates, the Feynman path integral in the Pauli basis is written as sum of products of transition amplitudes,

[11] Which has been proven to hold for some random circuit families, including 1D random circuits in at least logarithmic depth [11, 46] and IQP circuits [32, 33].

$$p_C(s) = \sum_{P_0,...P_D \in \mathcal{P}_N} \mathrm{Tr}(|s\rangle\langle s|\, P_D) \cdot \mathrm{Tr}(P_D U_D P_{D-1} U_D^\dagger) \cdots \tag{1.69}$$

$$\mathrm{Tr}(P_1 U_1 P_0 U_1^\dagger) \cdot \mathrm{Tr}(P_0\, |0^N\rangle\langle 0^N|). \tag{1.70}$$

Let $f(C, P, s)$ be the contribution of the Pauli path $P = (P_0, P_1, ..., P_D) \in \mathcal{P}_N^{D+1}$ to outcome probability $p_C(s)$, the expression can be rewritten as

$$p_C(s) = \sum_{P \in \mathcal{P}_N^{D+1}} f(C, P, s). \tag{1.71}$$

The utility of this representation becomes clear in the presence of noise. The single-qubit depolarizing channel can be written as $\mathcal{N}(\rho) = (1 - \frac{3\varepsilon}{4})\rho + \frac{3\varepsilon}{8} I \cdot \mathrm{Tr}(\rho)$. This noise channel has a special property of being diagonalizable in the Pauli basis, and in particular

$$\mathcal{N}(I) = I, \;\; \mathcal{N}(P) = (1 - \frac{3\varepsilon}{4})P, \; \forall P \in \{X, Y, Z\}. \tag{1.72}$$

This directly implies that the contribution of a Pauli path to the noisy probability distribution is exponentially suppressed with its Hamming weight

$$p_{C,\varepsilon}(s) = \sum_{P \in \mathcal{P}_N^{D+1}} (1 - \frac{3\varepsilon}{4})^{|P|} f(C, P, s). \tag{1.73}$$

The objective of the proof is to approximate $p_{C,\varepsilon}(s)$ by truncating the sum in (1.71) to the first k terms, for some small enough k, and proving that this approximation is close enough to the noisy distribution in total variation distance. This implies that the algorithms needs to keep track of an only polynomial number of terms, so that the problem becomes tractable. We do not elaborate further here; the full proof can be found in [4].

1.5.4 Towards early fault-tolerant quantum advantage

In the presence of noise, the evidence for classical hardness of sampling problems vanishes without quantum error-correction and fault-tolerance. Recent experimental advances, particularly on atom array platforms [16, 131], however suggests that quantum devices may soon be capable of limited fault-tolerant quantum computation. This motivates the use of some form of error reduction

or correction techniques, prior to the advent of universal fault-tolerant quantum computing—in what we refer to as the *early fault-tolerant* regime.

The computational power of machines operating in this regime [126] remains uncertain. Nonetheless, early fault-tolerant quantum computers will undoubtedly exploit hardware-specific features to enhance performance for specific algorithms. For instance, atom array experiments [16, 131] offer two key advantages: (i) enhanced connectivity enabled by the reconfigurability of the array, which facilitates architectures based on transversal gates, and (ii) improved scalability in qubit number thanks to a high-level of multiplexing. Given that one of the main limitations of these experiments is the restricted number of repeated stabilizer measurement rounds, a natural early fault-tolerant strategy is to implement a highly parallelizable computation using transversal gates. The fact that IQP circuits are a non-universal class of circuits make them a good candidate in this respect since they are easier to make fault-tolerant. In particular, they can bypass the limitations of the Eastin-Knill theorem which states that a universal gate set cannot be implemented with transversal gates [52] (see Subsection 1.3.3). In Chapter 4, we present a construction tailored to the native capabilities of atom array experiments, based on the parallel fault-tolerant implementation of sparse IQP circuits.

1.6 Outline of the manuscript

The primary goal of this manuscript is to explore how fault-tolerant architectures can be adapted to the practical constraints of modern quantum platforms. In Chapters 2 and 3, we investigate local quantum memories from local decoders, offering a promising route toward real-time quantum error correction, which remains one of the major challenges for superconducting architectures with fast clock cycles. In Chapter 2, we analyze the limitations of Toom's rule and demonstrate that the two-dimensional decoding grid can be compressed into just two rows of bits, while still achieving improved performance at low error rates and modest system sizes — making it an attractive short-term solution. Building on this, Chapter 3 shows that by equipping stabilizer sites with a small amount of classical memory, the logical lifetime can be extended arbitrarily by increasing the system size, while maintaining good performance. In parallel, we observe that atom array experiments offer an opportunity to implement large-scale algorithms with a more flexible connectivity. In Chapter 4,

we propose leveraging these features to demonstrate quantum advantage in the early fault-tolerant regime. Finally, Chapter 5 provides concluding remarks and future perspectives on this work.

Chapter 2

Protecting a bit of information with local majority votes

This chapter covers unpublished work and part of work published in [119].

Contents

2.1 Introduction

Topological codes protect quantum information by enforcing local constraints on qubits arranged on a surface [85], easing experimental implementation. The locality of constraints alone however doesn't guarantee a fully local memory if decoding is global (i.e. centralized), which comes at the cost of an additional space-time overhead to handle measurement errors, along with a more complex

experimental architecture. Local decoders offer a promising alternative by replacing centralized decoding with a parallel, streamlined architecture based on simple local rules, reducing hardware constraints.

A simple example of a local decoder in the classical setting is Toom's rule [147], that protects a bit of information in a 2D lattice through the repeated application of a simple local update rule emulating a digital version of the Ising model. Despite its theoretical appeal, its error-correction capacity is limited by the tendency of faults to aggregate into clusters, leading to suboptimal performances. To mitigate this, we propose in this chapter to interleave the standard Toom update with an alternative shuffling routine designed to disrupt fault clustering. If swapping qubits is accessible at low cost, this suffices to drastically improve performances. Remarkably, we also find that with a well-designed shuffling strategy, the two-dimensional grid can be compressed into just two rows of bits, while still achieving improved performance for modest system sizes, making it an appealing short-term solution. Combined with biased noise qubits, like cat qubits, which simplify the problem of quantum error-correction to classical error-correction, those constructions lead to a fully local quantum memory.

2.1.1 Limitations of Toom's rule

The limitations of Toom's rule are twofold. First, the automaton is defined on a two-dimensional periodic lattice, while state-of-the-art superconducting hardware requires two-dimensional nearest-neighbor connectivity. Implementing a quantum memory already necessitates lifting the periodic boundary conditions, but this alone is insufficient if the goal is to later implement logical operations. This is because code blocks interact only locally at their boundaries, which prevents the implementation of transversal gates—unless the code blocks can be stacked in a two-layer, two-dimensional architecture. This makes the usual embedding of the quantum repetition code on a one-dimensional lattice preferable.

The second limitation of Toom's rule concerns its performance. A logical bit-flip of the n-qubit quantum repetition code corresponds to flipping all qubits individually; we say that the bit-flip distance is n. Optimal (albeit non-local) decoders [53, 49] exist that can correct all errors of weight up to $\frac{n-1}{2}$. In this case, there exists a threshold below which the logical bit-flip rate scales as

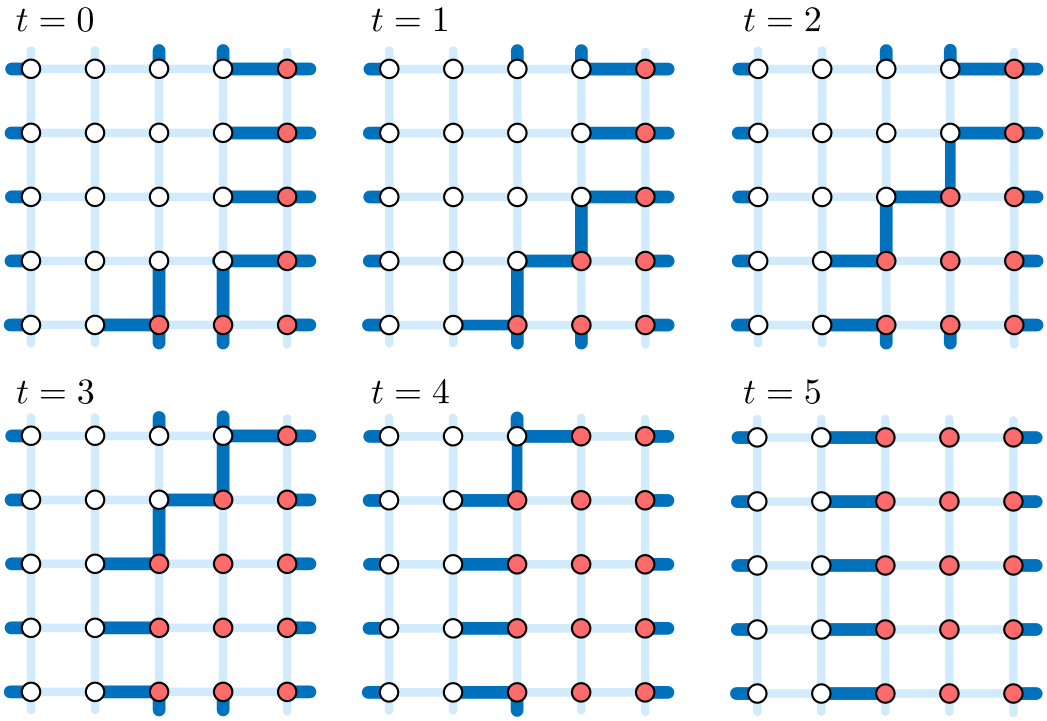

Figure 2.1: Example of an intial error configuration of weight $\mathcal{O}(\ell)$ in red leading to a fault cluster of weight $> n/2$, recall that $n = \ell^2$. Consequently, any subsequent global decoding would yield a logical error.

$$\mathbb{P}(\text{logical bit-flip}) \propto n(\varepsilon/\varepsilon_{\text{th}})^{\frac{n+1}{2}}. \tag{2.1}$$

On the other hand, when the quantum repetition code is embedded in a $\ell \times \ell$ two-dimensional lattice and decoded using Toom's rule, the logical error probability is asymptotically exponentially suppressed, but only in $\ell = \Theta(\sqrt{n})$. This scaling arises from the existence of error clusters of weight $\Theta(\ell)$ that can lead to logical failure. An example is illustrated in Figure 2.1: there, an initial error of weight $\ell + (1 + \ell)/2$ is propagated by ℓ successive applications of Toom's rule to the $(1+\ell)/2$ rightmost columns of the lattice, ultimately producing a fault of weight greater than $n/2$.

Because those pathological error configurations may be rare, and hence not necessarily the dominant source of errors depending on the choice of parameters, in the absence of numerical simulations it is difficult to assess whether they are of practical relevance. We numerically demonstrate in Figure 2.2 (b) that this asymptotic scaling is already relevant in a practical regime of parameters, by fitting the exponent of the logical error rate below threshold as a function of n. This behavior reflects the spatial correlation of faults within the automaton, arising from the fact that a cluster is gradually erased from its boundary. We refer to this phenomenon as *fault aggregation*, and propose mitigating it through local qubit permutations implemented via SWAP circuits.

2.1.2 Main results

We define a new family of local decoders on a square 2D surface i.e. without periodic boundary counditions. The update rule follows a periodic pattern of a Toom layer followed by layers of SWAP gates implemented in parallel. The latter layers aim at shuffling qubits around the lattice according to a pattern refered to as a SWAP network. The chosen SWAP network is detailed in Subsection 2.2.1 and the boundary conditions are fixed in Subsection 2.2.2.

The *k-local Toom's rule* (k-LTR) is defined in Section 2.2 from interspersing one layer of majority vote on a square lattice with k successive layers of SWAP gates forming a circuit of depth k. We compare the k-local Toom's rule with another variant where the SWAP gates circuit of depth k is replaced by a random permutation of all qubits, and that we call *global Toom's rule* (GTR). Note that a random permutation of qubits can be implemented in depth $\Theta(\sqrt{n})$ using e.g. sorting networks [135]. Interestingly, in a model with a noise strengh

independent from the number of the SWAP gates, we numerically show that despite being highly structured, when increasing the SWAP depth the former type of deterministic shuffling approaches rapidly the performances of global random shuffling (see Figure 2.11), for which we prove that the logical error rate is exponentially suppressed in n in the phenomenological model with data qubit flip probability $\varepsilon_d = \varepsilon > 0$ and without measurement errors ($\varepsilon_m = 0$).

Theorem 8 (Global shuffling threshold)**.** *There exist $\varepsilon_{\text{th}} > 0$ and $c > 0$ such that for $\varepsilon < \varepsilon_{\text{th}}$ the logical error probability of the GTR applied τ times in the phenomenological model with $\varepsilon_d = \varepsilon$ and $\varepsilon_m = 0$, satisfies $\overline{P}^{(\tau)} \leq \tau \cdot \exp(-cn)$.*

We show that $\varepsilon_{\text{th}} > 0.0001$ in the proof provided in the next section, but our numerical simulations (in the more general phenomenological model) indicate that this bound is likely to be far from tight. Note that unlike in standard threshold theorems, the constant c is independent of ε. This is due to the specific fact that randomness here arises not only from the noise but also from the global permutation. As a result, an adversarial permutation can amplify a constant number of errors into a logical failure. However, such permutations occur with exponentially small probability, which ensures that the scheme still exhibits a threshold.

Finally, in an alternative proposal, we also find that with a well-designed deterministic shuffling strategy, the two-dimensional grid can be compressed into just two rows of qubits, while remarkably achieving improved performance at low error rates and modest system sizes. This *shearing rule* is introduced in Section 2.3.

2.2 Breaking fault-aggregation on a square lattice

2.2.1 Shuffling qubits with SWAP networks

The k-local shuffling is defined from a periodic arrangement of elementary SWAP circuits, which we will refer to as the SWAP network. We consider a network illustrated in Figure 2.3 where horizontal and vertical SWAP gates are applied alternately between neighboring qubits[1]. This procedure is repeated

[1]We use the definition of Toom's rule presented in the introduction, where in contrast to alternative automata construction, qubits are vertices and parity checks are edges.

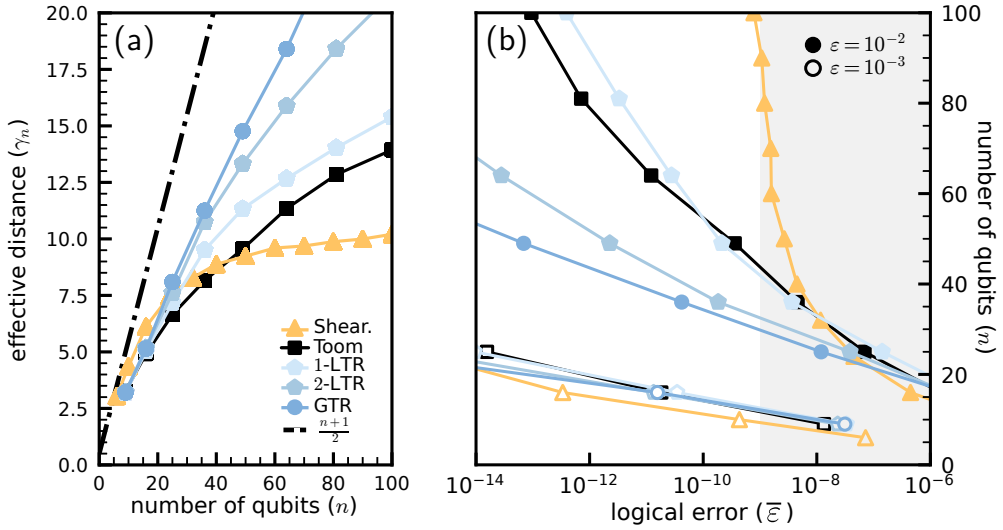

Figure 2.2: (a) Effective distance γ_n as a function of n for the Toom's rule, global Toom's rule and k-local Toom's rule for $k \leq 2$ in the phenomenological model with $\varepsilon_d = \varepsilon_m = \varepsilon$. The dashed line represents the scaling of the repetition code in $\frac{n+1}{2}$ when decoded using minimum weight perfect matching [53, 49]. (b) Estimate of the number of physical qubits necessary to reach a given logical error rate for each local decoder. The estimates are made from the ansatz $An(B\varepsilon)^{\gamma_n}$ with a different parameter γ_n for each n, fitted on the grey region, see Figure 2.11. The lines in both plots are for visual guidance only and do not represent actual data.

twice, with the second round offset by one qubit to target a different set of pairs. More formally, if we denote by $\mathrm{SWAP}((i_1, j_1), (i_2, j_2))$ the SWAP gate between qubits (i_1, j_1) and $(i_2, j_2) \in \mathcal{L}$ the different parallel SWAP layers that we consider are

$$P_0 = \prod_{\substack{i<\ell,\, j<\ell-1 \\ j\equiv 0 \,(\mathrm{mod}\ 2)}} \mathrm{SWAP}((i,j),(i,j+1)), \tag{2.2}$$

$$P_1 = \prod_{\substack{i<\ell-1,\, j<\ell \\ i\equiv 0 \,(\mathrm{mod}\ 2)}} \mathrm{SWAP}((i,j),(i+1,j)), \tag{2.3}$$

$$P_2 = \prod_{\substack{i<\ell,\, j<\ell-1 \\ j\equiv 1 \,(\mathrm{mod}\ 2)}} \mathrm{SWAP}((i,j),(i,j+1)), \tag{2.4}$$

$$P_3 = \prod_{\substack{i<\ell-1,\, j<\ell \\ i\equiv 1 \,(\mathrm{mod}\ 2)}} \mathrm{SWAP}((i,j),(i+1,j)). \tag{2.5}$$

Consider an automaton defined by successive application of parallel SWAP layers $P_{i(\mathrm{mod}\,4)}$ each followed by a subsequent noise layer, incrementing i by 1 at each iteration. The k-local Toom's rule is then simply defined by interpersing a Toom's rule layer followed by a noise layer, every k parallel SWAP layers. Remark that this kind of shuffling does not break all macroscopic invariant configurations of Toom's rule, for example rows and columns are respectively only translated vertically and horizontally. Because the SWAP networks send neighbouring qubits in opposing directions, it however does prevent new errors to aggregate on an already existing structure. Note that alternative SWAP networks are possible but numerical simulations for several variants that we tried have indicated similar performance.

2.2.2 Local majority votes on a square lattice

Toom's rule is defined on a torus. The boundary counditions can however simply be fixed so that the decoder can be defined on a square lattice with nearest neighbours connectivity. To avoid errors accumulating in a corner, the orientation of the majority vote in the bulk of the lattice is changed at time intervals which are logarithmic in the system size, similarly as in [149]. The boundary conditions are then updated accordingly, as illustrated in Figure 2.4.

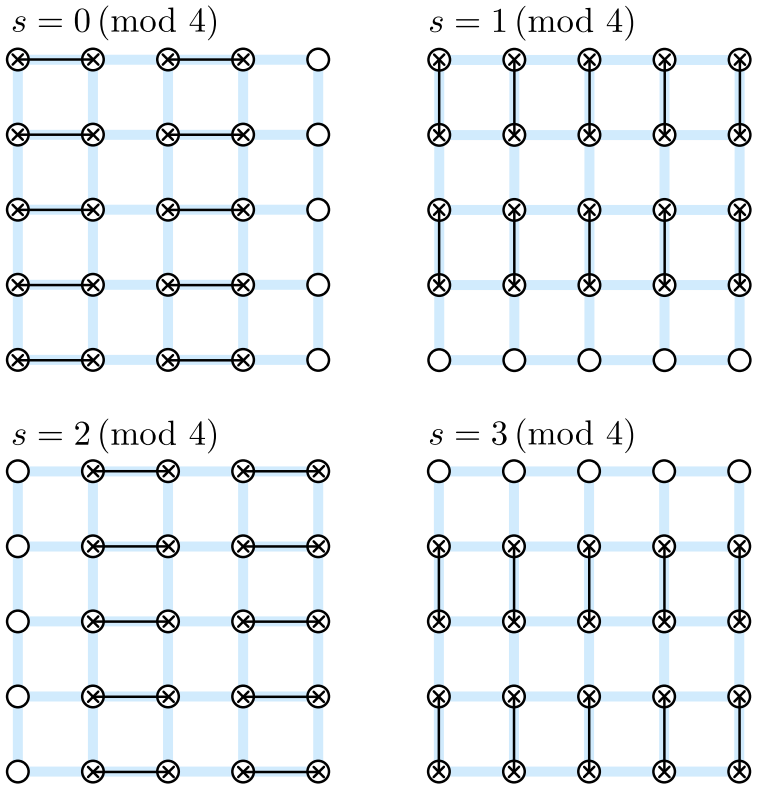

Figure 2.3: SWAP network for local shuffling on a 5×5 non-periodic lattice: horizontal and vertical SWAP gates are applied alternately between neighboring qubits. This procedure is repeated twice, with the second round offset by one qubit to target a different set of pairs. k-local Toom's rule is defined by interspersing one layer of local majority vote with k layers of parallel SWAP gates following the periodic pattern.

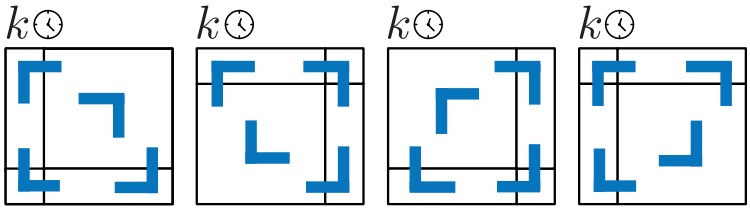

Figure 2.4: Schematic representation of the temporal pattern of Toom's rule where the blue arrow indicate the orientation of the three qubits majority vote. For example, in the first iterations, the majority vote is performed with the east and south, east and north, and west and north neighbors for the leftmost column, the bottom left corner, and the lowest row, respectively. Each type update rule is applied $k = \Theta(\log n)$ times.

We remark that alternating the orientation of the majority vote at time intervals $k = \Theta(\log n)$ will asymptotically lead to the persistence of logarithmic-size error clusters that the automaton fails to eliminate. A simple example is an error cluster supported on the first NW–SE diagonal of length greater than k, starting from the south-west corner. In such a case, the domain wall along the diagonal is only displaced alternately k sites to the left and right by the automaton, preventing its complete removal. While this mechanism should limit the suppression of logical errors in the asymptotic regime to polynomial rather than exponential scaling, numerical results presented in Figure 2.2 show that this does not significantly impair the automaton's performance for practical parameter ranges (less than a hundred qubits).

2.2.3 Global shuffling

Here, we rigorously establish the existence of a non-zero threshold for a global decoding scheme based on a 2-periodic update rule, referred to as the Global Toom Rule (GTR). The GTR consists of a random global permutation followed by a local majority-vote correction, and is analyzed in the phenomenological noise model wihout measurement errors for the sake of simplicity. Note that we do not expect measurement errors to change the qualitative behavior of the automaton since for a local decoder a single measurement impacts at most a constant number of qubit, hence measurement errors play a role similar to that of data errors.

We prove that, below a finite noise threshold, the logical error probability decays exponentially with the system size after a polynomial number of

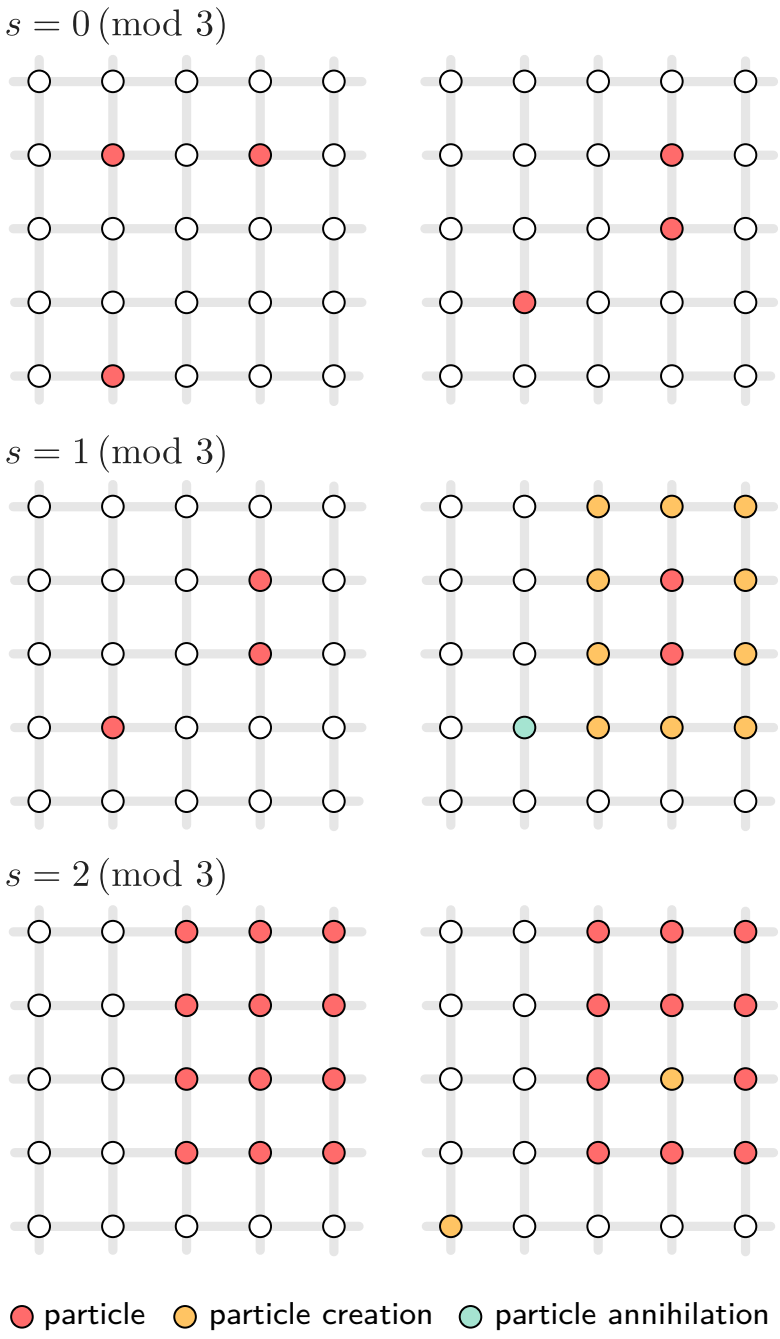

Figure 2.5: Illustration of the 3-periodic Shuffling Particles Game. In the first step, particles are randomly shuffled on the 2D grid. Isolated particles annihilate in the next step, while a particle is created on each adjacent site to a cluster of particles (2 or more). Finally, a particle is created with i.i.d probability on each empty site.

rounds. The proof proceeds by reducing the quantum dynamics to a classical probabilistic process—the Shuffling Particles Game (SPG)—which conservatively over-approximates the spread of faults. We show that, in the SPG, both the spontaneous creation and amplification of particle clusters are exponentially suppressed when the noise is below threshold. This reduction allows us to derive an explicit bound on the logical error rate, completing the threshold proof.

2.2.3.1 Proof of threshold

Let the 2D periodic lattice $\mathcal{L} = \mathbb{Z}_\ell \times \mathbb{Z}_\ell$ whose vertices and edges are respectively assigned qubits and Z-type Pauli stabilizers (parity check). We consider here the dynamics with 2-periodic update rule composed of (i) a random global permutation, (ii) a layer of Toom's like majority votes. The noisy version of the automaton is then defined by adding a layer of i.i.d bit-flip noise in between each iteration.

In the proof we will distinguish between errors, that arise in the form of a X-type Pauli gate at some specific space-time location with i.i.d probability ε, and faults at some given iteration, that corresponds to the set of X-type Pauli gates to apply to return to the initial code-state, or stated differently, to the accumulated X errors at a given time of the computation. Note that this distinction is possible because the repetition code is non-degenerate so that we can track without ambiguity faults along the time-evolution of the automaton.

Proof of Theorem 8. We denote by $E^{(t)}, F^{(t)} \subset \mathcal{L}$ the random variables corresponding respectively to the support of instantaneous X-type Pauli error and cumulated X errors at time $t \geq 0$. Because of the random global permutation at the begining of each iteration, we will be ultimately interested in number of faults, denoted $N^{(t)} := |F^{(t)}|$.

We prove a threshold theorem for the GTR in the ideal-measurement phenomenological model using a reduction to a simplified classical dynamics that we call the *Shuffling Particles Game* (SPG) induced by $E^{(t)}$. Consider the previously defined lattice whose sites are assigned a binary variable representing the presence or absence of a point-like particle and the set of non-empty locations is $\tilde{F}^{(t)}$. The dynamics that we consider correspond to the 3-periodic update rule composed of (i) a random global permutation of particles over the lattice, (ii) isolated particles are removed and for each non-single particle cluster we create a particle on all empty adjacent sites (including diagonal adjacent

sites), and (*iii*) an excitation event occurs on each site with i.i.d probability ε, a particle is created on an empty site, while occupied sites remain unchanged. The game is illustrated in Figure 2.5. This simplified dynamics is a conservative approximation of the original one in the following sense.

Lemma 1 (Reduction to SPG)**.** *Consider the sequences of random variables $F^{(t)}$ and $\tilde{F}^{(t)}$ defined from the same realization of underlying error and permutation processes. For all $t \geq 0$, it holds that $F^{(t)} \subset \tilde{F}^{(t)}$.*

This directly implies that $N^{(t)} \leq |\tilde{F}^{(t)}|$ and that it suffices to upper-bound $\tilde{N}^{(t)} := |\tilde{F}^{(t)}|$. It is left to prove that in the SPG, below some threshold, the probability to increase the number of particles is exponentially suppressed.

Lemma 2 (Exponential suppression of errors in the SPG)**.** *Consider the sequence of random variable $\tilde{N}^{(t)}$. There exists some ε_{th} and $\eta_{\text{th}} > 0$ such that if $\varepsilon < \varepsilon_{\text{th}}$ and for some $\eta < \eta_{\text{th}}$, for all $t \geq 0$ it holds that*

$$\mathbb{P}(\tilde{N}^{(t+1)} > \eta n \,|\tilde{N}^{(t)} < \eta n) \leq \mathcal{O}((\varepsilon/\varepsilon_{\text{th}})^{\eta n/2} + (\eta/\eta_{\text{th}})^{\eta n/2}). \tag{2.6}$$

The first term accounts for new errors at time t while the latter accounts for the amplification of previously existing errors by the correction step. The two Lemmas are proven in the next subsection. Combining repeated application of Lemma 2 with Lemma 1 gives Theorem 8.

□

2.2.3.2 Reduction to the Shuffling Particles Game

We start by proving that the dynamics can be conservatively represented by the Shuffling Particles Game, which simplifies the essential aspects of the original dynamics: (*i*) isolated faults are erased, and (*ii*) fault clusters (of more than 2 faults) can expand at their boundaries. In this simplified model, these properties are restated as: (*i′*) only isolated faults are erased, and (*ii′*) fault clusters always expand at their boundaries.

Proof of Lemma 1. The proof is done by recursion. In the following, we assume particles are discernable so that one can follow back the past trajectory of a given particle. Assume that for some $t \geq 0$ we have $F^{(t)} \subset \tilde{F}^{(t)}$, we want to prove that $F^{(t+1)} \subset \tilde{F}^{(t+1)}$. Let $f \in F^{(t+1)}$, f corresponds to a X-type Pauli fault at time $t+1$. Either $f \in E^{(t+1)}$ and in this case since $E^{(t+1)} \subset \tilde{F}^{(t+1)}$

we have $f \in \tilde{F}^{(t+1)}$. If $f \notin E^{(t+1)}$ then necessarily f corresponds to (i) a fault $f' \in F^{(t)}$ sent to site f from the global permutation or (ii) was created by the correction operation at time t. If (i) we also directly have $f \in \tilde{F}^{(t+1)}$ because $f' \in F^{(t)} \subset \tilde{F}^{(t)}$ by assumption. If (ii) then necessarily f is next to at least 2 other pre-existing faults and the simplified correction step implies $f \in \tilde{F}^{(t+1)}$. We conclude by recursion. □

2.2.3.3 Threshold for the Shuffling Particles Game

We now prove that if the number of particles of the Shuffling Particles Game is below some threshold ratio at a given iteration, it remains so at the next iteration with exponentially high probability.

Proof of Lemma 2. Let $\tilde{N}^{(t)}$ the random variable associated to the number of particles within the lattice at time $t \geq 0$, initialized in $\tilde{N}^{(0)} = 0$. Note that the frontier of a cluster of $p > 1$ particles is made of at most $6p$ sites (the worst case being 2 diagonally adjacent particles) so that $\tilde{N}^{(t)}$ can be upper bounded by

$$\tilde{N}^{(t+1)} \leq 6C[\tilde{N}^{(t)}] + B^{(t)}. \tag{2.7}$$

Where $C[p]$ is the random variable associated to the number of particles in a cluster after randomly placing p particles on the 2D grid and $B^{(t)}$ follows a binomial law of parameters (n, ε). The proof uses a key Lemma on the probability distribution of $C[p]$, where we introduce the constant $z = 8$ corresponding to the number of neighbors of a site.

Lemma 3 (Exponentially rare clusters)**.** *Let $p, q \in [\![1, n]\!]$ be with $q \leq p$, we have*

$$\mathbb{P}(C[p] \geq q) \leq \sum_{i \geq \lceil q/5 \rceil} \binom{p}{i} (\frac{zp}{n-p})^i. \tag{2.8}$$

We are now ready to bound the probability that at some time t the number of particles starts exceeding a fraction of n (the total number of sites). Let $\eta < 1$ that will be specified later. Since $B^{(t)}$ and $\tilde{N}^{(t)}$ are independent, the union bound gives

$$\mathbb{P}(\tilde{N}^{(t+1)} > \eta n \,|\, \tilde{N}^{(t)} \leq \eta n) \leq \mathbb{P}(6C[\tilde{N}^{(t)}] > \frac{\eta n}{2} \,|\, \tilde{N}^{(t)} \leq \eta n) + \mathbb{P}(B^{(t)} > \frac{\eta n}{2}). \tag{2.9}$$

The first term is bounded using Lemma 3

$$\begin{aligned}
\mathbb{P}(6C[\tilde{N}^{(t)}] > \frac{\eta n}{2} \,|\, \tilde{N}^{(t)} \leq \eta n) &\leq \mathbb{P}(6C[\tilde{N}^{(t)}] = \frac{\eta n}{2} \,|\, \tilde{N}^{(t)} = \lfloor \eta n \rfloor) && (2.10)\\
&\leq \sum_{i \geq \lfloor \eta n \rfloor / 60} \binom{\lfloor \eta n \rfloor}{i} (\frac{z \lfloor \eta n \rfloor}{n - \lfloor \eta n \rfloor})^i && (2.11)\\
&\leq \sum_{i \geq \lfloor \eta n \rfloor / 60} (\frac{60 e z \eta}{1 - \eta})^i. && (2.12)
\end{aligned}$$

Where we have used the standard upper bound on binomial coefficients $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$. For $\eta < \eta_{\text{th}} := (60ez + 1)^{-1} \simeq 0.00077$, the sum converges so that

$$\mathbb{P}(6C[\tilde{N}^{(t)}] > \frac{\eta n}{2} \,|\, \tilde{N}^{(t)} \leq \eta n) \leq \frac{q_1^{\frac{\lfloor \eta n \rfloor}{60}}}{1 - q1} \quad \text{for} \quad q_1 = \frac{60 e z \eta}{1 - \eta} < 1. \tag{2.13}$$

The second term corresponds to at least $\lceil \frac{\eta n}{2} \rceil$ spontaneous bit-flip errors which occurs with probability

$$\begin{aligned}
\mathbb{P}(B^{(t)} \geq \frac{\eta n}{2}) &= \sum_{i > \frac{\eta n}{2}} \binom{n}{i} \varepsilon^i (1 - \varepsilon)^{n-i} && (2.14)\\
&\leq \sum_{i > \frac{\eta n}{2}} (\frac{2e\varepsilon}{\eta})^i. && (2.15)
\end{aligned}$$

For $\varepsilon < \varepsilon_{\text{th}} := \frac{\eta}{2e} \simeq 0.0001$ where for the numerics we have taken $\eta = 0.0007$, the sum converges and is upper-bounded by

$$\mathbb{P}(B^{(t)} \geq \frac{\eta n}{2}) \leq \frac{q_2^{\frac{\eta n}{2}}}{1 - q_2} \quad \text{for} \quad q_2 = \frac{2e\varepsilon}{\eta} < 1. \tag{2.16}$$

This finishes the proof. Now we come back to the proof of Lemma 3. □

Proof of Lemma 3. We sequentially place p particles uniformally at random on

empty sites of the 2D lattice and we denote by X_i for $i \leq p$ the random variables defined by

$$X_i = \begin{cases} 1 & \text{if the particle } i \text{ is adjacent to any particle } j < i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Note that necessarily we have $X_1 = 0$ because the lattice is empty at that time. The history for all $i \leq p$ is noted $X^p = (X_1, ..., X_p) \in \{0,1\}^p$. The number of particles in a cluster at the end of the placement can be upper-bounded as a function of the X_i by $C[p] \leq 5 \cdot |X^p|$. To see this it suffices to notice at each placement, either there is no contact and the number of clusterized particle is constant, or there is one in which case we can increase the number of particles by at most 5, corresponding to a new particle in the middle of four isolated particles (N, E, S, W) or (NE, SE, SW, NW). It is left to upper-bound the probability $\mathbb{P}(|X^p| \geq \lceil q/5 \rceil)$. To do so we decompose the event into histories of equal Hamming weight

$$\mathbb{P}(C[p] \geq q) \leq \mathbb{P}(|X^p| \geq \lceil q/5 \rceil) \tag{2.18}$$

$$\leq \sum_{i \geq \lceil q/5 \rceil} \Big(\sum_{\substack{x \in \{0,1\}^p \\ |x| = i}} \mathbb{P}(X^p = x) \Big). \tag{2.19}$$

Then the probability of a given collision historic $\mathbb{P}(X^p = x)$ for $x = (x_1, ..., x_p) \in \{0,1\}^p$ can be lower-bounded by recursion

$$\mathbb{P}(X^p = x) = \mathbb{P}(X_p = x_p \,|\, (X_1, ..., X_{p-1}) = (x_1, ..., x_{p-1})) \tag{2.20}$$

$$\times \, \mathbb{P}((X_1, ...X_{p-1}) = (x_1, ..., x_{p-1})) \tag{2.21}$$

$$\leq (\frac{zp}{n-p})^{x_p} \times \mathbb{P}((X_1, ...X_{p-1}) = (x_1, ..., x_{p-1})) \tag{2.22}$$

$$\leq (\frac{zp}{n-p})^{\Sigma x_i} = (\frac{zp}{n-p})^{|x|}. \tag{2.23}$$

The last inequality holds because for all $i \leq p$ the probability that the particle i is adjacent to a particle $j < i$ is upper-bounded by the probability that the particle p. Stated differently, the collision probability is increasing with the

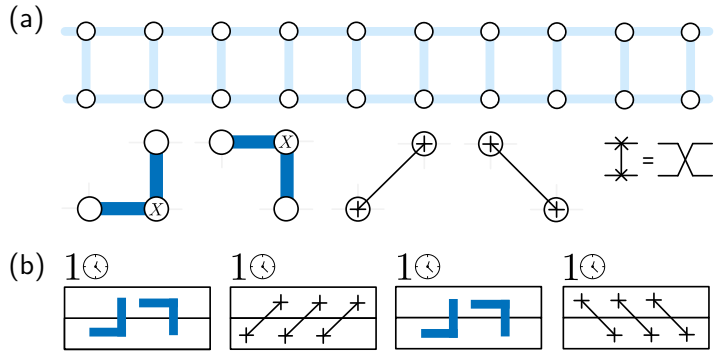

Figure 2.6: (a) Representation of the 1D repetition code on two stacked periodic rows of qubits. Qubits are assigned to vertices and Z stabilizers are assigned to edges. (b) The automaton follows a 4-periodic update rule alternating local majority votes and swapping qubits along the two diagonals.

number of particles already on the grid so that it can be upper-bounded by the probability of a collision for the last particle. This property is itself upper-bounded by $\frac{zp}{n-p}$ (the ratio of sites adjacent to p particles over empty sites, maximum when all particles are isolated in the sense that no site is adjacent to two or more particles). Combining with (2.18) gives

$$\mathbb{P}(C[p] \geq q) \leq \sum_{i \geq \lceil q/5 \rceil} \Big(\sum_{\substack{x \in \{0,1\}^p \\ |x|=i}} (\frac{zp}{n-p})^i \Big) \tag{2.24}$$

$$\leq \sum_{i \geq \lceil q/5 \rceil} \binom{p}{i} (\frac{zp}{n-p})^i. \tag{2.25}$$

This finishes the proof. □

2.3 Squeezing out the second dimension

2.3.1 The Shearing-rule

Here, we introduce a new cellular automaton, the *shearing-rule*, defined on two rows of qubits and that exhibits error correction properties for a practical range of parameters. The automaton is obtained from collapsing one dimension of Toom's rule on a torus to length 2, and by interspersing majority votes with a well chosen SWAP network. Although the shearing-rule does not possess a

threshold similarly to all simple constructions of 1D automata without classical memories [70], the logical error is exponentially suppressed up to system size $n \lesssim 40$ and offers a practical advantage over other q1D automata described in the introduction by eliminating the need for a classical memory. As a result, it presents an appealing short-term solution for local decoding of the repetition code.

Let n be an even integer. Consider the periodic lattice $\mathcal{L} = \mathbb{Z}_2 \times \mathbb{Z}_{n/2}$ and assign respectively to its vertices and edges, qubits and Z-type stabilizers (parity check). The Z-type stabilizers are defined for $i \in \{0, 1\}$ and $j < n/2$ by

$$S_{i,j} := Z_{i,j} Z_{i,j+1}, \tag{2.26}$$

$$S_j := Z_{0,j} Z_{1,j}. \tag{2.27}$$

The operators can easily be seen to generate the n-qubit repetition code stabilizer group. The shearing-rule is defined on $\mathcal{L}$, with following elementary operations illustrated in Figure 2.6: (i) X-type correction for qubits of the top row (resp. bottom) in odd parity with their bottom (resp. top) and left neighbours, (ii) qubits permutation along the left diagonal and (iii) qubits permutation along the right diagonal. The automaton consists of the repeated sequential application of (i), (ii), (i), (iii) in that order. Note that the orientation of the majority vote of step (i) can be changed on the bottom row to make the error correction work on a non-periodic lattice. Up to a row permutation, we note that the automaton induces a similar dynamics to that of [70] (that adapts Toom's *Two-Line Voting* (TLV) automaton [122] to the quantum setting but with a large circuit complexity overhead), where a cluster over two rows is separated by the dynamics into two separate clusters respectively on the top and bottom row. At this point the two clusters are erased independently by the automaton. An example of erasure of an error cluster is given in Figure 2.7. This asymmetric displacement between the two rows of the error cluster is emphasized in a simplified representation we refer to as the shifting frame, illustrated in Figure 2.8. There, only the odd-numbered iterations are shown, with every second one displayed as a vertical reflection.

Note that for the repetition code, the permutation of two qubits can be performed via a SWAP gate but also by a parity measurement followed by an XX correction conditioned on a -1 parity outcome. This representation would

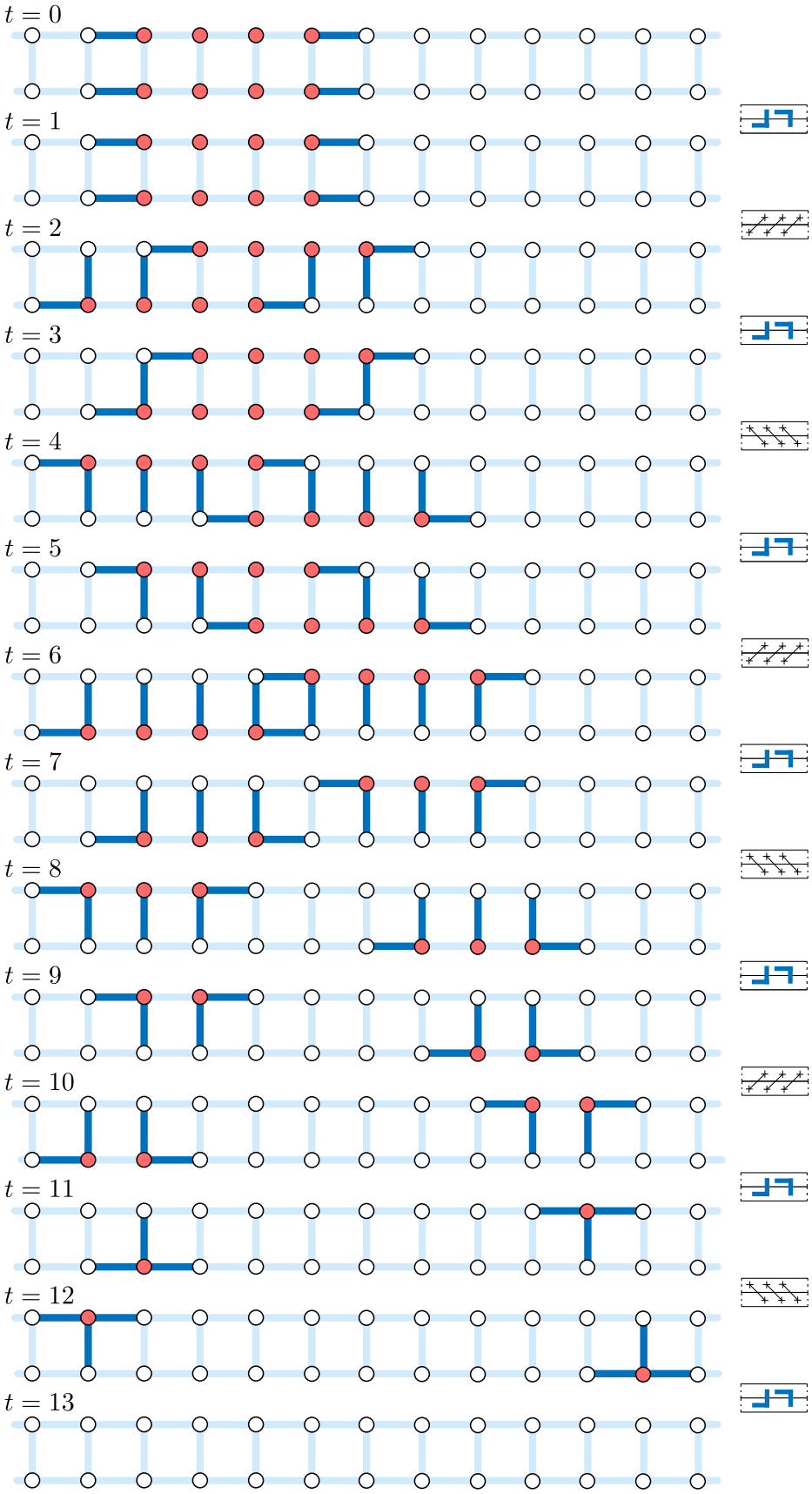

Figure 2.7: Erasure of errors by successive application of the shearing-rule.

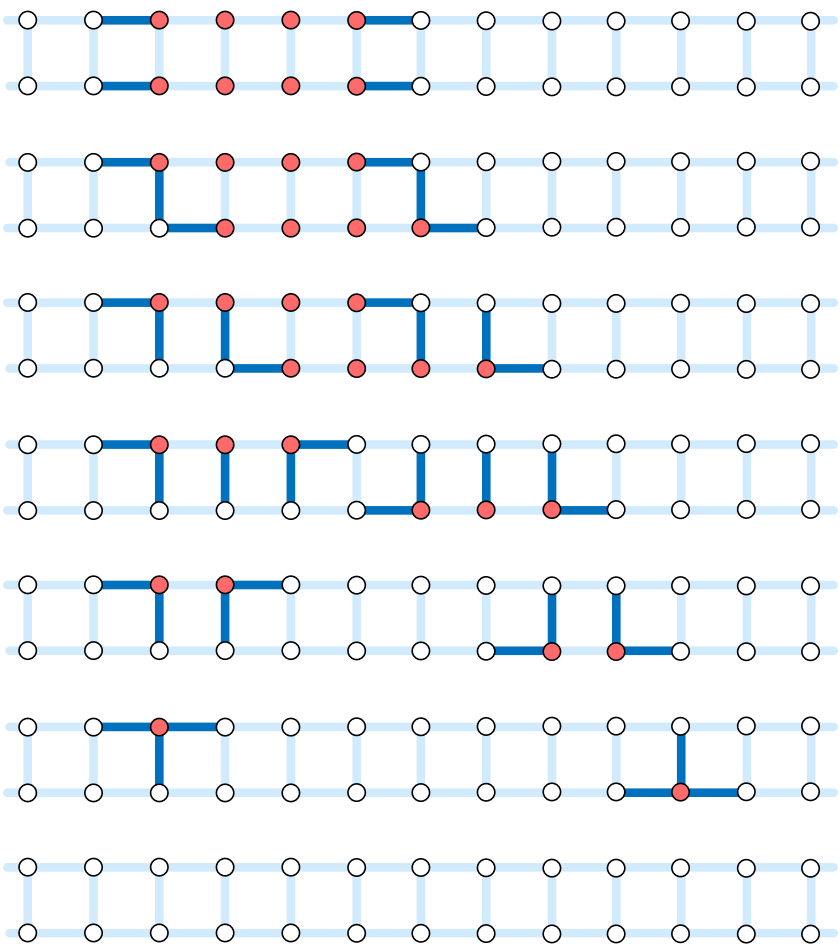

Figure 2.8: An alternative representation of the error cluster erasure from Figure 2.7, showing only the odd-numbered iterations, one out of two is alternatively displayed as a vertical reflection. Since there is one majority vote and one swapping layer between each stage that is represented, the upper rows is shifted by 1 to the right at between each stage, causing the initial error to split into two disjoint clusters. We refer to this reference frame as the shifting frame.

generalize better to quantum codes, that unlike the quantum repetition code don't have permutation invariant stabilizer in general[2]. Concerning majority votes, a straightforward circuit implementation uses a Toffoli-gate per majority vote. The Toffoli count is crucial because, in practical physical implementations, three-qubit gates are likely to introduce the most errors. Notably, our construction requires only a third as many Toffoli gates per site per correction step compared to [70].

2.3.2 Logical failure from logarithmic weight errors

The saturation of the logical error rate from a certain system size on can be understood as follows. We remark that the error correction process can be perturbated in the presence of space-time errors. In the example of Figure 2.9, the size of an initial rectangular error cluster is multiplied by some $\eta > 1$ in the presence of an adversarial pair of errors (one on the left, one on the right), before the cluster separates in two. Repeated similar adversarial errors give a space-time error cluster of weight $\log n$, preventing any exponential suppression of the logical error.

We also give a more general intepretation motivating the next chapter. To simplify the discussion, we will consider the classical version of the automaton. In this setting, the two-row binary automaton can be viewed as a single-row automaton with local state space $\{00, 01, 10, 11\}$. Notice that states 01 (respectively 10) propagate one site to the right (respectively left) at each time step, unless the neighboring state on the left (respectively right) is either 00 or 11, in which case the minority bit flips. In this perspective, the automaton acts as a one-dimensional system where computation is performed via 'signals'—namely the 01 and 10 states—emanating from the boundaries of error clusters. However, because these signals are themselves subject to noise, the probability that they propagate over long distances decays exponentially. We confirm this intuition in Figure 2.10 by numerical evidence of exponentially decaying correlation in distance e.g. for qubits of the first row. This results in a finite effective interaction range and the loss of error correction beyond a certain cluster size.

We however remark that such signals convey purely classical information—even in the quantum setting—and could therefore be directed to a local classical register, which can reasonably be assumed to be noiseless. This observation mo-

[2]With the notable exception of [117]

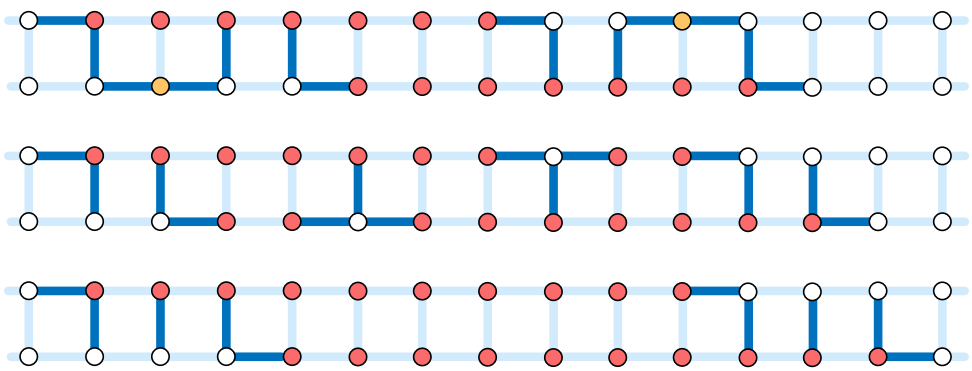

Figure 2.9: Illustration of the error amplification mechanism in the shifting frame. Starting from two vertically stacked, laterally shifted lines of K faulty qubits (represented in red), we consider the dynamics in the presence of two new adversarial errors (represented in yellow). In this frame, the new fault on the top row remains immobile but one new error is created to its left at each stage, until the gap with the faulty line on the left is completely filled. The same dynamics arises on the bottom row up to a global translation by 1 to the right between each stage. Importantly, the final structure is similar to the initial structure with each line length increased by $\Theta(K)$. Repeating the mechanism yields a macroscopic fault cluster from $\Theta(\log n)$ adversarial errors.

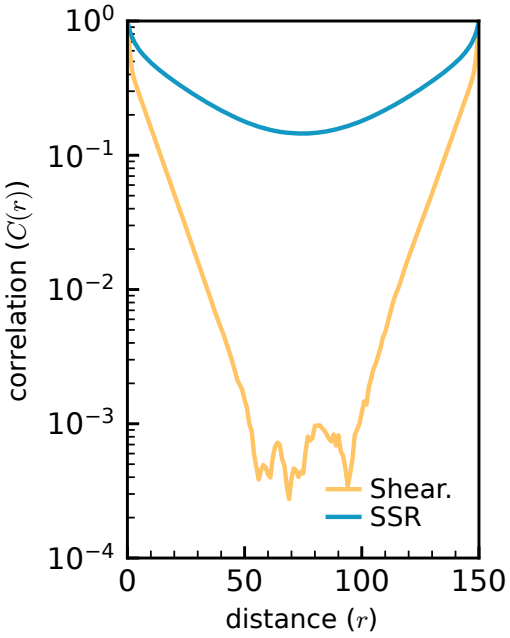

Figure 2.10: Pearson correlation $C(r) = \langle u_0 u_{0+r} \rangle - \langle u_0 \rangle \langle u_{0+r} \rangle$ between sites $(0, i)$ and $(0, i + r)$ as a function of distance r. The correlation is computed from the simulation of classical analogue of the shearing rule of length $n/2 = 150$, under a phenomenological noise model with $\varepsilon_d = \varepsilon_m = 5\%$. The observed exponential decay of correlations with distance indicates a lack of long-range order, which in turn prevents asymptotic suppression of logical errors. For comparison with system with a threshold, we also show results for the SSR decoder (in blue), introduced in the following chapter.

tivates the constructions in the next chapter, where each site of the quantum code is equipped with a classical automaton. This automaton can access the corresponding check value, communicate with neighboring automata, update its local classical register and apply local Pauli corrections accordingly.

2.4 Performances

2.4.1 Logical error rate

We numerically study the stability of the automata from Monte Carlo simulations in the phenomenological error model where each (data) qubit and stabilizer measurement result are flipped with probability ε. The automaton is initialized in the zero configuration and we denote by $\overline{P}(\tau)$ the probability of a logical flip (in this case defined by a majority of 1) at time τ. The logical error rate $\overline{\varepsilon}$ is then defined as the normalized logical flip probability and is shown in Figure 2.11.The data is fitted for each automaton with the ansatz $An(B\varepsilon)^{\gamma_n}$ with a different parameter γ_n for each n, where γ_n is plotted in Figure 2.2. γ_n corresponds to the weight of the typical space-time error cluster leading to a logical failure. Note that the saturation of γ_n for the shearing-rule from $n \simeq 30$ to the existence of logarithmic weight space-time error configuration leading to a logical failure described in Subsection 2.3.2. We use the fit to estimate the number of physical qubits necessary to reach a given logical error rate for $\varepsilon \in \{0.001, 0.01\}$.

2.4.2 A remark on SWAP noise

Note that in all numerical simulations, we suppose that the noise strength during SWAP layers is independent of the SWAP depth. This is not realistic in general but is a good proxy to estimate the gain obtained from swapping qubits if the associated noise is particularly smaller than say other 2-qubit gates, this is for example the case of atom arrays platforms [16] where SWAP gates would simply correspond to moving qubits around. We nonetheless show that in this setting, two layers of parallel SWAP operations suffice to achieve comparable performance over a relevant range of parameters. This is perhaps not suprising since according to percolation theory, largest error clusters grow logarithmically with the lattice size, hence logarithmic SWAP depth should be enough for the

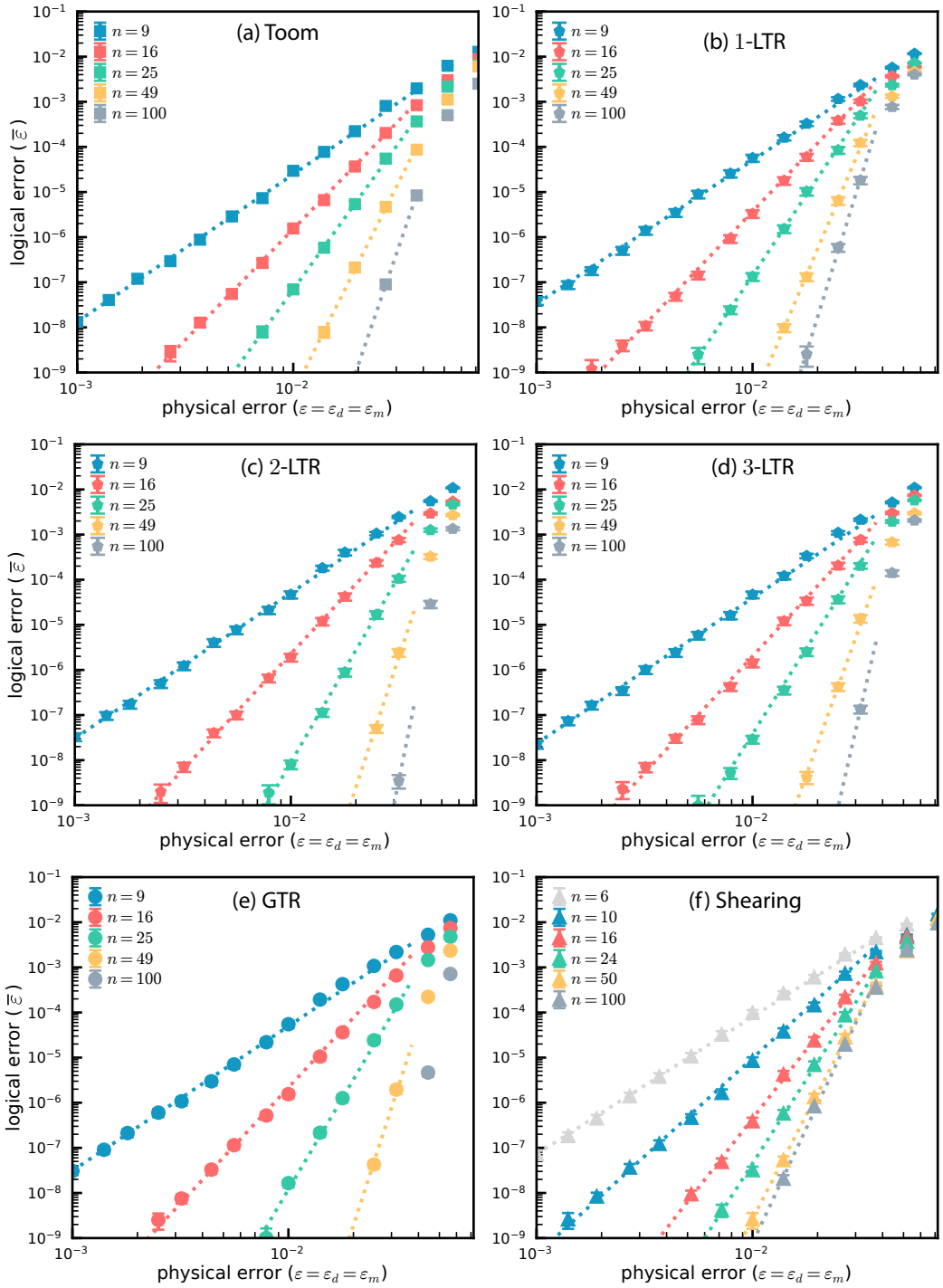

Figure 2.11: Logical error rate as a function of the physical error rate for several system sizes in the phenomenological model, for all automata considered in the chapter. The logical error rate is obtained from the normalization of the logical flip rate estimated on Monte Carlo simulations for a fixed time τ large enough so that $\bar{\varepsilon}$ is independent of τ. For each automaton the data is fitted with the ansatz $An(B\varepsilon)^{\gamma_n}$ with a different parameter γ_n for each n, where γ_n is plotted in Figure 2.2.

shuffling to appear global relative to individual error clusters.

Chapter 3

Local decoders for defect matching in 1D

This chapter covers the work that was published in [119].

Contents

3.1 Introduction

Local decoders offer a promising path toward real-time quantum error correction by replacing centralized classical decoding, with inherent hardware constraints, by a natively parallel and streamlined architecture from a simple local transition rule. A simple example is Toom's rule that protects a bit of information on a 2D lattice, that is to say with an extra dimension compared to the usual repetition code. In the last chapter we have shown that although one can collapse the grid to only two rows and retain good practical performance, this shearing rule, as thereby defined, lacks a threshold.

We propose addressing this shortcoming with a new type of local decoder for the quantum repetition code in one dimension, making use of access to small local classical memories. The signal-rule decoders interpret odd parities between neighboring qubits as defects, attracted to each other via the exchange of classical point-like excitations, represented by a few bits of local memory. We prove the existence of a threshold in the code-capacity model and present numerical evidence of exponential logical error suppression under a phenomenological noise model, with data and measurement errors at each error correction cycle. Compared to previously known local decoders that suffer from sub-optimal threshold and scaling, our construction significantly narrows the gap with global decoders for practical system sizes ($n \lesssim 30$) and error rates. When combined with biased-noise qubits, such as cat qubits, the signal-rule decoders enable a fully local quantum memory in one dimension.

3.1.1 Local decoders of the 1D repetition code

We focus here on local decoders for the quantum repetition code under Pauli X errors. While this is a very simple code, it can be combined with biased-noise qubits [99, 112, 127] to yield a quantum memory [71, 133, 40, 128, 102]. In addition, it serves as a toy model for the surface code, but in one spatial dimension instead of two.

The n-qubit 1D repetition code is defined by placing qubits on the n edges on a cycle and stabilizers $S_i := Z_{(i-1,i),(i,i+1)}$ on its vertices $i \in \mathbb{Z}_n$. It encodes a single logical qubit, with logical codewords $|k\rangle_L := |k\rangle^{\otimes n}$ for $k \in \{0, 1\}$. The syndrome of an X-type error E defined on the edges of the cycle corresponds to the boundary of the error. It will be convenient to represent it as $\Sigma = \partial E := \{\sigma_1, \ldots, \sigma_\ell\} \subseteq \mathbb{Z}_n$, i.e. as the set of vertices where the values of the incident

edges differ. We take the convention that $\sigma_j < \sigma_{j+1}$ and remark that this set has even cardinality. The corresponding vertices carry point-like excitations that we call *defects*.

The decoding problem is equivalent to finding a correct matching of these defects. There are only two solutions, corresponding to the true error E and to its complement $\mathbb{Z}_n \setminus E$. The decoder succeeds if it correctly recovers E. An optimal decoder is readily available if it has access to the full syndrome since it suffices to choose the matching that minimizes the weight of the error. In addition to this *code-capacity scenario* where each qubit belongs to E independently with probability ε, we will also be concerned with the *phenomenological error model* where each (data) qubit is flipped at each time step with probability ε_d and each stabilizer measurement result is flipped with probability ε_m.

3.1.2 Main results

In this chapter, we define two new 1D-local decoders for the quantum repetition code, the *asymmetric signal-rule* (ASR) and the *symmetric signal-rule* (SSR). The ASR is the simpler one to describe and analyze, but its symmetrized version, the SSR, leads to better performance. We compare the decoders with a variant of Toom's rule on a flat 2D surface and with the *shearing rule* defined in the last chapter.

We first prove a threshold theorem for the ASR in the code-capacity model, when the rule is applied on each site for $\tau = \mathcal{O}(n)$ time steps.

Theorem 9 (ASR code-capacity threshold)**.** *There exist $\varepsilon_{th} > 0, \alpha > 0$ and $\tau = \mathcal{O}(n)$ such that for $\varepsilon < \varepsilon_{th}$, the logical error rate $\overline{\varepsilon}$ of the ASR applied for τ time steps to an initial error where each qubit is flipped independently and identically with probability ε satisfies $\overline{\varepsilon} \leq \exp(-n^{\alpha})$.*

We show that $\alpha > 0.12$ in the proof, but our numerical simulations (in the more general phenomenological model) indicate that this bound is far from tight. The proof combines the fact that an isolated cluster of errors is erased in a time proportional to its size, with a general technique of hierarchical decomposition of error configurations from renormalization group decoders [29, 74, 60, 93]. A cluster of errors belonging to a given level of the hierarchy is erased independently from upper levels so that only error clusters from sufficiently high levels can induce a logical flip. Note that a similar result holds with a polylogarithmic

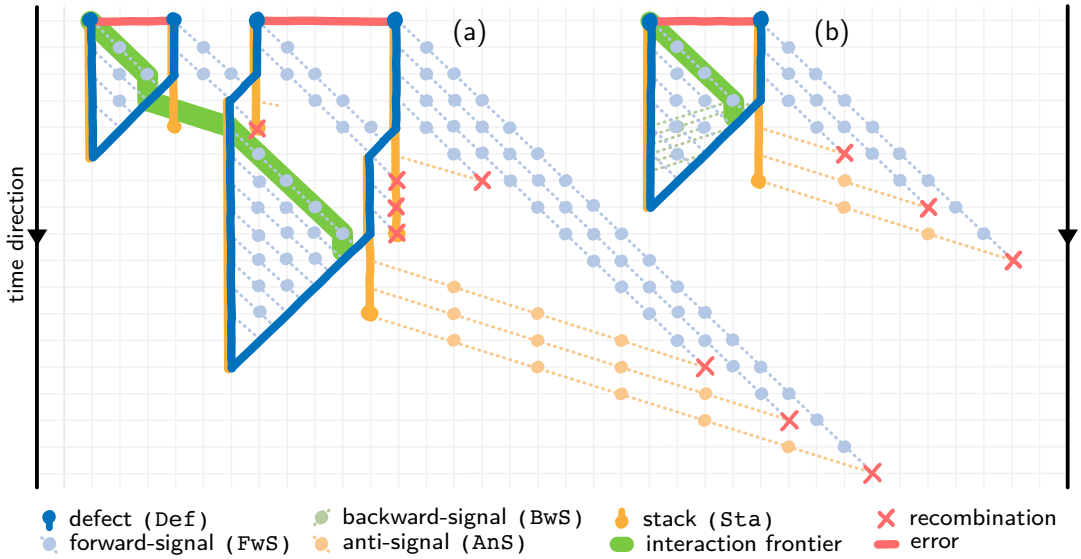

Figure 3.1: (a) Simplified representation of the erasure of a complex cluster where anti-signals are only represented at their creation when on the left of the rightmost syndrome, and backward-signals are not represented. (b) Space-time representation of the erasure of a simple error cluster with time going downwards. Each defect sends forward-signals to its right until forward-signals sent from the left defect reaches the right defect, at which point the right defect is displaced to the left for each forward-signal it receives until it recombines with the left defect. Forward-signals that have induced a defect displacement transform into backward-signals that propagate in the opposite direction to recombine with the left stack. Forward-signals sent by the right defect recombine with faster anti-signals created from the decrement of the right stack when it no longer coincides with a defect.

number of decoder iterations, but yields only subexponential error suppression. The proof of erasure of an isolated cluster is outlined in the main text, and included alongside the hierarchical decomposition in Section 3.4.

Then we numerically evaluate the performance of SSR for the phenomenological model, as illustrated in Figure 3.2, and pick $\varepsilon_d = \varepsilon_m = \varepsilon$ unless stated otherwise. We find that the SSR approaches the performance of global decoders in terms of effective distance for small system sizes, and we show evidence for asymptotic exponential suppression of the logical error probability in n^α with $\alpha > 0$. In practice, the 1D SSR outperforms Toom's rule for practically relevant parameters, without exhibiting any saturation of the logical error rate for large n.

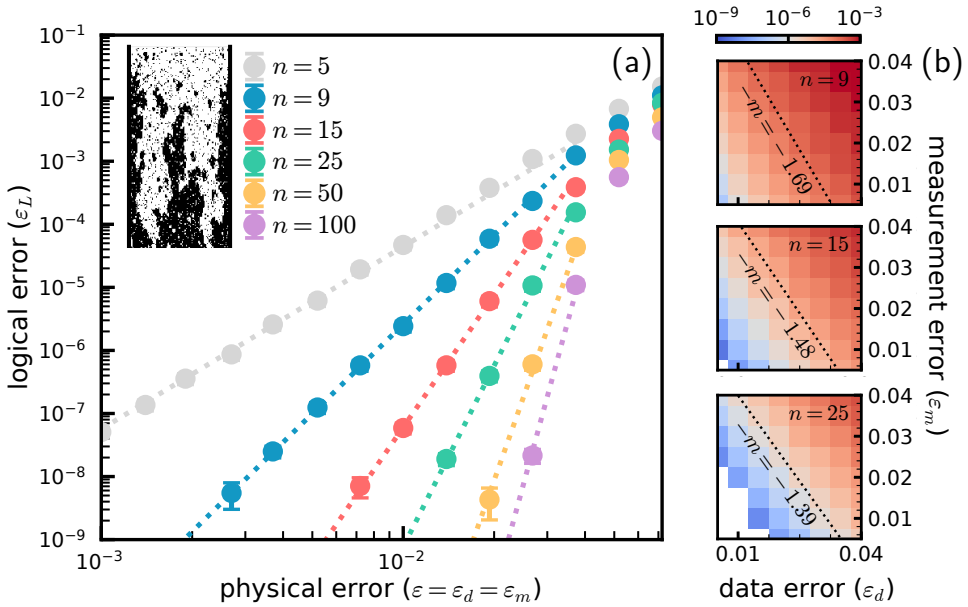

Figure 3.2: (a) Logical error rate as a function of the physical error rate for several system sizes in the phenomenological model. The logical error rate is obtained from the normalization of the logical flip rate estimated on Monte Carlo simulations for a fixed time τ large enough so that $\overline{\varepsilon}$ is independent of τ. The data is fitted with the ansatz $An(B\varepsilon)^{\gamma_n}$ with a different parameter γ_n for each n, where γ_n is plotted in Figure 3.5. B^{-1} is estimated to 6.6%. (Inset) Space-time representation of a logical bit-flip for $n = 100$, $\tau = 200$ and $\varepsilon = 6\%$, with time flowing downwards. (b) Logical error rate as a function of the data and measurement error probabilities ε_d and ε_m. We fit $\overline{\varepsilon}$ to the ansatz $A_n(\varepsilon_d + \varepsilon_m/m)^{\gamma_n}$ where γ_n is taken from (a) and draw an associated contour line for $n = 9, 15, 25$.

3.2 Signal-rule decoders

3.2.1 Asymmetric signal-rule

The local decoder is defined via a transition rule that updates classical variables assigned to each vertex of $\mathbb{Z}_n$. Each such site hosts four binary registers encoding the presence of four types of point-like particles: defects, forward-signals, backward-signals and anti-signals, as well as an additional integer register, the stack, that serves as a reservoir of anti-signals. A site is represented on Figure 3.3 (c) and a configuration $u^{(t)} \in U_n := (\mathbb{Z}_2^4 \times \mathbb{N})^n$ of the automaton at time t corresponds to the value of all variables on all sites. The automaton mediates an attractive interaction between defects through the exchange of signals. We sketch its dynamics here. At each iteration, a defect emits a forward-signal,

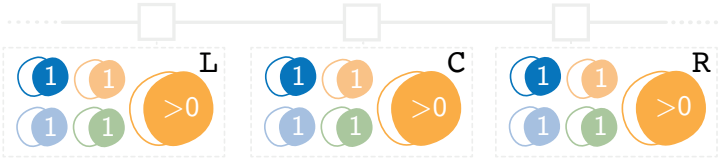

Figure 3.3: Local representation of the automaton variables. Each site is doted with four binary variables depicted as colored particles: in dark blue for defects, sky blue for forward signals, light orange for anti-signals, and green for backward signals. Finally, the stack (integer variable) is shown in orange. At each iteration of the decoder, the state of site `C` is updated based on its parity check and the states of neighboring sites `L` and `R`.

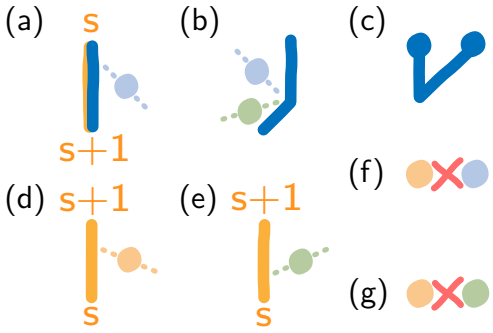

Figure 3.4: Summary of particles creation and annihilation rules with time going down. (a) Creation of a forward-signal and stack increment in the presence of a defect, (b) a forward-signal is reflected into a backward-signal when encountering a defect, the defect is displaced by 1 to the left (c) adjacent defects recombine (d) non-empty stack decrements to create an anti-signal in the absence of a defect (e) non-empty stack decrements upon receiving a backward-signal, (f) recombination of anti-signal with a forward-signal, (g) with a backward-signal.

which propagates to the right until meeting another defect. In that case, the encountered defect moves one step to the left and the forward-signal becomes a backward-signal traveling more rapidly to the left. To keep track of the forward-signals that have been emitted, one increments the local stack at the defect site when a forward-signal is sent, and decrements it when a backward-signal comes back, in which case the backward-signal is also annihilated. In addition, when a stack is no longer associated with a defect (because the latter has moved to the left), it sends anti-signals that move quickly to the right and whose goal is to recombine with the remaining forward or backward-signals. A sequence of automaton configuration $u^{(t)} \in U_n$ is represented on Figure 3.1 (a-b).

Algorithm 3 describes the update rule on a central site `C`. The automaton can

read the values of the registers at its neighboring left and right sites, `L` and `R`, but can only update the registers at `C`. Each step of the rule is applied in parallel on all the sites. The state of the automaton at the beginning of an iteration on site $\texttt{X} \in \{\texttt{L}, \texttt{C}, \texttt{R}\}$ is given by the values $\texttt{Var.X}$ for $\texttt{Var} \in \{\texttt{Def}, \texttt{FwS}, \texttt{BwS}, \texttt{AnS}, \texttt{Sta}\}$. Within an iteration we use the temporary variables `Tmp` and `Cor` for information transfer between sites and to save the correction. Note that in the phenomenological model, the value of $\texttt{Def.C}$ is reset to the parity measurement between the left and right edges at the beginning of each iteration, and we apply an X-type correction to the left edge if $\texttt{Cor.C} = 1$ at the end. Anti-signals and backwards-signals travel respectively at speed $k_a \geq 3$ and $k_b \geq 2$. We set $k_a = k_b = 3$ in the following and in numerical simulations. Finally, the symmetric signal-rule is obtained by combining an ASR as above, pointing to the right, with a second ASR pointing to the left so that the attraction works in both directions. A description of the interaction between the two ASR rules is included in Section 3.2.2.

Algorithm 3: ASR update rule - part 1/2

Input: Local variables `Var.X` for `Var` in {`Def`, `FwS`, `BwS`, `AnS`, `Sta`} and `X` in {`L`, `C`, `R`}

Output: Local variables

Initialize `Cor.C` and `Tmp.C` to 0;

Correct neighboring defects;

if $(\texttt{Def.L}, \texttt{Def.C}, \texttt{Def.R}) = (0, 1, 1)$ **then**

  Set `Cor.C` to 1; set `Def.C` to 0;

if $\texttt{Cor.L} = 1$ **then**

  Set `Def.C` to 0

Send forward-signals;

if $(\texttt{Def.C}, \texttt{Def.R}) = (1, 0)$ **and** $\texttt{FwS.C} = 0$ **then**

  Set `FwS.C` to 1; increment `Sta.C` by 1;

Propagate forward-signals to the right;

Set `Tmp.C` to `FwS.C`; `FwS.C` to `Tmp.L`; and `Tmp.C` to 0 ;

Correction and signals reflection;

if $\texttt{Def.C} = \texttt{FwS.C} = 1$ **then**

  Set `Tmp.C` to 1; set `Def.C` to 0; set (`FwS.C`, `BwS.C`) to $(0, 1)$ **if** $(\texttt{FwS.C}, \texttt{BwS.C}) = (1, 0)$

if $\texttt{Tmp.R} = 1$ **then**

  Set `Cor.C` to 1; set `Def.C` to 1; set (`FwS.C`, `BwS.C`) to $(0, 1)$ **if** $(\texttt{FwS.C}, \texttt{BwS.C}) = (1, 0)$

Propagate backward-signals by k_b to the left and recombine with anti-signals and stack;

repeat k_b *times*

  Set `Tmp.C` to `BwS.C` and `BwS.C` to `Tmp.R`;

  if $\texttt{BwS.C} = \texttt{AnS.C} = 1$ **then** set both to 0;

  if $\texttt{BwS.C} = 1$ **and** $\texttt{Sta.C} > 0$ **then** set `BwS.C` to 0 and decrement `Sta.C` by 1;

Send anti-signals;

if $\texttt{Def.C} = \texttt{AnS.C} = 0$ **and** $\texttt{Sta.C} > 0$ **then**

  Set `AnS.C` to 1; decrement `Sta.C` by 1;

The algorithm continues on the next page.

Algorithm 4: ASR update rule - part 2/2

Propagate anti-signals by k_a to the right and recombine with forward and backward-signals;
repeat $k_a - 1$ *times*
    Set `Tmp.C` to `AnS.C` and `AnS.C` to `Tmp.L`;
    if `AnS.C` $=$ `FwS.C` $= 1$ **then** set both to 0;
    if `AnS.C` $=$ `BwS.C` $= 1$ **then** set both to 0;
Set `Tmp.C` to `AnS.C` and `AnS.C` to `Tmp.L`;
if `AnS.C` $=$ `BwS.C` $= 1$ **then** set both to 0;

3.2.2 Symmetric signal-rule

The symmetric signal-rule (SSR) is obtained by combining an ASR pointing to the right, with a second ASR pointing to the left so that the attraction works in both directions. This comes at the cost of doubling the number of variables. We indicate by 1 and 2 the ASR pointing respectively to the right and to the left so that the state of the automaton at the beginning of an iteration on site $\texttt{X} \in \{\texttt{L}, \texttt{C}, \texttt{R}\}$ is given by the values `Var.B.X` for $\texttt{Var} \in \{\texttt{Def}, \texttt{FwS}, \texttt{BwS}, \texttt{AnS}, \texttt{Sta}\}$ and $\texttt{B} \in \{1, 2\}$. The variables of each ASR evolve simultaneously and independently according to Algorithm 3, with the exception of the two correction steps described in the algorithms below. The first one corresponds to the recombination of neighboring defects while the second one deals with displacing defect receiving forward-signals. In the two cases the modification aims at building agreement between the two ASR to avoid correcting twice the same error. First, we ensure that adajcent defect are still merged by avoiding that the two ASR cancel each other correction.

Algorithm 5: SSR update rule - Correct neighboring defects

if $(\texttt{Def.1.L}, \texttt{Def.1.C}, \texttt{Def.1.R}) = (0, 1, 1)$ **then**
    Set `Tmp.1.C` to 1;
if $(\texttt{Def.2.L}, \texttt{Def.2.C}, \texttt{Def.2.R}) = (1, 1, 0)$ **then**
    Set `Tmp.2.C` to 1;
if $\texttt{Tmp.1.C} = 1$ **or** $\texttt{Tmp.2.R} = 1$ **then**
    Set `Cor.C` to 1; set `Def.1.C` and `Def.2.C` to 0;
if $\texttt{Cor.L} = 1$ **then**
    Set `Def.1.C` to 0;
if $\texttt{Cor.R} = 1$ **then**
    Set `Def.2.C` to 0;
Set `Tmp.1.C` and `Tmp.2.C` to 0;

And similarly, a defect receiving two forward-signals (one from the right, one from the left) is left still while said forward-signals are reflected into backward-signals.

Algorithm 6: SSR update rule - Correction and signals reflection

if $\texttt{Def.1.C} = \texttt{FwS.1.C} = 1$ **then**
    Set `Tmp.1.C` to 1 ;
    Set (`FwS.1.C`, `BwS.1.C`) to $(0, 1)$ **if** $(\texttt{FwS.1.C}, \texttt{BwS.1.C}) = (1, 0)$;
if $\texttt{Def.2.C} = \texttt{FwS.2.C} = 1$ **then**
    Set `Tmp.2.C` to 1;
    Set (`FwS.2.C`, `BwS.2.C`) to $(0, 1)$ **if** $(\texttt{FwS.2.C}, \texttt{BwS.2.C}) = (1, 0)$;
if $\texttt{Tmp.1.C} + \texttt{Tmp.2.R} = 1$ **then**
    Set `Cor.C` to 1; set `Def.1.C` and `Def.2.C` to 0;
if $\texttt{Cor.L} = 1$ **then**
    Set `Def.1.C` to 1 ;
    Set (`FwS.1.C`, `BwS.1.C`) to $(0, 1)$ **if** $(\texttt{FwS.1.C}, \texttt{BwS.1.C}) = (1, 0)$;
if $\texttt{Cor.R} = 1$ **then**
    Set `Def.2.C` to 1 ;
    Set (`FwS.2.C`, `BwS.2.C`) to $(0, 1)$ **if** $(\texttt{FwS.2.C}, \texttt{BwS.2.C}) = (1, 0)$;

3.2.3 Erasure of a finite error configuration

Because the ASR is simpler to analyze formally, we chose to consider this variant for the threshold proof. The main step in the proof of Theorem 9 is to show that on an infinite lattice $\mathbb{Z}$, the ASR erases any finite-size error $E \subset \mathbb{Z}$ of width Δ in $\mathcal{O}(\Delta)$ time steps. The case of a connected error pattern, illustrated on Figure 3.1 (b), is the simplest one. The initial defects are located at sites σ_1 and $\sigma_2 = \sigma_1 + \Delta$. The first forward-signal emitted from σ_1 reaches σ_2 at time $t = \Delta$, the defects recombine at time $t = 2\Delta$, and the choice $k_a \geq 2$ ensures that all forward-signals have been caught by anti-signals at time $t \leq 3\Delta$, guaranteeing that the cellular automaton has reached the zero-error configuration.

In general, an error may consist of $m \geq 1$ clusters, giving rise to m pairs of defects $\Sigma = \{\sigma_1, ..., \sigma_{2m}\} \subset \mathbb{Z}$ with $\sigma_1 < ... < \sigma_{2m}$. These defects can interact via the exchange of signals, leading to a complex dynamics, as depicted on Figure 3.1 (a). We show that all defects and excitations (either signals or stack increments) eventually recombine in a time linear in the width $\Delta := \sigma_{2m} - \sigma_1$ of the error. Note that the leftmost defect σ_1 never moves when working on an infinite lattice. We give a brief outline of the proof here, with the full proof deferred to Section 3.4. The proof works by a reduction to the single cluster case. The main idea is to define space-time *interaction frontier* (depicted in green in Figure 3.1 and defined in Section 3.4) such that below this frontier, the odd-numbered defects remain immobile and recombine with their right neighbor, as if there was only a single pair of defects. We show that the frontier reaches the last defect in time $\mathcal{O}(\Delta)$.

It is left to prove that after all defects have recombined, all excitations recombine too in time linear in the width Δ of the error so that the decoder converges correctly to the zero configuration. This follows from a conservation law between excitations: we assign a $+1$ charge to every forward-signal and backward-signal, and a -1 charge to every anti-signal and stack increment, while the defects don't carry any charge. We prove that the total charge within the automaton is conserved (a corollary of which the stack on a given site is bounded by $2n$ on a finite size lattice) and that each positive charge can be paired up with a negative charge on its left. Since stack increments eventually transforms into anti-signals in the absence of defects, choosing $k_a \geq 2$ ensures that the anti-signals will catch forward-signals, and that all excitations will eventually recombine. Note that the same conservation law holds for the two

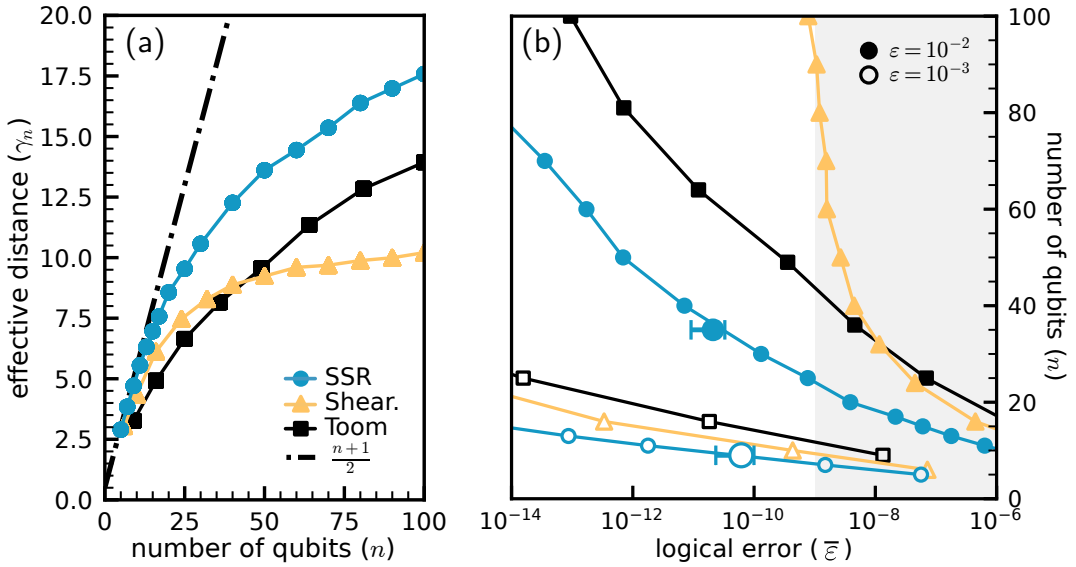

Figure 3.5: (a) Effective distance γ_n as a function of n for the SSR and Toom's rule. The dashed line represents the scaling of the repetition code in $\frac{n+1}{2}$ when decoded using minimum-weight perfect matching [53, 49]. (b) Estimate of the number of physical qubits necessary to reach a given logical error rate. The estimates are made from the ansatz $An(B\varepsilon)^{\gamma_n}$ with a different parameter γ_n for each n, fitted on the grey region, see Figure 3.2. Larger points with error bars in the white region are obtained numerically and confirm the validity of the estimates in that regime. The lines in both plots are for visual guidance only and do not represent actual data.

ASR making up the symmetric signal-rule.

3.2.4 Performance and ressource requirements

We plot the logical error rate as a function of the physical error rate and n in Figure 3.2 (a), and as a function of the data error and measurement error probabilities ε_d and ε_m in Figure 3.2 (b). We observe in the latter that the logical error rate depends on a linear function of ε_d and ε_m which confirms the intuition that, in a local decoding scheme, measurement errors and data errors play similar roles.

We compare the SSR decoder with a variant of Toom's rule defined on a flat 2D surface, and with the *shearing-rule* introduced in the last chapter. The SSR decoder experiences fractal-like error configurations leading to logical failure. This is also the case of the ASR, and we present an example for the more restricted code-capacity model in Figure 3.6 (b). Such configurations are respon-

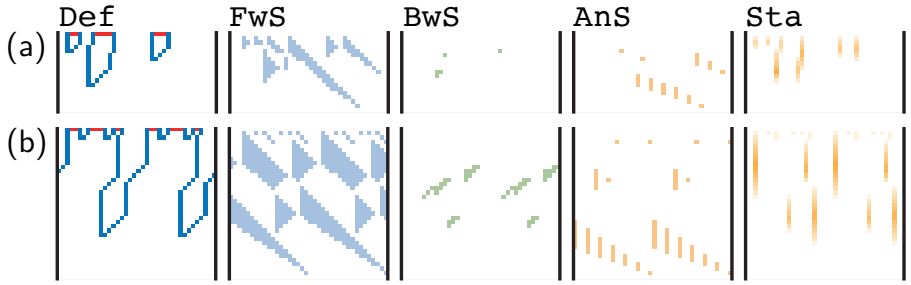

Figure 3.6: (a) Complete space-time representation of each variable during the correction of the error configuration illustrated in Figure 3.1 and (b) of a fractal-like error configuration of weight $w < n/2$ resulting in a logical error. Binary variables are represented with the usual colors and we use a color gradient for the stack variable.

sible for the sublinear scaling of the effective distance and are characteristic of renormalization-group decoders [132]. We assess the practical performance of the various decoders by fitting $\bar{\varepsilon}$ with an ansatz of the form $An(B\varepsilon)^{\gamma_n}$, where the exponent γ_n is allowed to depend on n: see Figure 3.5 (a). For the SSR decoder, we obtain $A = 2.1 \times 10^{-3}$ and $B^{-1} = 6.6\%$. Note that comparing γ_n to $(n+1)/2$ helps assess the performance loss relative to global decoders like minimum-weight perfect matching that decode up to the optimal distance. The behavior of γ_n as a function of n for the SSR decoder provides evidence of the exponential suppression of the logical error rate with increasing system size in the asymptotic regime. In contrast, γ_n saturates from $n \simeq 30$ onward for the shearing-rule. For small system sizes, however, numerical simulations show that the SSR and Shearing-rule decoders exhibit similar performance, both outperforming Toom's rule.

Every global decoder can be implemented as a local decoder if given enough time, at the cost of considerable local classical memories. We show here that only a few bits on each automation site are required for the ASR and SSR decoders to work. The ASR requires one binary register for each of the four types of point-like excitations, a register able to store an integer to represent the stack – typically on a logarithmic number of bits with binary encoding – and a few auxiliary bits for information transfer and local computation. This material cost simply doubles for the SSR, and we numerically evaluate below the size of the stack register that is needed in practice.

Consider a sequence of configurations in the phenomenological model upon application of the SSR decoder. Let $M^{(t)} \leq 2n$ be the random variable corre-

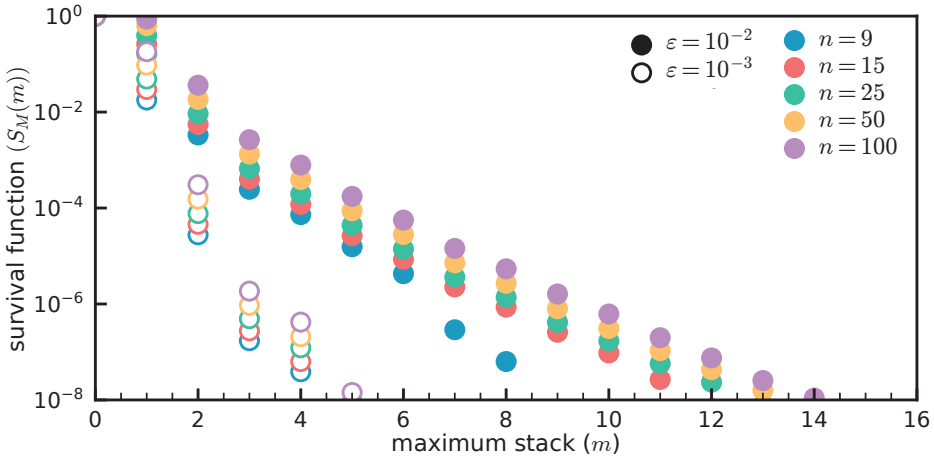

Figure 3.7: Survival function of the maximum stack height M of a configuration upon successive SSR applications in the phenomenological model. $S_M(m) := \mathbb{P}(M \geq m)$ is estimated from Monte-Carlo simulations, and shown for various system sizes and physical error rates. Binary encoding of the stack further reduces the practical memory requirements to $\log m$

sponding to the maximum value held in any of the $2n$ stacks (ASR pointing to the right or left, for each site) at time $t \geq 0$. For a Markovian dynamics, $M^{(t)}$ quickly converges to the time-independent associated random variable M. We estimate its survival function $S_M(m) := \mathbb{P}(M \geq m)$ from $M^{(t)}$ for $t \leq \tau = 1000$ on $\geq 10^6$ trajectories, and where τ is large compared to the convergence time of the normalized logical flip probability, see SM. We plot a numerical estimate of $S_M(m)$ for different physical error rates and system sizes in Figure 3.7, and we observe that $S_M(m)$ is exponentially suppressed in m in the regime $m = \mathcal{O}(n)$. The variable $S_M(m)$ can be used to estimate the maximum value that a stack should be able to encode to achieve a given logical error rate. For example, in the worst case considered in the simulations ($n = 100$ and $\varepsilon = 10^{-2}$), the SSR decoder only requires stack with values ≤ 14 to reach a logical error rate of 10^{-8}. This can be done on 4 bits for each of the two stacks of a site.

3.3 Numerical simulations

3.3.1 Markovian dynamics within the phenomenological model

In usual decoding schemes, the information from a stabilizer measurement is erased after some time usually linear in the distance of the code so that the entropy is evacuated from the scheme. This is not the case with the ASR and SSR decoders where a forward-signal, i.e. a previous odd parity measurement, a priori does not have a definite lifetime. This makes the markovianity of the dynamics non-trivial while this property is required in order to define a constant logical error rate. We confirm the markovianity of the dynamics from numerical simulations.

The automaton is initialized in the zero configuration and we denote by $\overline{P}(\tau)$ the probability of a logical flip (in this case defined by a majority of 1) at time τ. We observe numerically in Figure 3.8 (a) that $\overline{P}(\tau)$ follows asymptotically a Poisson-like behavior, i.e. of the form $\overline{P}(\tau) \simeq [1 - (1 - 2\overline{\varepsilon})^{\tau}]/2$ for some $\overline{\varepsilon} > 0$ defining the logical error rate. Note that in the absence of anti-signals ensuring a balance between excitations, the occurrence rate of logical failure would typically increase with time as signals accumulate within the system.

The logical failure of the decoder results in the general case from space-time error clusters. Hence the simulation time should be large enough to account for the contribution of the most likely error clusters resulting in a logical flip. This numerically corresponds to the convergence of the normalized logical flip probability $\overline{P}(\tau)/\tau$ towards the logical error rate, which is shown in Figure 3.8 (b). We define the convergence time τ_n for n qubits to be the minimum time after which $\overline{P}(\tau)/\tau > \overline{\varepsilon}/2$, and we show τ_n as a function of n in Figure 3.8 (c). Numerical fits give $\tau_n = \mathcal{O}(n)$, and at the exception of high logical error rates close to the threshold, numerical simulations use $\tau = 1000$ which is one order of magnitude greater than any expected convergence time. Note that since the logical error rate is normalized, longer simulation time does not reduce numerical performances.

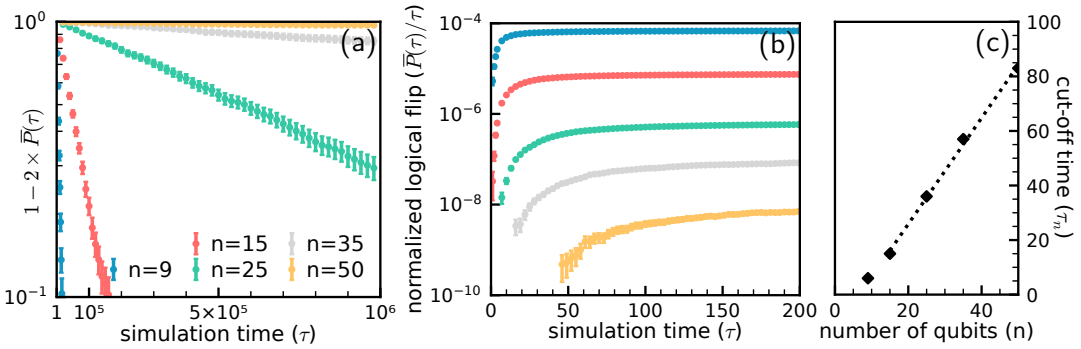

Figure 3.8: (a) Population in a majority of 0 states, i.e. $1-2\overline{P}(\tau)$, as a function of τ, a constant slope in logarithmic scale indicates a logical flip probability independent of the time of the simulation. (b) $\overline{P}(\tau)/\tau$ as a function of τ. The logical error rate $\overline{\varepsilon}$ is estimated in the asymptotic regime where $\overline{P}(\tau)/\tau$ is a constant (c) The average convergence time of the automaton defined as $\tau_n := \min\{\tau_0 \geq 0 \mid \overline{P}(\tau)/\tau > \overline{\varepsilon}/2, \tau \geq \tau_0\}$ as a function of n. We fit τ_n with the linear ansatz $an + b$ and we find $a = 1.95$ and $b = -13$.

3.4 Asymmetric signal-rule on an infinite lattice

The main step in the proof of Theorem 11 (proven in Section 3.5) is to show that on an infinite lattice $\mathbb{Z}$, the ASR erases any finite-size error $E \subset \mathbb{Z}$ of width Δ in $\mathcal{O}(\Delta)$ time steps. Recall that the use of infinite lattices is merely a proof technique to derive the threshold theorem for the finite case (Theorem 11). Let $u \in U := (\mathbb{Z}_2^4 \times \mathbb{N})^{\mathbb{Z}}$ be a configuration of the automaton, and $\Sigma^{(t)} := \Sigma(u^{(t)})$ be the set of defects of the configuration of the automaton at time $t \geq 0$. We establish the following erasure theorem:

Theorem 10 (Linear Erasure)**.** *Let $m \geq 1$ and $\Sigma := \{\sigma_1, \ldots, \sigma_{2m}\}$ with $\sigma_1 < \cdots < \sigma_{2m}$. Let $\Delta := \sigma_{2m} - \sigma_1$ and $u^{(t)}$ initialized with $\Sigma^{(0)} = \Sigma$ and all other variables set to zero. We have for all $t \geq 0$,*

$$\Sigma^{(t)} \subset [\sigma_1, \sigma_1 + \Delta], \quad \operatorname{supp}(u^{(t)}) \subset [\sigma_1, \sigma_1 + 78\Delta]. \tag{3.1}$$

and for all $t > 77\Delta$,

$$\operatorname{supp}(u^{(t)}) = \varnothing. \tag{3.2}$$

Importantly, the decoder returns to the zero configuration after the initial error is corrected, that is to say all excitations have recombined. We introduce the *interaction frontier* in Section 3.4.1 that is central to the proof of the theorem, provided in Section 3.4.2. We establish some properties of the interaction

frontier in Section 3.4.5 and 3.4.6 that prove that the initial error is eventually corrected. It is left to ensure that all excitations recombine which follows from the properties of the charge distribution within the automaton that are proven in Section 3.4.4.

3.4.1 Interaction frontier

The proof uses the non-decreasing space-time *interaction frontier* $\varphi^{(t)} \in \mathbb{Z}$ defined for $t \leq t_r$, where t_r will be specified later. Below this frontier, the odd defects remain immobile and recombine with their right neighbor, as if there was only a single pair of defects. We introduce the set of forward signals (denoted $\Phi_F^{(t)} := \Phi_F(u^{(t)})$) and the set of negative charges — i.e., the locations of non-zero stacks or anti-signals — (denoted $\Phi_N^{(t)}$), both defined analogously to $\Sigma^{(t)}$ for defects. The interaction frontier $\varphi^{(t)}$ is defined recursively from $\varphi^{(0)} = \min(\Sigma)$ by

$$\varphi^{(t+1)} = \begin{cases} \varphi^{(t)} + 1, & \text{if } A \\ \varphi^{(t)}, & \text{if } \neg A \text{ and } B \\ \min((\Sigma^{(t+1)} \cup \Phi_F^{(t+1)}) \cap \mathbb{Z}_{>\varphi}^{(t)}), & \text{if } \neg A \text{ and } \neg B \end{cases} \tag{3.3}$$

where we use the notations $\mathbb{Z}_{>\varphi}^{(t)} = (\varphi^{(t)}, +\infty)$ and

$A : \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$ (the right neighbor of the frontier hosts either a defect or a forward signal)

$B : |\Sigma^{(t)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}| \equiv 1 \pmod 2$ (there is an odd number of defects on the left of the frontier)

The definition remains valid until some time $t_r \geq 0$ such that $\Sigma^{(t_r+1)} \cap \mathbb{Z}_{>\varphi^{(t_r)}} = \varnothing$, i.e. until there are no defects remaining on the right of the frontier. Intuitively, the interaction frontier follows the propagation of a forward-signal until the said forward-signal recombine or reaches a defect. In such case, the forward-signal is attached to a defect or another forward-signal depending on the parity of the number of defects on the left of the frontier. The interaction frontier is illustrated in Figure 1a of the main text that we also include here. Note that in the figure we represent independently two interaction frontiers for two error clusters as if they were on separate lattices.

3.4.2 Proof of Erasure

We start by a brief outline of the proof. We show that the interaction frontier reaches the last defect in a time linear in the initial error width (using Lemma 4, illustrated in Figure3.9 and proven in Section 3.4.6). At this point all defects are on the left of the interaction frontier. This region is characterized so that pairs of defect recombine independently (Lemma 5, illustrated in Figure 3.10 and proven in Section 3.4.5). It is left to prove that the dynamics of the automaton induces the recombination of all remaining excitations (Lemma 6, illustrated in Fig 3.11 proven in Section 3.4.4).

Before formally stating the relevant lemmas, we introduce some additional notations. Let Σ_1 and Σ_2 denote the sets of odd and even defects, respectively. We consider the two possible unions of integer open intervals formed between the even and odd defects, depending on whether the odd defects are positioned on the left or the right.

$$\Sigma_{k,k+1} := \bigcup_{\substack{1\leq i\leq 2m-1 \\ i\equiv k \,(\mathrm{mod}\ 2)}} (\sigma_i, \sigma_{i+1}). \tag{3.4}$$

We adopt the abbreviated notation $\Sigma_{12} := \Sigma_{1,2}$ and $\Sigma_{21} := \Sigma_{2,3}$ hereafter. Note that the integer interval $[\min(\Sigma), \max(\Sigma)]$ is partitioned as follows:

$$[\min(\Sigma), \max(\Sigma)] = \Sigma_1 \sqcup \Sigma_2 \sqcup \Sigma_{12} \sqcup \Sigma_{21}. \tag{3.5}$$

All these notations naturally extend to incorporate time dependence. With these definitions in place, we are now prepared to introduce the key lemmas required for the proof of Theorem 10.

Lemma 4 (From one defect to another)**.** *Let $t_1 \geq 0$ such that $\varphi^{(t_1)} \in \Sigma^{(t_1)}$ and $\Sigma^{(t_1)} \cap \mathbb{Z}^{(t_1)}_{>\varphi} \neq \varnothing$, there exists $t_2 \in (t_1, t_1 + 3(\sigma_{2m} - \varphi^{(t_1)}))$ such that $\Sigma^{(t_2)} \cap \mathbb{Z}^{(t_2)}_{>\varphi} = \varnothing$ or $\varphi^{(t_2)} \in \Sigma^{(t_2)}$ with $\varphi^{(t_2)} - \varphi^{(t_1)} \geq (t_2 - t_1)/11$.*

Lemma 4 states that the interaction frontier exhibits a coarse-grained nonzero velocity: although the frontier may remain stationary over several iterations, its average speed over time is lower-bounded by 1/11. This guarantees that the interaction frontier will eventually reach the right-most defect.

Lemma 5 (Independent matching)**.** *Let $t \geq 0$ and $\varphi^{(t)} \in \mathbb{Z}$ be the interaction*

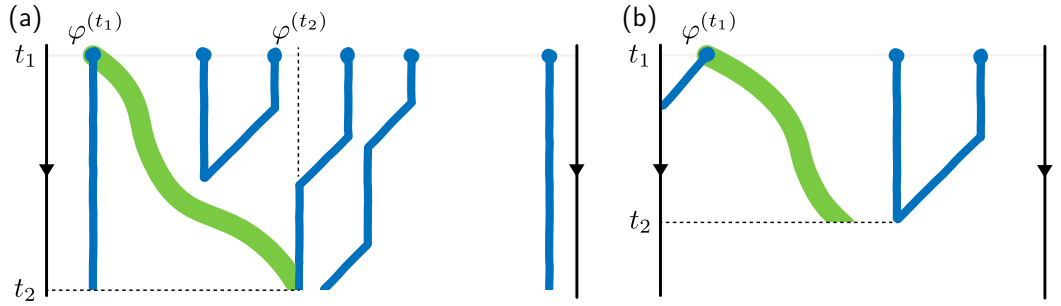

Figure 3.9: Illustration of the two possible cases of Lemma 4 that characterize the increase of the interaction frontier depicted in green. (a) The interaction frontier goes from $\varphi^{(t_1)} \in \Sigma^{(t_1)}$ to $\varphi^{(t_2)} \in \Sigma^{(t_2)}$ in time $t_2 - t_1 \leq 11(\varphi^{(t_2)} - \varphi^{(t_1)})$. (b) At time t_2 we have $\Sigma^{(t_2)} \cap \mathbb{Z}^{(t_2)}_{>\varphi} = \varnothing$. The repeated application of Lemma 4 implies that at some point all defects are on the left of the interaction frontier.

frontier, the following holds

$$\left(\Sigma^{(t)}_{12} \cap \mathbb{Z}^{(t)}_{\leq\varphi}\right) \subset \Phi^{(t)}_F, \tag{3.6}$$

$$\left(\Sigma^{(t)}_{12} \cap \Phi^{(t)}_N \cap \mathbb{Z}^{(t)}_{\leq\varphi}\right) = \varnothing, \tag{3.7}$$

$$\left(\Sigma^{(t)}_{21} \cap \Phi^{(t)}_F \cap \mathbb{Z}^{(t)}_{<\varphi}\right) = \varnothing. \tag{3.8}$$

In addition, for all $\sigma_a \in \Sigma^{(t)}_2, \sigma_b \in \Sigma^{(t)}_1$ *such that* $\sigma_a < \sigma_b \leq \varphi^{(t)}$*, we have*

$$\sigma_b - \sigma_a \geq 2. \tag{3.9}$$

Here we characterize the region on the left of the interaction frontier (see Figure 3.10) so that the interval between successive even and odd defects if filled with forward-signals, while the converse type of interval is without forward-signals and of length at least 2. This ensures that even and odd defects on the left of the frontier recombine independently.

Lemma 6 (Excitations recombination)**.** *Let* $t_r \geq 0$*,* $z \in \mathbb{Z}$ *and* $\delta \geq 0$ *such that*

$$\Sigma^{(t_r)} = \varnothing \text{ and } \operatorname{supp}(u^{(t_r)}) \subset [z, z+\delta] \tag{3.10}$$

then for all $t \geq t_r$ *we have*

$$\operatorname{supp}(u^{(t)}) \subset [z, z+6\delta] \tag{3.11}$$

Figure 3.10: Illustration of Lemma 5, which characterizes the region to the left of the interaction frontier (shown in green). The open interval between an even and the subsequent odd defect is filled with forward signals and contains no negative charges. In contrast, the open interval between an odd and the following even defect has length at least 2 and contains no forward signals. This structure ensures that even defects are isolated from interference originating on the left, allowing defect pairs to recombine independently.

and for all $t \geq t_r + 5\delta$ *we have*

$$\operatorname{supp}(u^{(t)}) = \varnothing. \tag{3.12}$$

Lemma 6 states that once all defects have recombined, all excitations eventually recombine in linear time with distance between the left-most and the right-most excitation at that time. In this sense, the correction of an error cluster occurs through a build-up of excitations within the automaton, which later recombine to restore the system to its initial zero configuration.

Proof of Theorem 10. The proof is illustrated in Figure 3.11. Lemma 4 implies that there exists $t_q > 0$ such that $\Sigma^{(t_q)} \subset \mathbb{Z}^{(t_q)}_{\leq\varphi}$. In order to see this, we denote by $\{t_i\}_{i\geq 1}$ with $t_1 = 0$ the strictly increasing set of times such that $\varphi^{(t_i)} \in \Sigma^{(t_i)}$. The application of Lemma 4 for each t_i gives

$$\begin{aligned} t_k =& \sum_{i=1}^{k-1} (t_{i+1} - t_i) && (3.13)\\ \leq& 11 \sum_{i=1}^{k-1} (\varphi^{(t_{i+1})} - \varphi^{(t_i)}) \leq 11\Delta. && (3.14) \end{aligned}$$

This upper bound implies that the sequence necessarily stops for a large enough $k = q - 1$. With Lemma 4, this means that there exists $t_q \leq 11\Delta$ such that $\Sigma^{(t_q)} = \varnothing$ or $\Sigma^{(t_q)} \cap \mathbb{Z}^{(t_q)}_{>\varphi} = \varnothing$.

At this point, all defects are either removed, or are to the left of the interaction frontier. By Lemma 5, the open interval between odd and even defects is filled with forward-signals, while the interval between even and odd defects has length at least 2 and contains no forward-signals. Consequently, each odd de-

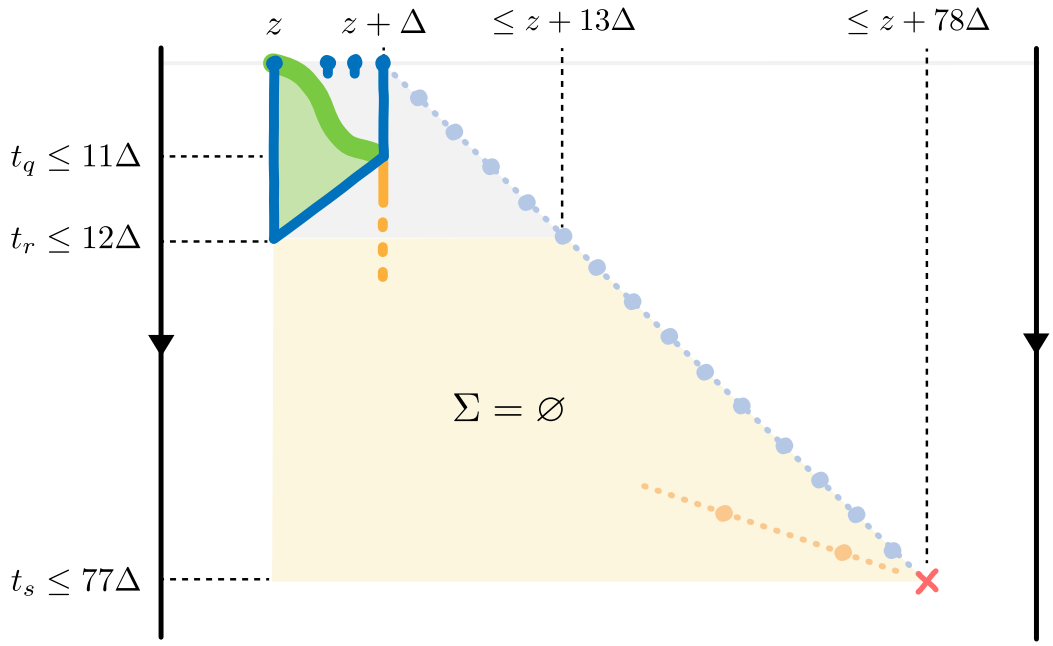

Figure 3.11: Illustration of the proof of Theorem 10 (not to scale). Let Σ be a set of defects with even cardinality and diameter Δ. By applying Lemma 4, we show that the interaction frontier reaches the final defect by time $t_q \leq 11\Delta$. At this point, the region to the left of the frontier (depicted in light green) satisfies the conditions of Lemma 5, ensuring that all defects recombine by time $t_r \leq 12\Delta$. The subsequent evolution of the automaton (yellow area), after defect recombination, is governed by Lemma 6, which bounds the time of the last excitation recombination by $t_s \leq 77\Delta$. In the worst case, the final forward signal to recombine may have propagated a distance of at most 77Δ from position $z + \Delta$, i.e., up to $z + 78\Delta$.

fect remains stationary, while the corresponding even defect moves leftward at each iteration until they recombine. This ensures that the automaton becomes defect-free at time $t_r = t_q + \Delta$. Finally, the rightmost part of the support of $u^{(t)}$ is bounded by the position of a forward-signal emitted from σ_{2m} at the first iteration, which has propagated for t_r steps. Applying Lemma 6 with $z = \sigma_1$ and $\delta = t_r + \Delta = 13\Delta$ completes the proof of Theorem 10. □

3.4.3 Charge conservation rules

We start by analyzing the dynamics of the automaton in terms of charged particles, which will be useful in the rest of the proof. We assign a $+1$ charge to every forward-signal and backward-signal, and a -1 charge to every anti-signal

and stack increment, while the defects don't carry any charge. Let $q_i(u)$ be the total charge on site $i \in \mathbb{Z}$ of configuration $u \in U$. We define $Q(u)$ and $Q_z(u)$ the total charge respectively on all sites of configuration u and on sites $< z \in \mathbb{Z}$. We use the following notation to account for time dependency

$$Q^{(t)} := Q(u^{(t)}) = \sum_{i \in \mathbb{Z}} q_i(u^{(t)}), \qquad Q_z^{(t)} := Q_z(u^{(t)}) = \sum_{i<z} q_i(u^{(t)}). \quad (3.15)$$

We summarize the key charge properties of the automaton in three Facts stated below, with proof given at the end of the subsection.

Fact 1 (Charge deficit)**.** *For all $t \geq 0$ we have*

$$Q^{(t)} = 0, \quad (3.16)$$

and for all $z \in \mathbb{Z}$,

$$Q_z^{(t)} \leq 0. \quad (3.17)$$

Fact 1 states that the global charge is conserved within the automaton, and that every positive charge can be paired up with a corresponding negative charge on its left. Note that the former property is also true on a finite lattice of size n, a corollary of which is that the stack variable on a given site is upper bounded by $2n$.

Fact 2 (No recombination across defects)**.** *Let $t \geq 0$ and $\sigma_a^{(t)} < \sigma_b^{(t)} \in \Sigma^{(t)}$ be a pair of successive defects. No charge at time t in the interval $[\sigma_a^{(t)}, \sigma_b^{(t)})$ recombines with a charge from $\mathbb{Z} \setminus [\sigma_a^{(t)}, \sigma_b^{(t)})$ during iteration t.*

Fact 2 states that charges on the two sides of a defect at the beginning of an iteration cannot recombine together during said iteration.

Fact 3 (Charge between defects)**.** *For all $t \geq 0$ and $\sigma \in \Sigma^{(t)}$ we have*

$$Q_\sigma^{(t)} = 0. \quad (3.18)$$

Finally, Fact 3 states that the total charge between two defects is always zero. Note that this also directly imply that the total charge in left open interval between two defects is always null. The proof of Lemma 6 follows directly from Fact 1: every positive charge can be paired up with a corresponding negative

charge on its left. Since stack increments eventually transforms into anti-signals in the absence of defects, and since anti-signals propagates faster than forward-signals, all excitations will eventually recombine. Facts 2 and 3 will be used in the proof of Lemmas 5 and 6. We give the proof of all facts here.

Proof of Fact 1. 3.16 follows directly from the creation and recombination rules that always include both a $+1$ and a -1 charge. 3.17 follows from the fact that $+1$ and -1 charges are created on the same site, but a -1 charge cannot bypass a $+1$ charge to its right, because anti-signals recombine with any $+1$ charge they encounter, and a $+1$ charge cannot bypass a -1 charge to its left, because backward-signals recombine with any -1 charge they encounter. □

Proof of Facts 2 and 3. Fact 3 is true at time $t = 0$. We prove that if Fact 3 holds at time t then Fact 2 is true for iteration t (Step 1), and that if Fact 3 holds at time t and Fact 2 is true for iteration t then Fact 3 holds at time $t+1$ (Step 2), finishing the proof by immediate recursion.

Step 1. We start by proving that if Fact 3 holds at time t then Fact 2 holds for iteration t, i.e. here we assume that the total charge between two defects was null at time t, and we want to prove that charges do not recombine accross defects. Let $\sigma_a^{(t)} < \sigma_b^{(t)} \in \Sigma^{(t)}$ and $\sigma_a^{(t+1)} < \sigma_b^{(t+1)} \in \Sigma^{(t+1)}$ be a pair of successive defects respectively at time t and $t+1$. Note that such identification is not ambiguous because the automaton either recombines defects by pair or displaces defects by one to the left. All types of charge recombinations across a defect during iteration t in chronological order are

- (i) A backward-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ recombining with an anti-signal from $\mathbb{Z}_{<\sigma_a}^{(t)}$.
- (ii) A backward-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ recombining with a stack increment from $\mathbb{Z}_{<\sigma_a}^{(t)}$.
- (iii) An anti-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ recombining with a forward-signal from $\mathbb{Z}_{\geq\sigma_b}^{(t)}$.

We omit the reciprocal event for each recombination type, i.e. the analogous process occurring in the adjacent defect interval, since such cases are symmetric and follow by identical arguments. It is clear that since Fact 3 holds at time t, Fact 1 implies that a backward-signal propagating to the left will encounter a negative charge from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ before it can exit. Since negative charges cannot

move to the left and since the automaton checks after each backward-signals displacement whether recombination with negative charges are possible, the backward-signal will recombine before exiting $[\sigma_a^{(t)}, \sigma_b^{(t)})$ and no recombination with a negative charge from $\mathbb{Z}_{<\sigma_a}^{(t)}$ is possible. We have ruled out events (i) and (ii) during iteration t. Since a positive charge within $[\sigma_a^{(t)}, \sigma_b^{(t)})$ will always remain on the left of a forward-signal from $\mathbb{Z}_{\geq\sigma_b}^{(t)}$, a similar argument rules out event (iii) and finishes the proof of Fact 2 for iteration t.

Step 2. Now let us assume that Fact 3 holds at time t and that Fact 2 is true for iteration t, i.e. the total charge between two defects was null at time t and no charge recombination have occurred across the defects during iteration t, and we want to prove that the total charge between two defects remains null at time $t+1$. In the absence of recombination across defects, the remaining possible violations of Fact 3 at time $t+1$ correspond to charges from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ outside of $[\sigma_a^{(t+1)}, \sigma_b^{(t+1)})$ at time $t+1$. Omitting reciprocal events similarly as in Step 1, the four possible cases are

(i) A backward-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ in $\mathbb{Z}_{<\sigma_a}^{(t+1)}$ at time $t+1$.

(ii) A forward-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ in $\mathbb{Z}_{\geq\sigma_b}^{(t+1)}$ at time $t+1$.

(iii) An anti-signal from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ in $\mathbb{Z}_{\geq\sigma_b}^{(t+1)}$ at time $t+1$.

(iv) A stack from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ in $\mathbb{Z}_{\geq\sigma_b}^{(t+1)}$ at time $t+1$.

First, notice that no mechanism allows a negative charge (which can only move to the right) to bypass a defect (which can only move to the left) to its left within an iteration. Since in addition because of Fact 2 a negative charge from $[\sigma_a^{(t)}, \sigma_b^{(t)})$ cannot have recombined with a positive charge from $\mathbb{Z}_{<\sigma_a}^{(t)}$, a backward-signal will always recombine before bypassing a defect to its left which forbids (i). This implies that the site of a defect and the site on its left are without backward-signals. Consequently, forward-signals encountering a defect always transform into backward-signals, which forbids (ii). A corollary of forbidding (ii) is that all positive charges between two successive defects remain on the left of the right defect. Therefore, the same argument as for (i) but in the other direction rules out cases (iii) and (iv). This finishes the proof. □

3.4.4 Recombination of all excitations after correction

This section is devoted to the proof of Lemma 6 that states that in a defect-free configuration, the successive application of the ASR ensures that all excitations eventually recombine. The key ingredient in the proof is Fact 1 that states that the global charge within the automaton is conserved and equal to zero, and that every positive charge can be paired with a negative charge on its left. In this situation, the higher speed of anti-signals ensures that all signals and stacks recombine in the end, which gives Lemma 6.

Proof of Lemma 6. Let $t_r \geq 0$, $z \in \mathbb{Z}$ and $\delta \geq 0$ be such that $\Sigma^{(t_r)} = \varnothing$ and $\mathrm{supp}(u^{(t_r)}) \subset [z, z+\delta]$: no new charge can be created at this point. Because the positive charge on a given site is bounded by 2 (one forward-signal and one backward-signal) the total positive charge of the automaton at time t_r is upper bounded by $2(\delta+1)$, and Fact 1 directly implies that the total number of negative charges is upper bounded by the same quantity. This negative charge is distributed between stacks and anti-signals.

For $t \geq t_r$, we consider the rightmost site with a non-empty stack. At least one anti-signal leaves this site at each iteration, since either the site is occupied by an anti-signal and the stack cannot decrement, or the stack decrements by 1 and creates an anti-signal propagating to the right. The cumulative sum of this quantity from iteration t_r on is however ultimately bounded by the number of negative charges within the automaton at time t_r, previously upper bounded by $2(\delta+1)$. This directly implies that all stacks are necessarily empty at time $t_i = t_r + 2(\delta+1)$. It is now left to upper bound the time of the last recombination between the last positive charge and the last anti-signal. In the worst case, this corresponds to an anti-signal located on site z at time t_i, and the rightmost forward-signal on site $z+\delta$ at time t_r. Since anti-signal propagates by 3 at each iteration the recombination occurs at time t_s such that

$$3(t_s - t_i) = \delta + (t_s - t_r), \tag{3.19}$$

which gives $t_s - t_r \leq \lceil (7\delta+6)/2 \rceil \leq 5\delta$ if $\delta \geq 3$, and it can be checked independently that the bound still works if $\delta \leq 2$. During this time the right most forward-signal has propagated by at most 5δ, this finishes the proof. □

3.4.5 Defect recombination on the left of the interaction frontier

This section is devoted to the proof of Lemma 5 that states that on the left of the interaction frontier, an open interval between two successive odd and even defects in that order is filled with forward-signals but without negative charges (stack increments or anti-signals), while the open interval in the converse order is without forward-signals. Since an even defect is also separated by a distance of at least two with the next odd defect, and since forward-signals propagate by 1 at each iteration, the two defects do not interact. This means that even defects are displaced by 1 to the left at each iteration, until they recombine with the odd defect on its left that remains immobile. The operation takes a time equal to the distance between the two defects. The Lemma can be well understood graphically from Figure 3.1 (a) but the proof is quite technical and may be skipped on a first reading. In the proof, we use two facts on the interaction frontier that are proven in Section 3.4.7.

Proof of Lemma 5. The proposition is true for $t = 0$ where we have $\varphi^{(0)} = \min(\Sigma)$. Let us assume that Lemma 5 is true up to some time $t \geq 0$, i.e. we have

$$\left(\Sigma_{12}^{(t)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}\right) \subset \Phi_F^{(t)}, \tag{3.20}(t)$$

$$\left(\Sigma_{12}^{(t)} \cap \Phi_N^{(t)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}\right) = \varnothing, \tag{3.21}(t)$$

$$\left(\Sigma_{21}^{(t)} \cap \Phi_F^{(t)} \cap \mathbb{Z}_{<\varphi}^{(t)}\right) = \varnothing. \tag{3.22}(t)$$

and

$$\forall \sigma_a \in \Sigma_2^{(t)}, \sigma_b \in \Sigma_1^{(t)}, \text{ if } \sigma_a < \sigma_b \leq \varphi^{(t)}, \text{ then } \sigma_b - \sigma_a \geq 2. \tag{3.23}(t)$$

We want to prove that Lemma 5 still holds at time $t + 1$, i.e. $(3.20)(t + 1)$, $(3.21)(t + 1)$, $(3.22)(t + 1)$ and $(3.23)(t + 1)$. We decompose the proof by characterizing first the interval $\mathbb{Z}_{\leq\varphi}^{(t)}$ in Step 1, and the intervals $(\varphi^{(t)}, \varphi^{(t+1)}]$ and $[\varphi^{(t)}, \varphi^{(t+1)})$ in Step 2.

Step 1. First, let us argue that we still have at time $t + 1$

$$\left(\Sigma_{12}^{(t+1)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}\right) \subset \Phi_F^{(t+1)}, \qquad (3.20^*)(t+1)$$
$$\left(\Sigma_{12}^{(t+1)} \cap \Phi_N^{(t+1)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}\right) = \varnothing, \qquad (3.21^*)(t+1)$$
$$\left(\Sigma_{21}^{(t+1)} \cap \Phi_F^{(t+1)} \cap \mathbb{Z}_{<\varphi}^{(t)}\right) = \varnothing, \qquad (3.22^*)(t+1)$$

Notice that proving the Lemma would require $Z_{\leq\varphi}^{(t+1)}$ (resp. $Z_{<\varphi}^{(t+1)}$) instead of $Z_{\leq\varphi}^{(t)}$ (resp. $Z_{<\varphi}^{(t)}$). We also need

$$\forall \sigma_a \in \Sigma_2^{(t+1)}, \sigma_b \in \Sigma_1^{(t+1)}, \text{ if } \sigma_a < \sigma_b \leq \varphi^{(t)}, \text{ then } \sigma_b - \sigma_a \geq 2. \quad (3.23^*)(t+1)$$

We start by proving $(3.21^*)(t+1)$. Note first that the open interval was without negative charge at time t and since charges do no propagate across defects because of Fact 3 it is only left to verify that no negative charge on the left defects site enters the open interval. Since the left neighbouring sites of odd defects on the left of the interaction frontier are without defect or forward-signal at time t, either odd defects remain immobile and the associated stack does not decrement, or they recombine with a defect on its right in which case the stack is necessarily empty. In the two cases we have $(3.21^*)(t+1)$.

Now we turn to $(3.20^*)(t+1)$ and $(3.22^*)(t+1)$. Since forward-signals propagate by 1 at each iteration between two defects it is sufficient to consider what happens at the creation of the forward-signal on some defect $\Sigma^{(t)} \ni \sigma < \varphi^{(t)}$: the forward-signal should be erased if the defect is even, and it should keep propagating if the defect is odd. Intuitively, for isolated defects, even defects have a forward-signal on their left that induce the defect displacement, the associated stack then decrements into an anti-signal that erases the previously created forward-signal. Odd defects however remain immobile and forward-signals it created keep propagating. We formally discuss all cases in the following, including defects adjacent to each other. The two main cases are again:

$$D_1 : \sigma \in \Sigma_1^{(t)}, \qquad D_2 : \sigma \in \Sigma_2^{(t)}.$$

The subdivision of the case is as follows

D_1 Here $\Sigma_1^{(t)} \ni \sigma \leq \varphi^{(t)}$. $(3.23)(t)$ implies that $\sigma - 1 \notin \Sigma^{(t)}$: hence either $\sigma+1 \notin \Sigma^{(t)} \cup \Phi^{(t)}$, and the defect remains immobile and emits

a forward-signal, or $\sigma + 1 \in \Sigma^{(t)}$ and the two defects recombine.

D_2 Here $\Sigma_2^{(t)} \ni \sigma \leq \varphi^{(t)}$. $(3.20)(t)$ implies that $\sigma - 1 \in \Sigma^{(t)} \cup \Phi_F^{(t)}$. We distinguish between the cases (i) $\sigma+1 \notin \Sigma^{(t)}$ and (ii) $\sigma+1 \in \Sigma^{(t)}$. If (i), either $\sigma-1 \in \Phi_F^{(t)} \setminus \Sigma^{(t)}$ and the defect σ is displaced to the left, or $\sigma - 1 \in \Sigma^{(t)}$ and $(3.23)(t)$ implies that $\sigma - 2 \notin \Sigma^{(t)}$, so that the two defects σ and $\sigma - 1$ recombine. If (ii), either $\sigma - 1 \in \Phi_F^{(t)} \setminus \Sigma^{(t)}$ and the two defects σ and $\sigma + 1$ recombine, or $\sigma - 1 \in \Sigma^{(t)}$ which reduces to the subcase of (i) discussed above. In all cases an emitted forward-signal is erased by the anti-signal created from the stack decrement.

Note that because of $(3.23)(t)$ two adjacent defects recombining are necessarily an odd and an even defect in that order. In this case the left-open interval is the singleton $\{\sigma_a\}$ that is without forward-signal and backward-signal, which implies the absence of negative charge by Fact 3. The recombination of the two defect then leaves the corresponding sites empty which verifies $(3.22^*)(t+1)$.

It is left to prove $(3.23^*)(t+1)$. Since odd defects $\sigma_b \leq \varphi^{(t)}$ remain immobile unless they recombine, the only possible decrease of a distance between successive even and odd defects concerns an odd defect moving from $\varphi^{(t)} + 1$ at time t to $\varphi^{(t)}$ at time $t+1$. This violates $(3.23^*)(t+1)$ only if $\varphi^{(t)} - 1$ is also occupied by a defect at that time. The latter defect was either (i) on site $\varphi^{(t)} - 1$ at time t, or (ii) on site $\varphi^{(t)}$ at time t. (i) is not possible because the even defect would have been displaced to the left, and if (ii) the two defects would have recombined unless $\varphi^{(t)} - 1$ was occupied by a defect at time t, which is forbidden by Fact 4. All cases lead to a contradiction, this finishes the proof of $(3.23^*)(t+1)$.

Fact 4 (Impossible case)**.** *Let $t \geq 0$ and $\varphi^{(t)} \in \mathbb{Z}$ be the interaction frontier at time t, then $\{\varphi^{(t)} - 1, \varphi^{(t)}, \varphi^{(t)} + 1\} \not\subset \Sigma^{(t)}$.*

Step 2. It is now left to extend the result to the incremented region on the left of the interaction frontier, i.e. the interval between $\varphi^{(t)}$ and $\varphi^{(t+1)}$, i.e.

$$\left(\Sigma_{12}^{(t+1)} \cap (\varphi^{(t)}, \varphi^{(t+1)}]\right) \subset \Phi_F^{(t+1)}, \qquad (3.20^\dagger)(t+1)$$

$$\left(\Sigma_{12}^{(t+1)} \cap \Phi_N^{(t+1)} \cap (\varphi^{(t)}, \varphi^{(t+1)}]\right) = \varnothing, \qquad (3.21^\dagger)(t+1)$$

$$\left(\Sigma_{21}^{(t+1)} \cap \Phi_F^{(t+1)} \cap [\varphi^{(t)}, \varphi^{(t+1)})\right) = \varnothing. \qquad (3.22^\dagger)(t+1)$$

With the convention that for $z \in \mathbb{Z}$, $[z, z) = (z, z] = \varnothing$. We also need

$$\forall \sigma_a \in \Sigma_2^{(t+1)}, \sigma_b \in \Sigma_1^{(t+1)}, \text{ if } \sigma_a < \sigma_b \leq \varphi^{(t+1)}, \text{ then } \sigma_b - \sigma_a \geq 2. \quad (3.23^\dagger)(t+1)$$

We start by verifying that $(3.20^\dagger)(t+1)$, $(3.21^\dagger)(t+1)$ and $(3.22^\dagger)(t+1)$ hold for the three cases corresponding to the three possible updates of $\varphi^{(t)}$ to $\varphi^{(t+1)}$. Intuitively, the proof can be understood as follows. If A, the interaction frontier either follows the free propagation of a forward-signal or the frontier reaches a defect. In both cases, the previous position of the frontier is now filled by a forward-signal. If $\neg A$ and B, the incremented region is empty and all desired properties are trivially true. Finally, $\neg A$ and $\neg B$ corresponds to the recombination of the forward-signal to which was attached the interaction frontier in the interval between an even and an odd defect. In that case the interaction frontier is updated to the next defect or forward-signal so that the right-open incremented region is without such excitations.

We formally treat all cases in the following. Each case is subdivided into at most four cases corresponding to the position of $\varphi^{(t)}$ with respect to odd and even defects, using following notations

$$F_1 : \varphi^{(t)} \in \Sigma_1^{(t)}, \qquad F_2 : \varphi^{(t)} \in \Sigma_2^{(t)}, \qquad F_{12} : \varphi^{(t)} \in \Sigma_{12}^{(t)}, \qquad F_{21} : \varphi^{(t)} \in \Sigma_{21}^{(t)}.$$

Since $B = (P_1 \text{ or } P_{12})$, we obtain the following subdivision of cases:

- A F_1 Here $(\varphi^{(t)}, \varphi^{(t+1)}] = \{\varphi^{(t+1)}\}$, with $\varphi^{(t)} \in \Sigma_1^{(t)}$ and $\varphi^{(t+1)} = \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$. We distinguish between (i) $\varphi^{(t+1)} \in \Sigma^{(t+1)}$ and (ii) $\varphi^{(t+1)} \in \Phi_F^{(t+1)} \setminus \Sigma^{(t+1)}$ with $\varphi^{(t+1)} \in \Sigma_{12}^{(t+1)}$. If (i), we have $(\varphi^{(t)}, \varphi^{(t+1)}] \cap \Sigma_{12}^{(t+1)} = \varnothing$. If (ii), we have necessarily $\varphi^{(t+1)} \notin \Phi_N^{(t+1)}$ otherwise the forward-signal would have recombined, hence $[\varphi^{(t)}, \varphi^{(t+1)}) \cap \Phi_N^{(t+1)} = \varnothing$. In both cases, we have $(3.20^\dagger)(t+1)$ and $(3.21^\dagger)(t+1)$. Finally, recall that $[\varphi^{(t)}, \varphi^{(t+1)}) = \{\varphi^{(t)}\}$ with $\varphi^{(t)} \in \Sigma_1^{(t)}$. Because of $(3.22)(t)$ and $(3.23)(t)$, we have $\varphi^{(t)} - 1 \notin \Sigma^{(t)} \cup \Phi_F^{(t)}$, hence $\varphi^{(t)} \notin \Phi_F^{(t+1)}$ and $(3.22^\dagger)(t+1)$.
- F_2 Here $(\varphi^{(t)}, \varphi^{(t+1)}] = \{\varphi^{(t+1)}\}$, with $\varphi^{(t)} \in \Sigma_2^{(t)}$ and $\varphi^{(t+1)} = \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$. This means that $(\varphi^{(t)}, \varphi^{(t+1)}] \cap$

$\Sigma_{12}^{(t+1)} = \varnothing$ and we directly have $(3.20^\dagger)(t+1)$ and $(3.21^\dagger)(t+1)$. Finally, recall that $[\varphi^{(t)}, \varphi^{(t+1)}) = \{\varphi^{(t)}\}$ with $\varphi^{(t)} \in \Sigma_2^{(t)}$, because of $(3.20)(t)$ the defect recombines or is displaced to the left at the next iteration, hence $\varphi^{(t)} \notin \Phi_F^{(t+1)}$ and we have $(3.22^\dagger)(t+1)$.

F_{12} Here $(\varphi^{(t)}, \varphi^{(t+1)}] = \{\varphi^{(t+1)}\}$ with $\varphi^{(t)} \in \Sigma_{12}^{(t)}$ and $\varphi^{(t+1)} = \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$. The same reasoning used in case F_1 gives $(3.20^\dagger)(t+1)$ and $(3.21^\dagger)(t+1)$. Finally, recall that $[\varphi^{(t)}, \varphi^{(t+1)}) = \{\varphi^{(t)}\}$ with $\varphi^{(t)} \in \Sigma_{12}^{(t)}$, since defects are displaced by at most 1 per iteration, we have $\varphi^{(t)} \notin \Sigma_{21}^{(t+1)}$ and $(3.22^\dagger)(t+1)$.

F_{21} Here $(\varphi^{(t)}, \varphi^{(t+1)}] = \{\varphi^{(t+1)}\}$ with $\varphi^{(t)} \in \Sigma_{21}^{(t)}$ and $\varphi^{(t+1)} = \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$. The same reasoning used in case F_2 gives $(3.20^\dagger)(t+1)$ and $(3.21^\dagger)(t+1)$. Finally, recall that $[\varphi^{(t)}, \varphi^{(t+1)}) = \{\varphi^{(t)}\}$ with $\varphi^{(t)} \in \Sigma_{21}^{(t)}$, because of $(3.22)(t)$ we have either (i) $\varphi^{(t)} - 1 \in \Sigma_2^{(t)}$ or (ii) $\varphi^{(t)} - 1 \notin \Sigma^{(t)} \cup \Phi_F^{(t)}$. If (i) the defect recombines, or is displaced to the left at the next iteration and the emitted forward-signal is erased during the iteration, which reduces to (ii) where we clearly have $\varphi^{(t)} \notin \Phi_F^{(t+1)}$, hence $(3.22^\dagger)(t+1)$.

$\neg A$ and B F_1 Here $[\varphi^{(t)}, \varphi^{(t+1)}) = (\varphi^{(t)}, \varphi^{(t+1)}] = \varnothing$ and we directly have $(3.20^\dagger)(t+1)$, $(3.21^\dagger)(t+1)$ and $(3.22^\dagger)(t+1)$.

F_{12} Here $[\varphi^{(t)}, \varphi^{(t+1)}) = (\varphi^{(t)}, \varphi^{(t+1)}] = \varnothing$ and we directly have $(3.20^\dagger)(t+1)$, $(3.21^\dagger)(t+1)$ and $(3.22^\dagger)(t+1)$.

$\neg A$ and $\neg B$ F_2 Here $(\varphi^{(t)}, \varphi^{(t+1)}) \subset \Sigma_{21}^{(t+1)}$ and $\varphi^{(t+1)} \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$ so that $(\varphi^{(t)}, \varphi^{(t+1)}] \subset \Sigma_{21}^{(t+1)} \cup \Sigma_1^{(t+1)}$, hence $(3.20^\dagger)(t+1)$ and $(3.21^\dagger)(t+1)$. In addition because of $(3.20)(t)$ and $(3.21)(t)$ the left defect recombines, or is displaced to the left and the emitted forward-signal is erased so that $[\varphi^{(t)}, \varphi^{(t+1)}) \cap \Phi_F^{(t+1)} = \varnothing$, hence $(3.22^\dagger)(t+1)$.

F_{21} The case reduces to F_2.

It is now left to prove $(3.23^\dagger)(t+1)$. Since we have already proven $(3.23^*)(t+1)$, it is left to consider the case $\sigma_a \in \Sigma_2^{(t+1)}, \sigma_b \in \Sigma_1^{(t+1)}$, with $\sigma_a < \sigma_b \leq \varphi^{(t+1)}$ and

$\varphi^{(t)} < \sigma_b$. Here because of Fact 5 we have $\varphi^{(t+1)} = \sigma_b$. We also have $\sigma_a \leq \varphi^{(t)}$ since otherwise applying Fact 5 with σ_a and some other defect $\sigma < \sigma_a$ (always existing because σ_a even) would imply $\sigma_a = \varphi^{(t+1)} = \sigma_b$ in contradiction with the initial assumption.

Fact 5 (One defect at a time)**.** *Let $\sigma^{(t)} \in \Sigma^{(t)}$ and $\sigma^{(t+1)} \in \Sigma^{(t+1)}$ be the position of a defect at times t and $t+1$, such that $\varphi^{(t)} < \sigma^{(t)}$. If $\sigma^{(t+1)} \leq \varphi^{(t+1)}$, then $\varphi^{(t+1)} = \sigma^{(t+1)}$.*

Let us suppose, for the sake of contradiction, that $\sigma_b - \sigma_a = \varphi^{(t+1)} - \varphi^{(t)} = 1$. In this case because defects can only move by at most 1 to the left per iteration, possible cases are restricted to

$$(i)\ \ \Sigma^{(t)} \cap [\varphi^{(t)}, \varphi^{(t)} + 2] = \{\varphi^{(t)}, \varphi^{(t)} + 1\}$$

$$(ii)\ \ \Sigma^{(t)} \cap [\varphi^{(t)}, \varphi^{(t)} + 2] = \{\varphi^{(t)}, \varphi^{(t)} + 2\},$$

$$(iii)\ \ \Sigma^{(t)} \cap [\varphi^{(t)}, \varphi^{(t)} + 2] = \{\varphi^{(t)} + 1, \varphi^{(t)} + 2\},$$

$$(iv)\ \ \Sigma^{(t)} \cap [\varphi^{(t)}, \varphi^{(t)} + 2] = [\varphi^{(t)}, \varphi^{(t)} + 2].$$

If (i) or (iv) the two left defects would have recombined unless $\varphi^{(t)} - 1 \in \Sigma^{(t)}$ which contradicts Fact 4. If (ii) the left defect would have been displaced to the left or recombined with a defect on the left because of $(3.20)(t)$. If (iii) the two left defects would have recombined. All cases lead to a contradiction, hence $\sigma_b - \sigma_a \geq 2$. We have finished the proof of $(3.20)(t+1)$, $(3.21)(t+1)$, $(3.22)(t+1)$ and $(3.23)(t+1)$. Lemma 5 follows by recursion. □

3.4.6 Increase of the interaction frontier

This section is devoted to the proof of Lemma 4 that states that the interaction frontier goes from one defect to the next one in linear time in the distance traveled. Since defects can only move to the left this directly implies that the frontier eventually reaches the last defect in linear time with the initial error width.

Proof of Lemma 4. Let $t_1 \geq 0$ be such that $\varphi^{(t_1)} \in \Sigma^{(t_1)}$ and $\Sigma^{(t_1)} \cap \mathbb{Z}^{(t_1)}_{>\varphi} \neq \varnothing$. If $\varphi^{(t_1)} \in \Sigma^{(t_1)}_2$, $\varphi^{(t)}$ is trivially strictly increasing until $t_2 > t_1$ such that $\Sigma^{(t_2)} \cap \mathbb{Z}^{(t_2)}_{>\varphi} = \varnothing$ or $\varphi^{(t_2)} \in \Sigma^{(t_2)}_1$, with $t_2 - t_1 \leq \sigma_{2m} - \varphi^{(t_1)}$ and $\varphi^{(t_2)} - \varphi^{(t_1)} \geq t_2 - t_1$.

In the following we consider the remaining case, $\varphi^{(t_1)} \in \Sigma_1^{(t_1)}$. Let us first prove that the following set is non-empty

$$T_2 = \{t \geq t_1 \mid |\Sigma^{(t)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}| \equiv 0 \pmod 2\} \quad = \{t \geq t_1 \mid \neg B\}. \tag{3.24}$$

The interaction frontier increases by 1 or remains constant at each iteration as long as we remain in the setting $|\Sigma^{(t)} \cap \mathbb{Z}_{\leq\varphi}^{(t)}| \equiv 1 \pmod 2$. Since $\varphi^{(t)} \in \Sigma^{(t)} \cup \Phi_F^{(t)}$, either the forward-signal propagates or the defect emits a forward-signal and $\varphi^{(t+1)} = \varphi^{(t)} + 1$, unless a recombination event occurred. However, the left defect is immobile because its left neighbouring site is without forward-signals or defects, see Lemma 5, this implies that no new negative charge is created in the open interval $(\sigma_1, \varphi^{(t)}]$ which upper bounds the number of recombination events. The total number of defects being even, if $\Sigma^{(t)} \neq \varnothing$ there will always be at least one defect on the right of $\varphi^{(t)}$ and this defect cannot move to the right. Hence T_2 is non-empty and we set $t_2 := \min(T_2)$. It is left to upper-bound $t_2 - t_1$. If $\Sigma^{(t_2)} = \varnothing$, because of defects parity necessarily the two defects $\varphi^{(t_2-1)}, \varphi^{(t_2-1)} + 1 \in \Sigma^{(t_2-1)}$ recombined and in this case $t_2 - t_1 = 1$ and $\varphi^{(t_2)} - \varphi^{(t_1)} \geq 1$. On the other hand, if $\Sigma^{(t_2)} \neq \varnothing$ and $|\Sigma^{(t_2)} \cap \mathbb{Z}_{\leq\varphi}^{(t_2)}| \equiv 0 \pmod 2$, because of Fact 5 we have $\varphi^{(t_2)} \in \Sigma^{(t_2)}$: we consider this case in the following.

We will now study how this defect located at $\varphi^{(t_2)}$ at time t_2 evolves with time. We denote by $\sigma^{(t)}$ its location at time t, so that $\sigma^{t_2} = \varphi^{(t_2)}$. Let us assume first for simplicity that $(\varphi^{(t_1)}, \sigma^{(t_1)}) \cap \Sigma^{(t_1)} = \varnothing$. In this situation no new negative charge is created within the open interval $(\varphi^{(t_1)}, \sigma^{(t)})$ for $t_1 \leq t \leq t_2$, and the total negative charge within the interval is initially upper bounded by $2(\sigma^{(t_1)} - \varphi^{(t_1)})$ at time t_1. Since the quantity also bounds the number of non-increasing steps, this gives the following upper bound on $t_2 - t_1$

$$t_2 - t_1 < (\sigma^{(t_1)} - \varphi^{(t_1)}) + 2(\sigma^{(t_1)} - \varphi^{(t_1)}) = 3(\sigma^{(t_1)} - \varphi^{(t_1)}). \tag{3.25}$$

Note that because $\sigma^{(t_1)} \leq \sigma_{2m}$ we also have $t_2 - t_1 \leq 3(\sigma_{2m} - \varphi^{(t_1)})$. The last displacement of the right defect σ before time t_2 is induced in the worst case by the last forward-signal emitted by the left defect at time $t_1 - 1$. The right defect displacement is upper bounded by $\lceil(\sigma^{(t_1)} - \varphi^{(t_1)})/2\rceil$ between t_1 and t_2, which gives the following lower bound on $\varphi^{(t_2)} - \varphi^{(t_1)}$

$$\varphi^{(t_2)} - \varphi^{(t_1)} \geq \lfloor \frac{\sigma^{(t_1)} - \varphi^{(t_1)}}{2} \rfloor. \tag{3.26}$$

Combining with with (3.25) and checking independently that $\varphi^{(t_2)} - \varphi^{(t_1)} \geq 1$, we obtain the desired inequality

$$\varphi^{(t_2)} - \varphi^{(t_1)} \geq \max(\lfloor (t_2 - t_1)/6 \rfloor, 1) \geq (t_2 - t_1)/11, \tag{3.27}$$

where we used that $t_2 - t_1$ is an integer.

Recall that we treated the simple case where the interval $(\varphi^{(t_1)}, \sigma^{(t_1)})$ was without defect. In the general case however, if $(\varphi^{(t_1)}, \sigma^{(t_1)}) \cap \Sigma^{(t_1)} \neq \varnothing$, necessarily those defects will recombine by pair before they would be reached by the interaction frontier. Let us consider the smallest $t_i > t_1$ such that $(\varphi^{(t_i)}, \sigma^{(t_i)}) \cap \Sigma^{(t_i)} = \varnothing$ with $\sigma^{(t_i)} \in \Sigma^{(t_i)}$ the defect σ at time t_i. Notice first that the previous reasoning can be applied similarly between t_1 and t_i. For iterations $t > t_i$, recombination events concern negative charges in the interval $[\varphi^{(t_i)}, \sigma^{(t_i)})$ at time t_i. Since the total charge of the interval is positive at that time, by direct combination of Facts 1 and 3, the previous reasoning is still valid and we have (3.27). □

3.4.7 Proof of useful facts

We now detail the proof of some Facts we used in the proof of Lemma 5.

Proof of Fact 4. Suppose, for the sake of contradiction, that using identical assumption we have

$$\{\varphi^{(t)} - 1, \varphi^{(t)}, \varphi^{(t)} + 1\} \subset \Sigma^{(t)} \tag{3.28}(t)$$

Since in the code-capacity model no new defects are created, and since defects can only be displaced by at most 1 to the left, the only possible configurations at time $t - 1$ can be grouped as follows

$$(i)\ \ \Sigma^{(t-1)} \cap [\varphi^{(t)} - 1, \varphi^{(t)} + 2] = \{\varphi^{(t)}, \varphi^{(t)} + 1, \varphi^{(t)} + 2\},$$

$$(ii)\ \ \Sigma^{(t-1)} \cap [\varphi^{(t)} - 1, \varphi^{(t)} + 2] = \{\varphi^{(t)} - 1, \varphi^{(t)} + 1, \varphi^{(t)} + 2\},$$

$$(iii)\ \ \Sigma^{(t-1)} \cap [\varphi^{(t)} - 1, \varphi^{(t)} + 2] = \{\varphi^{(t)} - 1, \varphi^{(t)}, \varphi^{(t)} + 1\},$$

$$(iv)\ \ \Sigma^{(t-1)} \cap [\varphi^{(t)} - 1, \varphi^{(t)} + 2] = \{\varphi^{(t)} - 1, \varphi^{(t)}, \varphi^{(t)} + 2\},$$

(v) $\Sigma^{(t-1)} \cap [\varphi^{(t)} - 1, \varphi^{(t)} + 2] = [\varphi^{(t)} - 1, \varphi^{(t)} + 2]$.

Because of the pre-processing correction step in the update rule, two defects recombine in cases (i) and (ii) which is not compatible with $(3.28)(t)$. The two left defects also recombine in cases (iii), (iv) and (v) unless $\varphi^{(t)} - 2 \in \Sigma^{(t-1)}$. In this case however, because of the update of the interaction frontier, we get $\{\varphi^{(t-1)} - 1, \varphi^{(t-1)}, \varphi^{(t-1)} + 1\} \subset \Sigma^{(t-1)}$, i.e. $(3.28)(t-1)$. It is clear that repeating the argument until $t = 0$ yields $\varphi^{(0)} \neq \min(\Sigma)$, in contradiction with the initialization of the interaction frontier. □

Proof of Fact 5. We distinguish between the three cases of update of $\varphi^{(t)}$ to $\varphi^{(t+1)}$. If A, then necessarily $\varphi^{(t+1)} = \varphi^{(t)} + 1 \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$ and because defects are displaced by at most 1 per iteration, we have either $\varphi^{(t+1)} = \sigma^{(t+1)}$ or $\varphi^{(t+1)} = \sigma^{(t+1)} + 1$. In the latter case, the defect was displaced to the left and necessarily its previous site $\sigma^{(t+1)} + 1$ is without forward-signal or defect, in contradiction $\sigma^{(t+1)} + 1 \notin \Sigma^{(t+1)} \cup \Phi_F^{(t+1)}$. Hence $\varphi^{(t+1)} = \sigma^{(t+1)}$ is the only possible case left. If $\neg A$ and B, then $\varphi^{(t+1)} = \varphi^{(t)}$ which is clearly incompatible with the assumptions. If $\neg A$ and $\neg B$ then because $\sigma^{(t+1)} \in \Sigma^{(t+1)} \cup \Phi_F^{(t+1)} \cap \mathbb{Z}_{>\varphi}^{(t)}$, necessarily $\varphi^{(t+1)} \leq \sigma^{(t+1)}$, hence $\varphi^{(t+1)} = \sigma^{(t+1)}$. This finishes the proof. □

3.5 Proof of threshold

The section is devoted to the proof of the code-capacity threshold theorem of the ASR decoder on a periodic lattice of size n. We consider the decoding to be successful if the initial error is corrected and if all variables of the decoder have returned to zero, conversely, a logical flip or a non zero automaton configuration is considered a logical error. We formally state the Theorem here.

Theorem 11 (Code-capacity threshold)**.** *Consider a family of 1D periodic lattices of size n. There exists $\varepsilon_{th} > 0, \alpha > 0$ and $\tau = \mathcal{O}(n)$ such that for $\varepsilon < \varepsilon_{th}$, the logical error rate $\bar{\varepsilon}$ of the ASR applied for τ time steps to an initial error where each qubit is flipped independently and identically with probability ε satisfies $\bar{\varepsilon} \leq \mathcal{O}(\exp\{[\frac{1}{2} n^{\alpha} \log \varepsilon / \varepsilon_{th}]\})$.*

The proof gives the following lower bounds $\varepsilon_{th} > 0.4\%$ and $\alpha > 0.12$, that are likely very conservative as indicated by numerical simulations in the stronger phenomenological model. The proof is inspired by previous work [93, 29, 74, 60] but is simplified to take advantage of the single dimension of the system. An

error configuration is decomposed into a hierarchy of connected components sufficiently far from each other such that a connected element of the lower level of the hierarchy is corrected independently from upper levels using Theorem 10. Logical errors then only arise in the presence of an element of the last level of the hierarchy. Counting the number of last-level representatives and bounding their weight gives Theorem 11.

Proof of Theorem 11. Let E be the random variable over edges of $\mathbb{Z}_n$ such that $\mathbb{P}(E) = \varepsilon^{|E|}(1-\varepsilon)^{n-|E|}$. We consider $u^{(t)}$ for $t \geq 0$ the sequence of ASR configurations with initial error configuration $\Sigma^{(0)} = \partial E$.

3.5.1 Hierarchical error decomposition

Define $L > 0$ to be optimized later. We define a level-0 chunk C_0 to be an element of E, that it to say a single edge of $\mathbb{Z}_n$ corresponding to an error. A level-k chunk $C_k = C_{k-1,1} \sqcup C_{k-1,2}$ is defined recursively to be the disjoint union of two level-$(k-1)$ chunks $C_{k-1,1}$ and $C_{k-1,2}$ such that $\mathrm{diam}(C_k) \leq L^k/2$. The level-$k$ error E_k is defined to be the union of all level-k chunks

$$E_k = \bigcup_i C_{k,i}. \tag{3.29}$$

By definition $E = E_0$, and we have the following sequence of inclusions for some $m > 0$,

$$E = E_0 \supseteq E_1 \supseteq ... \supseteq E_m \supsetneq E_{m+1} = \varnothing. \tag{3.30}$$

We can define $F_k = E_{k-1} \setminus E_k$ up to the last level of the hierarchy so that we obtain a disjoint decomposition of E

$$E = F_0 \sqcup ... \sqcup F_m. \tag{3.31}$$

Each subset F_k is furthermore decomposed into connected components that will be corrected independently. We say that a subset of errors $D_k \subset F_k$ is ℓ-connected if it cannot be split into two disjoint non-empty sets $B_{k,1}$ and $B_{k,2}$ separated by more than ℓ. That is to say, for any $B_{k,1}, B_{k,2} \neq \varnothing$, if $D_k = B_{k,1} \sqcup B_{k,2}$, then $d(B_{k,1}, B_{k,2}) \leq \ell$. An (ℓ, k)-connected component is a subset of F_k that is ℓ-connected and that is not strictly included in another ℓ-connected subset of F_k. From the structure of the decomposition of E into F_k we can

upper bound the size of connected components of F_k as well as lower bound their distance from each other.

Lemma 7 (Connected components [93, 29])**.** *Let $L \geq 6$ be some constant and a subset of errors $D_k \subseteq F_k$ be a (L^k, k)-connected component of F_k. Then,* $\mathrm{diam}(D_k) \leq L^k$ *and* $d(D_k, E_k \setminus D_k) > L^{k+1}/3$.

We include the proof of Lemma 7 at the end of the section for completeness. Intuitively, having the diameter of connected components to be bounded and different connected components to be far from each other will enable a local decoder such as the ASR to erase them independently. Combining Theorem 10 and Lemma 7 for the right choice of $\ell > 0$ then ensures that (ℓ, k)-connected components are corrected independently.

Lemma 8 (Hierarchical decoding)**.** *Let $L \geq 232$, and E be an error configuration that can be decomposed into $E = F_k \sqcup ... \sqcup F_{M-1}$ with $k \leq M-1$ for $M = \lfloor \log n / \log L \rfloor$. Consider D_k a (L^k, k)-connected component of F_k, D_k is corrected independently from $(F_k \setminus D_k) \sqcup F_{k+1} \sqcup ... \sqcup F_{M-1}$ in time $\tau_k \leq 77L^k$.*

Here, independent correction means that no excitation or defect originating from this (L^k, k)-connected component meets any excitation or defect from another (L^k, k)-connected component. This implies that the presence of this connected component within the initial error configuration does not affect the decoder outcome. Successive applications of Lemma 8 then directly show that a logical error is only possible in the presence of a level-M chunk in the decomposition of E. It is left to upper bound this probability.

Proof of Lemma 8. Consider an error E with following hierarchical decomposition $E = F_k \sqcup ... \sqcup F_{M-1}$ with $k \leq M-1$ for $M = \lfloor \log n / \log L \rfloor$ and D_k a (L^k, k)-connected component. Recall that we have $E = D_k \sqcup (E_k \setminus D_k)$.

We know that by Theorem 10 an isolated error $E' = D_k$ is corrected by $\tau_k < 77\mathrm{diam}(D_k)$ successive application of the ASR with excitations propagating up to distance at most $77\mathrm{diam}(D_k)$ to the right. Hence choosing L such that $77\mathrm{diam}(D_k) < d(D_k, E_k \setminus D_k)$ ensures that an excitation originating from another cluster on the left cannot reach D_k before it is erased, and that excitations from D_k cannot interfere with another cluster on the right. It suffices to replace the relevant diameters and distance by bounds from Lemma 7 to obtain a sufficient condition on L

$$77L^k < L^{k+1}/3, \tag{3.32}$$

and that choosing $L = 232$ ensures D_k is corrected independently from $E_k \setminus D_k$. □

3.5.2 Probability of a level-M chunk

Since for every $k \geq 1$ a level-k chunk is composed of two disjoint level-$(k-1)$ chunks, level-k chunks are of weight 2^k and their number of representatives up to translation, noted N_k, can be recursively upper bounded by

$$N_{k+1} \leq N_k^2 \times L^k \tag{3.33}$$

which initialized from $N_0 = 1$ gives for every $k \geq 1$

$$N_k \leq L^{2^k-(k+1)}. \tag{3.34}$$

Multiplying by n to account for translated configurations, the logical error probability can be upper bounded by a quantity doubly exponential in M

$$\overline{\varepsilon} \leq nL^{-(M+1)}(L\varepsilon)^{2^M}. \tag{3.35}$$

Using $M = \lfloor \log n / \log L \rfloor$ we retrieve Theorem 11 where $L = 232$, $\varepsilon_{th} = 1/L > 0.4\%$, $\alpha = \log 2 / \log L > 0.12$ and $\tau = 77n/L$. Note that by taking $M' = \log \log^c n$ for some $c > 1$, we obtain a threshold for a polylogarithmic number of decoder iterations, although this time the error suppression is only subexponential. □

Proof of Lemma 7. The proof by contradiction is taken from [93, 29] and included for completeness. Let D_k a be (L^k, k)-connected component of F_k. Suppose that either $\mathrm{diam}(D_k) > L^k$ or $d(D_k, E_k \setminus D_k) \leq L^{k+1}/3$. In the first case there exists $C_{0,1}$ and $C_{0,2} \in D_k$ two level-0 chunks such that $d(C_{0,1}, C_{0,2}) > L^k$. Necessarily $C_{0,1}$ and $C_{0,2}$ belong to two disjoint k-level chunks $C_{k,1}$ and $C_{k,2}$. In addition, since D_k is L^k-connected we can choose $C_{0,1}$ and $C_{0,2}$ such that $d(C_{0,1}, C_{0,2}) \leq 2L^k$. By the triangle inequality we have for $L \geq 6$

$$\begin{aligned} \operatorname{diam}(C_{k,1} \sqcup C_{k,2}) &\leq \operatorname{diam}(C_{k,1}) + d(C_{k,1}, C_{k,2}) + \operatorname{diam}(C_{k,2}) && (3.36)\\ &\leq L^k/2 + 2L^k + L^k/2 \leq L^{k+1}/2. && (3.37) \end{aligned}$$

This implies that $C_{k,1} \sqcup C_{k,2} \in E_{k+1}$ and subsequently $C_{0,1} \notin F_k$ which contradicts the initial assumption. In the latter case, i.e. if $d(D_k, E_k \setminus D_k) \leq L^{k+1}/3$, there exists $C_{0,1} \in D_k$ and $C_{0,2} \in E_k \setminus D_k$ such that $d(C_{0,1}, C_{0,2}) \leq L^{k+1}/3$. Let the two k-level chunks $C_{k,1}$ and $C_{k,2}$ be such that $C_{0,1} \in C_{k,1}$ and $C_{0,2} \in C_{k,2}$. Note that necessarily $C_{k,1} \cap C_{k,2} = \varnothing$ otherwise $C_{k,1} \cup C_{k,2}$ is L^k-connected and $C_{0,2} \in D_k$. By the triangle inequality we have for $L \geq 6$

$$\begin{aligned} \operatorname{diam}(C_{k,1} \sqcup C_{k,2}) &\leq \operatorname{diam}(C_{k,1}) + d(C_{k,1}, C_{k,2}) + \operatorname{diam}(C_{k,2}) && (3.38)\\ &\leq L^k/2 + L^{k+1}/3 + L^k/2 \leq L^{k+1}/2, && (3.39) \end{aligned}$$

which reduces to the contradiction of the former case. □

Chapter 4

Early fault-tolerant sparse IQP sampling in constant depth

This chapter covers the work that was published in [120].

Contents

4.1 Introduction

Sampling problems are tasks in which one generates a sample according to the probability distribution defined by a quantum circuit [106, 114, 73]. These

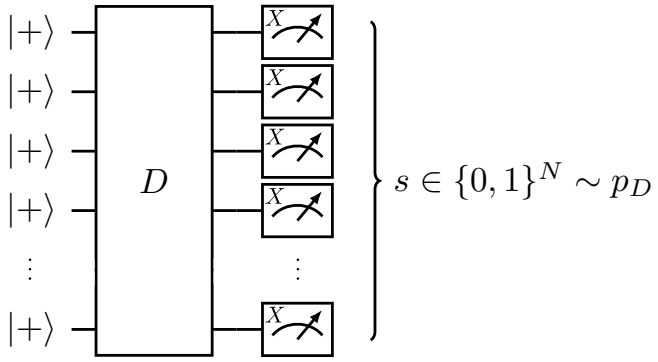

Figure 4.1: Reproduction of Figure 1.19 for convenience. IQP circuits on N qubits are defined by a unitary D diagonal in the computational basis with state preparation and measurements performed in the Hadamard basis. For sparse IQP circuits, D is a logarithmic-depth circuit consisting of T and CS-gates.

problems are considered a promising route to demonstrating quantum advantage, since they can become classically hard even for reasonably small circuits. In this chapter, we focus on the task of sampling from the output distribution of sparse IQP circuits, randomly generated using T and CS-gates as defined in the Introduction. Assuming two conjectures from complexity theory, Bremner et al. proved that there is no efficient classical algorithm for the sparse IQP problem [33]. This result is formally written as Theorem 6, that states that on average over the choice of D from the probability distribution over the family of sparse IQP circuits $\mathfrak{D}_N$, it is hard to sample classically from a distribution close to p_D.

While a fault-tolerant quantum computer can sample efficiently from such a distribution, we do not expect that this is the case for near-term quantum processors. In fact, the initial proposal [33] partially addressed this issue by considering a simple noise model where the quantum circuit is assumed to be ideal, except for some independent and identically distributed noise added to the classical value of the final outcomes. Unfortunately, this model is too naive and a more realistic noise model should assume that every gate suffers from some constant level of noise. In that case, because the number of gates is of order $N \log N$ in the circuit, it is immediate that noise will accumulate through the circuit and that the level of noise per qubit cannot be assumed to be constant, independent of N. Here we choose to consider a more general error model – the *local stochastic noise model* [66] – that includes well-known error models such as

the independent depolarizing noise channel but also allows for local correlated errors. In this model described in Section 4.5, errors are applied at each gate operation and the probability that faulty locations contain a specific set A is upper bounded by $p^{|A|}$. In this work we propose a physical implementation of sparse IQP circuits that is robust to this kind of noise, without requiring the full machinery of fault-tolerant quantum computation.

4.2 Main result

We show how to perform a fault-tolerant version of (sparse) IQP sampling with a constant-depth quantum circuit and with a space overhead that is only polylogarithmic in the width of the original circuit. Notably, all logical operations are performed from transversal gates in 3D. We note that [109] addressed a similar question for a different sampling problem, but the constant depth was obtained at the price of a polynomial overhead in terms of qubits because of differences in the initial computational problem and of the magic state distillation protocol necessary to its fault-tolerant implementation. In addition, it neglects some polynomial-time classical computation necessary for the error correction but during which errors can accumulate, while we bring down the complexity of error correction to polylogarithmic-time, making it less of an issue for future implementations. We note that similar computation times, for correcting a surface code of logarithmic size for instance, are often neglected in the literature.

In order to avoid multiple rounds of costly error correction, our main strategy is to make the encoded circuit of constant depth rather than logarithmic. This is challenging since the target logical circuit has logarithmic depth, and we want in addition to make it fault-tolerant. To this end, we design a family of quantum error-correcting codes on which sparse IQP circuits can be implemented in depth 1, meaning that they are fully parallelized. This is possible thanks to the commuting nature of sparse IQP gates [80]. In addition, we prove that the initial state can be encoded in constant quantum depth by performing stabilizer measurements. The only part of the process which is not implemented in constant depth is the final step of the state preparation: it consists of a single interaction with a classical computer that must compute a correction to apply, which depends on the stabilizer measurement results. In our scheme, this classical computation requires a polylogarithmic time because one needs to compute a correction for quantum patches of logarithmic size. We remark

problem	space overhead	depth	advantage	assumptions
factoring [63]	polylog	poly, adapt.	superpoly	factoring hard
graph state [109]	poly	$\mathcal{O}(1)$, adapt.	superpoly	PH $= \infty$ & ACH
sIQP [this work]	polylog	$\mathcal{O}(1)$, adapt.	superpoly	PH $= \infty$ & ACH
magic square [27]	polylog	$\mathcal{O}(1)$	quasi-log	unconditional

Table 4.1: Potential candidates for the demonstration of robust quantum advantage. The advantage is relative between the quantum depth and its minimal classical counterpart. Factoring displays a superpolynomial advantage provided that factoring is hard classically, but requires the full machinery of fault-tolerance. Graph state sampling and sparse IQP sampling also give a large advantage, under stronger assumptions (that the Polynomial Hierarchy does not collapse, and with an Average Case Hardness conjecture) and can be implemented with an adaptive circuit of constant depth. Finally the magic square problem leads to an unconditional advantage with a non-adaptative circuit of constant depth, but only offers a logarithmic advantage compared to classical computing.

that similar time complexities are often neglected in the literature of quantum fault-tolerance [66, 3, 86], and it may in fact not be a very problematic issue in practice.

Given a circuit $D \in \mathfrak{D}_N$ and precision $\delta > 0$, we construct the circuit $C_D(\delta)$ that samples from a distribution that is δ-close to p_D in total variation distance after classical post-processing. While the final classical post-processing is not performed in constant time, this is not an issue since all the qubits have already been measured. We discuss this point in Section 4.5. The circuit $C_D(\delta)$ is illustrated in Figure 4.2 and we detail its construction in Section 4.3. For circuits D of depth $\Theta(\log N)$, the space overhead is polylogarithmic in the precision δ and in the number of logical qubits N. Given that the average depth of sparse IQP circuits is $\Theta(\log N)$, a simple Markov inequality further implies that the fraction of circuits admiting a depth larger than $\alpha \log N$ decreases as $\mathcal{O}(1/\alpha)$. Thus an arbitrarily large fraction of such circuits benefits from the above overhead scaling. A practical difficulty that we do not address here is that our scheme requires long-range interactions. We state our main result:

Theorem 12 (Constant depth quantum advantage)**.** *There exists a universal $\varepsilon_{\text{th}} > 0$ such that, for all $N \in \mathbb{N}$, $D \in \mathfrak{D}_N$ and $\delta > 0$, running a noisy version of the quantum circuit $C_D(\delta)$ by inserting local stochastic noise of strength $\varepsilon < \varepsilon_{\text{th}}$ after each step, yields samples from p_D up to precision δ (in total variation*

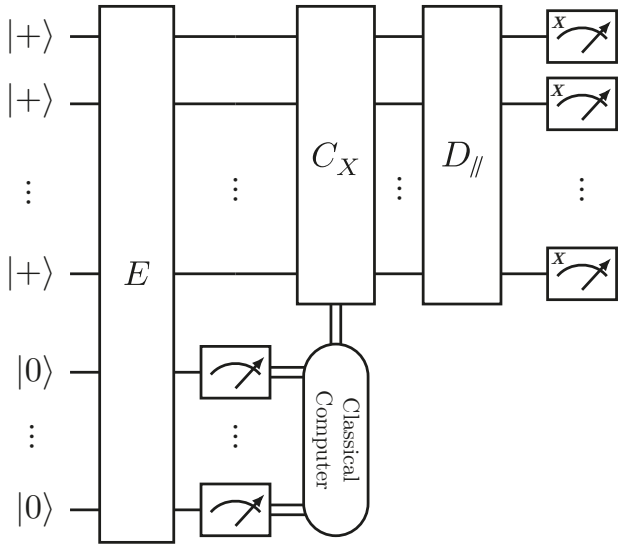

Figure 4.2: Our fault-tolerant implementation of a logical sparse IQP circuit D. The first layers E (stabilizer measurements) and C_X (adaptative error correction) prepare a logical state that is fed to a parallel version $D_{/\!/}$ of the sparse IQP circuit D, followed by single-qubit measurements. The overall circuit has constant (quantum) depth and acts on $N \times \text{polylog}(N)$ qubits. A single interaction with a classical computer is necessary to compute the correction C_X for the initial preparation. A final classical post-processing (not depicted) then computes a sample from the target distribution p_D.

distance) after classical post-processing.

Combining this with Theorem 6, our scheme demonstrates a super-polynomial quantum advantage for the task of sparse IQP sampling, assuming Conjecture 1 and that the Polynomial Hierarchy does not collapse.

To summarize our contribution, we reduce the fault-tolerance space overhead required to demonstrate a superpolynomial quantum advantage with a constant depth quantum circuit, from a large degree polynomial in [109] to a polylogarithmic overhead. A similar reduction is achieved for the classical computation complexity during the quantum computation. This comes at the cost of losing the local connectivity of the scheme.

4.2.1 Main concepts and ideas

4.2.1.1 The Tetrahelix code for fault-tolerant parallel computation

We recall that we aim to address two issues in order to get a final circuit of constant depth: we need to reduce the depth of the logical circuit for sparse

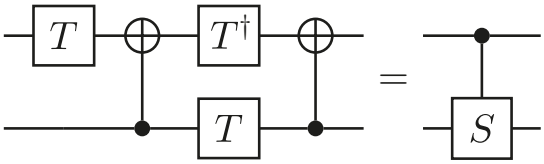

Figure 4.3: Implementation of the controlled-phase from controlled-not, T and $T^\dagger$ gates, all of those have transversal implementation on a tetrahedral color code. Switching T and $T^\dagger$ gives $CS^\dagger$

IQP from logarithmic to constant, and we need to find a fault-tolerant version that remains of constant depth. We achieve this by combining two ideas.

First we rely on *3D color codes* [25, 91] which admit transversal diagonal gates. More specifically, we will focus on the *tetrahedral code* subfamily that admits a transversal T-gate. Moreover, because these codes are CSS codes [38, 142], they also admit a transversal CNOT gate. Combining both, we see that tetrahedral codes also have transversal CS-gates, as shown on Figure 4.3. The second idea is that it is possible to fully parallelize an IQP circuit by using a GHZ encoding of each of the input qubits, in order to trade depth for width of the circuit. This means encoding a $|+\rangle$ as $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes k} + |1\rangle^{\otimes k})$ for some k corresponding to the number of gates supposed to be applied to the qubit, so that k is logarithmic in N. Then all k gates can be performed simultaneously by acting on a different qubit within the GHZ state. This is described in Figure 4.4. This new circuit has two shortcomings. First, despite being of constant depth, the logical phase-flip rate increases linearly with the size of the state since the measurement results of the k qubits within a GHZ state need to be aggregated. Second the preparation of bare GHZ states cannot be done fault-tolerantly in constant depth.

To solve these issues, we define a new stabilizer code – the *tetrahelix code* – combining the two ideas (3D color code for transversal gates and GHZ states for parallel implementation). We will detail the construction in Section 4.3, and only briefly explain its main properties here. The encoding is parameterized by two integers, k and L, accounting respectively for the parallelization capacity and the distance of the code. A k-tetrahelix code of distance L is defined by merging (in lattice surgery terms [79, 105, 96]) k tetrahedral codes of distance L along a 1-dimensional chain. Remarkably, the resulting $[\![\Theta(kL^3), 1, \Theta(L)]\!]$ tetrahelix code admits a depth-1 implementation of a logical sparse IQP circuit of depth k. This corresponds to a linear trade-off between the depth of the

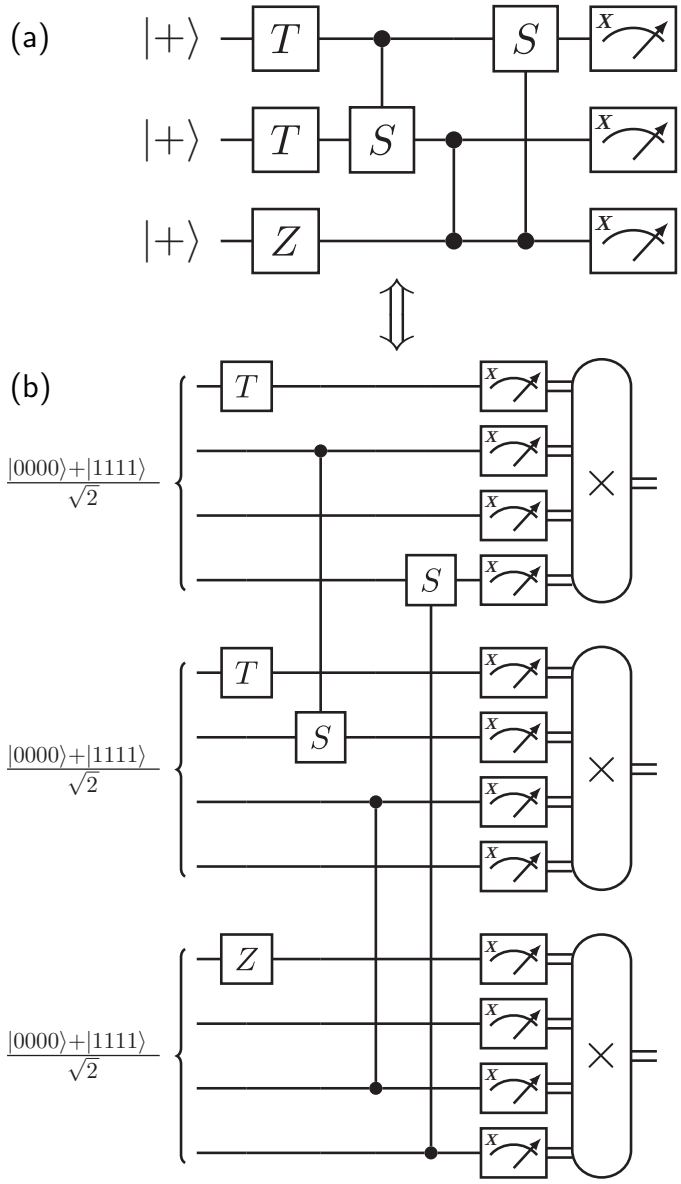

Figure 4.4: Each step $i \in \{1, ..., k\}$ of a circuit of depth k is simultaneously applied on the i^{th} physical qubit of all the GHZ states. The logical circuit (a) of depth 4 can be compiled in depth 1 up to classical decoding and state preparation by starting from GHZ state of size 4 and implementing circuit (b). The $\times$ blocks correspond to the classical decoding circuits.

initial logical circuit and the number of physical qubits:

Lemma 9. *Any sparse IQP circuit of depth k on N qubits can be implemented in depth 1 on N logical qubits encoded in k-tetrahelix codes.*

The constant depth of the circuit, together with the arbitrarily large distance L, ensures the fault-tolerance of the circuit up to a final classical post-processing to decode the results of the qubit measurements. We discuss in Section 4.5 how to achieve this by exploiting efficient decoders of color codes. The complexity of this step remains negligible compared to the super-polynomial quantum advantage of the overall circuit. The remaining challenge concerns the initial preparation of the encoded states of the tetrahelix code. One needs to ensure that such a preparation can also be done in constant depth and in a fault-tolerant manner.

4.2.1.2 Single-shot state preparation

A logical state of a quantum stabilizer code can always be prepared starting from a simple product state by measuring stabilizers and applying the appropriate correction to set the state in the code space. Such a scheme is however sensitive to measurement errors and fault-tolerance is usually achieved by repeating measurements. We circumvent this shortcoming by establishing the *single-shot* preparation of logical $|+\rangle$ states for the tetrahelix code.

In general, error correction based on erroneous measurements can induce large-weight physical errors whose accumulation could later translate into logical errors. In order to ensure fault-tolerance, one can prove that building on the particular structure of syndromes, the induced residual errors can be kept local with high probability. Such errors are then dealt with by the final classical decoding step. This property corresponds to single-shot decoding introduced by Bombín in [23]. Recall that throughout this thesis, $|\overline{x}\rangle$ denotes the logical encoded state $|x\rangle$ for $x \in \{0, 1, +, -\}$.

Lemma 10. *The tetrahelix code admits a single-shot preparation of $|\overline{+}\rangle$ / $|\overline{-}\rangle$ logical states, up to X stabilizers of the tetrahedral code.*

Note that, as argued in subsection 4.4.2, the X stabilizers need not be applied since they commute with sparse IQP encoded gates and hence can be propagated to the end of the circuit where they leave the final measurement unchanged.

The proof of the single-shot property of the k-tetrahelix code is detailed in Section 4.4 and relies on (i) the single-shot preparation of Hadamard basis states for 3D gauge color codes [23, 22], and (ii) the fact that the measurement errors occurring during code merging are detectable with the global stabilizer measurement outcomes.

We furthermore argue that the associated decoding can be performed on a classical computer in polylogarithmic-time with respect to N in Section 4.4. We consider it to be instantaneous to derive Theorem 12.

4.2.2 Sketch of proof of Theorem 12

The rest of the chapter is devoted to establishing Theorem 12. In Section 4.3, we first briefly review tetrahedral codes, from the 3D color code family. Next, we define the tetrahelix code family obtained by merging tetrahedral codes. We prove that the k-tetrahelix code reduces the depth k of a sparse IQP circuit to depth 1 (Lemma 9). In Section 4.4, we prove that the merging failure probability between two tetrahedral codes of distance L is exponentially suppressed in L and hence that k-tetrahelix encoded states in the Hadamard basis can be faithfully prepared in constant quantum depth (Lemma 10). In Section 4.5, we prove the fault-tolerance of the scheme. More precisely, we prove the existence of a non-zero error threshold independent of k, below which we arbitrarily suppress logical errors by increasing L for any encoded sparse IQP circuit.

4.3 Tetrahelix code

4.3.1 Overview of tetrahedral codes

Color codes are a family of topological codes introduced by Bombín and Martin-Delgado [21, 25, 91]. Their main feature is that they admit a transversal implementation of single-qubit phase gates, including the T-gate when the codes are 3-dimensional. In the following we focus on the subfamily of *tetrahedral codes* that encode a single qubit. Tetrahedral codes are defined on 3-dimensional color complexes, that we will call 3-colexes as in [24], of a tetrahedral shape as described in Figure 4.5 with the vertices corresponding to the data qubits. 3-Colexes are 3D lattices with the properties that (i) each cell is assigned one of four colors such that no two adjacent cells are of the same color; (ii) three

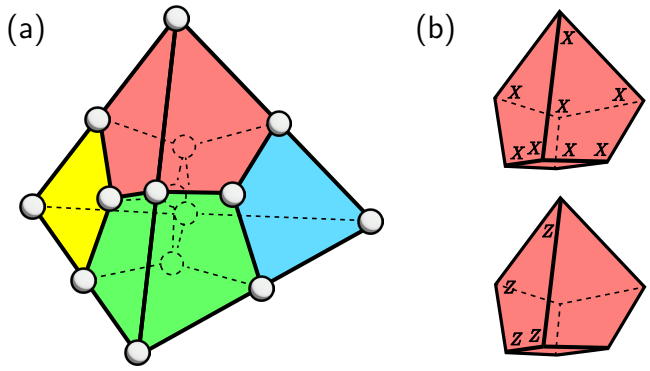

Figure 4.5: (a) The [15,1,3] Reed-Muller code is the smallest example of tetrahedral codes. Here qubits are on vertices and $\overline{X}$ and $\overline{Z}$ logical operators can be chosen respectively on a face and an edge of the tetrahedron. (b) X stabilizers are supported on cells (elements of $\mathcal{C}$) and Z stabilizers on faces (elements of $\mathcal{F}$).

colors appear on each external facet of the complex, and such a facet is associated with the missing color (in Figure 4.5 these facets correspond to the four triangular external boundaries of the tetrahedron); (*iii*) each vertex is incident to a cell or facet of all possible colors.

In the following we denote by $L \in \mathbb{N}$ the number of vertices on the edges of the lattice, and will correspond to the code distance, as explained below. The construction of a tetrahedral 3-colex is not unique for a given L but if one relies on tessellations of uniform density, then the resulting codes each encode a single logical qubit in $m = \Theta(L^3)$ physical qubits, and all display the properties that we will require.

Let us recall the formal definition of a tetrahedral code on m qubits, which will serve as a building block for the *tetrahelix code*. We start from a tetrahedral 3-colex with set of vertices $\mathcal{V}$, faces $\mathcal{F}$ and cells $\mathcal{C}$. Physical qubits are associates with vertices, so $m = |\mathcal{V}|$. The tetrahedral code associated to this colex is a CSS code with stabilizers given by:

$$S_X^1 = \langle X(c),\ c \in \mathcal{C} \rangle, \tag{4.1}$$

$$S_Z^1 = \langle Z(f),\ f \in \mathcal{F} \rangle. \tag{4.2}$$

Here, each cell c or face f is identified as a binary vector of length m with ones at the locations corresponding to the associated vertices, and we define $X(a) := \otimes_{i=1}^m X_i^{a_i}$ for $a \in \{0,1\}^m$. (and similarly for $Z(a)$). In words, the X

stabilizer associated to a 3-cell c is the product of Pauli X operators on all the vertices in the boundary of c. In particular, X stabilizers are associated by the 3-cells of the colex and Z stabilizers are associated with the faces.

The fundamental property of tetrahedral color codes is that for each code of the family there exists a partition of vertices $\mathcal{V} = \mathcal{V}^+ \cup \mathcal{V}^-$ such that applying the gate T on $\mathcal{V}^+$ and $T^\dagger$ on $\mathcal{V}^-$ implements an encoded logical T-gate [22]:

$$T(\mathcal{V}^+)T^\dagger(\mathcal{V}^-)\,|\overline{x}\rangle = \overline{T}\,|\overline{x}\rangle\,. \tag{4.3}$$

Similarly to encoded states $|x\rangle$ denoted by $|\overline{x}\rangle$, we denote by $\overline{U}$ the encoded logical unitary U. Together with the existence of transversal controlled-not gates, this implies the transversal implementation of the CS-gate (see Figure 4.3):

$$CS(\mathcal{V}^+)CS^\dagger(\mathcal{V}^-)\,|\overline{x}\rangle\,|\overline{y}\rangle = \overline{CS}\,|\overline{x}\rangle\,|\overline{y}\rangle\,, \tag{4.4}$$

where $CS(\mathcal{V}^+)$ denotes the transversal application of CS between the analogous sets $\mathcal{V}^+$ of two code blocks. $\overline{X}$ and $\overline{Z}$ logical operators are respectively surface-like and string-like and the X distance and Z distance scale as $\Theta(L^2)$ and $\Theta(L)$, respectively.

4.3.2 Construction of the tetrahelix code

The transversality of the sparse IQP gate set paves the way towards the fault-tolerant implementation of such circuits. This would however require repeated error correction cycles at each circuit step, that is a logarithmic number of times. Concatenating a tetrahedral code with a repetition code gives a family of codes that present the desired parallelization property. Unfortunately, it does not meet the criteria of constant depth preparation for the initial encoded states. We now define a new code, the *tetrahelix code*, that displays both properties: (i) depth-1 implementation of a sparse IQP circuit, (ii) constant-depth encoded state preparation in the Hadamard basis.

As briefly mentioned in Section 4.2.1.1, a tetrahelix code is obtained by merging tetrahedral codes in lattice surgery terms [79, 105, 152]. Here we detail the construction starting by merging two such codes of distance L, with respective sets of vertices $\mathcal{V}_1$ and $\mathcal{V}_2$ as described in Figure 4.6(a). We consider two codes which are exact mirror images of one another and we denote by $\varphi : \mathcal{V}_1 \to \mathcal{V}_2$ the bijection between the two sets of vertices. An external triangular

facet, $B_1 \subset \mathcal{V}_1$, together with its mirror image, $B_2 = \varphi(B_1)$, are chosen and every vertex is paired with the corresponding one on the other code. This pairing defines a set of pairs $\mathcal{P}_{1,2} := \{(v, \varphi(v)) | \forall v \in B_1\}$.

The merge operation consists in fusing the X stabilizers on the boundaries and adding new Z stabilizers of weight 2 associated to the paired qubits in $\mathcal{P}_{1,2}$. More precisely, the Z stabilizers are defined as $\langle Z(f), f \in \mathcal{F}^2 \rangle$ with

$$\mathcal{F}^2 := \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{P}_{1,2}. \tag{4.5}$$

Similarly, we define $\mathcal{C}^2$ the union of merged stabilizers and unmerged ones:

$$\mathcal{C}^2 := \mathcal{C}_1^* \cup \mathcal{C}_2^* \cup \mathcal{M}_{1,2} \tag{4.6}$$

with

$$\mathcal{C}_i^* := \{C \in \mathcal{C}_i | C \cap B_i = \varnothing\}, \tag{4.7}$$

and

$$\begin{aligned} \mathcal{M}_{1,2} := \{C_1 \cup C_2 | C_1 \in \mathcal{C}_1, C_2 \in \mathcal{C}_2, \\ \varphi(C_1 \cap B_1) = C_2 \cap B_2 \neq \varnothing\}. \end{aligned} \tag{4.8}$$

X stabilizers are then defined as $\langle X(c), c \in \mathcal{C}^2 \rangle$ corresponding to non-adjacent cells of each code and fused adjacent ones (paired according to their color as in Figure 4.6(a)). This construction ensures that X stabilizers commute with every Z stabilizer including the newly defined Z stabilizers with support on $\mathcal{P}_{1,2}$. We denote by $\overline{X}_i$ and $\overline{Z}_i$ the logical operators of the initial tetrahedral codes from which we define the logical operators of the new code. The 2-tetrahelix code encodes a single logical qubit for which the logical operators can be taken of the form $\overline{Z} = \overline{Z}_1$ (or $\overline{Z}_2$) and $\overline{X} = \overline{X}_1 \overline{X}_2$ with $\overline{X}_2$ chosen so that its support intersects on the same subset of $\mathcal{P}_{1,2}$ than $\overline{X}_1$ (to commute with the associated Z stabilizers).

Merging additional tetrahedra does not fundamentally change the analysis. Tetrahedra can be aligned in the shape of Figure 4.6(c) to form a chain of length $k \in \mathbb{N}$ so that each extremal vertex is shared between at most four tetrahedra. This ensures that at most four X stabilizers are fused together. This linear packing of regular tetrahedra is known as a Boerdijk-Coxeter helix or tetrahelix [20, 45] which motivates the name of the code.

Denoting by $\mathcal{V}_i$ the set of vertices of the i^{th} tetrahedral color code, we get a partition of the set of all vertices $\mathcal{V} = \cup_{i=1}^k \mathcal{V}_i$. With $\mathcal{P}_{i,i+1}$ denoting new

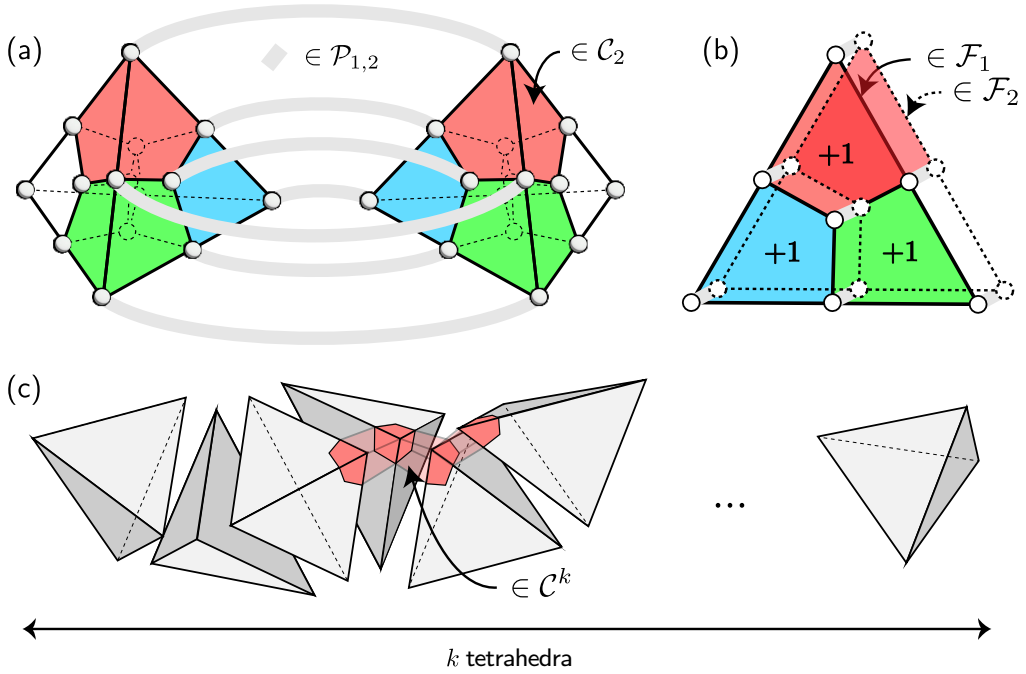

Figure 4.6: (a) Adjacent tetrahedra (here for $L = 3$) are merged by measuring pairs of qubits from $\mathcal{P}_{1,2}$ that become Z stabilizers of the new code. The colors of the second tetrahedron are chosen by convention so that merged X stabilizers are of the same color. (b) Every new Z stabilizer generator must be set to 1 to project the state in the code space. The new pair stabilizers value are not independent since they are all related to Z stabilizers of the two tetrahedral codes on the face on which the merge is performed. In particular the product of four pairs overlapping the same Z stabilizer (of support in $\mathcal{F}_1$) is equal to 1. (c) Merging additional tetrahedra with each other enables to form a chain of tetrahedra of length k. The optimal chain has the shape of an helix and corresponds to minimizing the number of merged X stabilizers (of support in $\mathcal{C}^k$) that is equal to four in this packing.

Z stabilizers between adjacent tetrahedra i and $i+1$, $\mathcal{F}^k$ and $\mathcal{C}^k$ are defined analogously as $\mathcal{F}^2$, and $\mathcal{C}^2$ to ensure stabilizer commutation:

$$\mathcal{F}^k := \bigcup_{i=1}^{k-1} (\mathcal{F}_i \cup \mathcal{P}_{i,i+1}) \cup \mathcal{F}_k, \tag{4.9}$$

$$\mathcal{C}^k := \bigcup_{i=1}^{k-1} (\mathcal{C}_i^* \cup \mathcal{M}_{i,i+1}) \cup \mathcal{C}_k^*, \tag{4.10}$$

where here $\mathcal{C}_i^*$ and $\mathcal{M}_{i,i+1}$ are defined recursively so that stabilizers can be merged across several tetrahedra (up to three on edges and up to four on summits). We define the k-tetrahelix code that encodes a single logical qubit in $\Theta(kL^3)$ physical qubits from its set of stabilizers:

$$S_X^k := \langle X(c),\ c \in \mathcal{C}^k \rangle, \tag{4.11}$$

$$S_Z^k := \langle X(f),\ f \in \mathcal{F}^k \rangle. \tag{4.12}$$

The logical $\overline{Z}$ operator can be chosen as any of the logical $\overline{Z}_i$ operators of the composing tetrahedral codes, while the $\overline{X}$ logical operator is a product of $\overline{X}_i$ operators recursively chosen so that $\overline{X}_i$ and $\overline{X}_{i+1}$ intersect with the same subset of $\mathcal{P}_{i,i+1}$.

4.3.3 Code distance

The X and Z distances of a code correspond to the minimal weights of X and Z logical operators. We denote by d_X^k and d_Z^k the X and Z distances of the k-tetrahelix code and prove that:

$$d_Z^k = \Theta(L) \quad \text{and} \quad d_X^k = \Theta(kL^2). \tag{4.13}$$

We prove the result for the 2-tetrahelix code by relating logical operators of the tetrahelix code to those of the initial tetrahedral codes, the result generalizes to arbitrary k-tetrahelix code by recursion. In this subsection we denote by $d_X^1 = \Theta(L^2)$ and $d_Z^1 = \Theta(L)$ the X and Z distances of a tetrahedral code of edge length L and number of qubits $m = \Theta(L^3)$.

Let us consider a logical operator $\overline{Z}$ of the 2-tetrahelix code. We index the vertices of the two composing tetrahedra in a symmetric manner with respect

to the paired facets. The logical operator $\overline{Z}$ is of the form of the tensor product of Pauli Z operators on each tetrahedron $\overline{Z} = Z(\mu) \otimes Z(\nu)$, with $\mu, \nu \in \{0,1\}^m$. We will show that up to multiplication by Z stabilizers we can transfer $Z(\mu) \otimes Z(\nu)$ to $Z(\mu+\nu) \otimes I$. This means that we transfer the physical Pauli Z operators from the second tetrahedron to the symmetric ones in the first one. We can then conclude by noticing that $Z(\mu+\nu)$ is a logical operator of the first tetrahedron and hence of weight larger or equal to d_Z^1. We thus have

$$|\overline{Z}| = |Z(\mu) \otimes Z(\nu)| \geq |\overline{Z}(\mu+\nu) \otimes I| \geq d_Z^1, \tag{4.14}$$

which concludes the argument.

Indeed, since an arbitrary logical $\overline{Z}_1$ operator of the first tetrahedron is also a logical operator of the tetrahelix code, there exists a Z stabilizer R_Z such that $\overline{Z} = \overline{Z}_1 \times R_Z$. Such a stabilizer is necessarily of the form:

$$R_Z = R_Z^1 \times R_Z^2 \times R_Z^{1,2}, \tag{4.15}$$

where R_Z^1 is a Z stabilizer of tetrahedron 1, R_Z^2 of tetrahedron 2, and $R_Z^{1,2}$ is a product of Z-stabilizers defined at the boundary (paired qubits in $\mathcal{P}_{1,2}$). It is clear that, multiplying $\overline{Z}$ by R_Z^2, maps the support from the bulk of the second tetrahedron to the paired facet of this tetrahedron. Next, we completely transfer this support to the facet of first tetrahedron by multiplying by $R_Z^{1,2}$. At this point, the support of the logical operator is entirely contained in the first tetrahedron. Now we apply the symmetric version of R_Z^2 defined on the first tetrahedron. This maps the original logical operator to $Z(\mu+\nu) \otimes I$.

The case of the X distance is straightforward as the product of $\overline{X}_1$ and $\overline{X}_2$ logical operators whose supports intersect with the same pairs of $\mathcal{P}_{1,2}$ yields a logical operator of the 2-tetrahelix code, and this product form is stable upon multiplication by X stabilizers. This stability is a direct consequence of the fact that the restriction of a tetrahelix X stabilizer to a single tetrahedron is a stabilizer of the tetrahedral code. Merging an additional tetrahedral code hence increases the X distance by d_X^1:

$$|\overline{X}| = |\overline{X}_1 \overline{X}_2| \geq 2d_X^1. \tag{4.16}$$

The same discussion between a $(k-1)$-tetrahelix code and a tetrahedral code generalises the proof by recursion to k-tetrahelix code for arbitrary k. Recalling

that $d_Z^1 = \Theta(L)$ and $d_X^1 = \Theta(L^2)$, we obtain the bounds of (4.13).

4.3.4 Parallel computation

We turn to the properties of the code concerning parallel computation. We establish Lemma 9 by showing that the encoded T-gate can be implemented in depth 1 on a single tetrahedron of the chain. A tetrahedral code on m physical qubits is a CSS code and logical states can therefore be written in the form

$$|\overline{x}_1\rangle = \frac{1}{\sqrt{|\mathcal{S}^1|}} \sum_{s_1 \in \mathcal{S}^1} |s_1 + x_1 L_1\rangle \,, \tag{4.17}$$

with the addition taken modulo 2 and $x_1 \in \{0,1\}$ and $\mathcal{S}^1 \subset \{0,1\}^m$ such that for $s_1 \in \mathcal{S}^1$ we have $X(s_1) \in S_X^1$. Similarly, $L_1 \in \{0,1\}^m$ represents an arbitrary $\overline{X}_1$ logical operator. The transversal implementation of the T-gate $T(\mathcal{V}^+)T^\dagger(\mathcal{V}^-) = \overline{T}$ on the tetrahedral code implies that each codeword gains the same phase from the application of $T(\mathcal{V}^+)T^\dagger(\mathcal{V}^-)$:

$$T(\mathcal{V}^+)T^\dagger(\mathcal{V}^-)\,|s_1 + x_1 L_1\rangle = e^{\frac{i\pi}{4}|x_1|}\,|s_1 + x_1 L_1\rangle \,. \tag{4.18}$$

The logical computational states of the k-tetrahelix code are given by

$$|\overline{x}\rangle = \frac{1}{\sqrt{|\mathcal{S}^k|}} \sum_{s \in \mathcal{S}^k} |s + xL\rangle \,, \tag{4.19}$$

for $x \in \{0,1\}$. Here $\mathcal{S}^k \subset \{0,1\}^{k\times m}$ is such that for $s \in \mathcal{S}^k$, we have $X(s) \in S_X^k$ and L is the vector associated to an arbitrary logical $\overline{X}$ operator. For each X stabilizer, $s \in \mathcal{S}^k$ is a concatenation of k vectors $s_i \in \mathcal{S}^1, i \in \{1,\ldots,k\}$: $s = [s_1, ..., s_k]$. Similarly, for the logical operator, the binary vector L is a concatenation of vectors L_i each representing a logical $\overline{X}_i$ operator of the i^{th} tetrahedral code. Focusing on tetrahedron i_0, and up to qubit re-ordering, we can thus write

$$|\overline{x}\rangle = \frac{1}{\sqrt{|\mathcal{S}^k|}} \sum_{s_{i_0} \in \mathcal{S}^1} |s_{i_0} + xL_{i_0}\rangle \otimes |\psi(s_{i_0}, x)\rangle \,. \tag{4.20}$$

The terms $|\psi(s_{i_0}, x)\rangle$ depend on s_{i_0} because of correlations between codewords restricted to different tetrahedra induced by overlapping X stabilizers, but this

does not impact our argument. Taking $\mathcal{V}_{i_0}^+$ and $\mathcal{V}_{i_0}^-$ as in (4.3), we have

$$T(\mathcal{V}_{i_0}^+)T^\dagger(\mathcal{V}_{i_0}^-)\,|s_{i_0}+xL_{i_0}\rangle = e^{\frac{i\pi}{4}|x|}\,|s_{i_0}+xL_{i_0}\rangle\,. \tag{4.21}$$

Combining (4.20) and (4.21) directly implies:

$$T(\mathcal{V}_{i_0}^+)T^\dagger(\mathcal{V}_{i_0}^-)\,|\overline{x}\rangle = \overline{T}\,|\overline{x}\rangle\,. \tag{4.22}$$

To extend the arguments to the CS-gate, we would need to use the gadget of Figure 4.3. Notice that while the T and $T^\dagger$-gates are applied on a single tetrahedron, the CNOT gates would need to be applied on all physical qubits (CSS property). However, as these CNOT gates come in pairs, they cancel each other outside the tetrahedron where the T-gates are applied. Thus, the CS-gate can also be applied between two arbitrary tetrahedral blocks of two tetrahelix codes:

$$CS(\mathcal{V}_{i_0}^+)CS^\dagger(\mathcal{V}_{i_0}^-)\,|\overline{x}\rangle\,|\overline{y}\rangle = \overline{CS}\,|\overline{x}\rangle\,|\overline{y}\rangle\,. \tag{4.23}$$

This implies that the k-tetrahelix code can implement in a single step an encoded T or a CS-gate on each tetrahedral block of the code. Therefore, we obtain a depth-1 parallel implementation of a depth-k sparse IQP circuit up to state preparation. This finishes the proof of Lemma 9. In the next section we prove that encoded states in the Hadamard basis can be prepared fault-tolerantly in constant quantum depth.

4.4 Constant-depth preparation of encoded states

The encoded states of the k-tetrahelix code in the Hadamard basis can be prepared by merging k associated encoded states of the composing tetrahedral blocks. In this section, we show that both the steps of preparing encoded tetrahedral states and their merging can be done in constant quantum depth.

4.4.1 Single-shot decoding of tetrahedral code

Since the state $|+\rangle^{\otimes m}$ is stabilized by all X stabilizers and by the $\overline{X}$ logical operator of a CSS quantum code, the projection over the logical $|\overline{+}\rangle$ can ideally be done by measuring the Z stabilizers and a single step of Pauli X corrections. Measurement errors however usually prevent such reliable encoded state

preparation in constant depth. Indeed, measurement errors induce residual data errors after Pauli corrections, which usually calls for many repeated measurements before such a correction is applied. Repeating measurements gives an extra dimension to the error syndrome where ancilla and data errors can be separated by the decoding algorithm which infers an error pattern close to the most likely one and hence an appropriate correction.

An alternative approach is to build on the structure of the error syndrome of some codes to ensure that a single round of local measurements is sufficient to ensure the locality of the residual errors with high probability. This strategy was first proposed by Bombín in [23] for 3D gauge color codes and is known as *single-shot decoding* which is a property of a quantum error-correcting code in conjunction with its decoder. In the case of 3D color codes, Z stabilizers on faces correspond to Z gauge operators of 3D gauge color codes. This ensures the single-shot decoding property up to a classical computation of polynomial complexity in the code size. Furthermore, the topological nature of the code implies that the measurements can be parallelized to a constant quantum depth. In conclusion, we have a constant depth preparation of encoded states in the Hadamard basis for the tetrahedral code up to local residual errors.

4.4.2 Single-shot merging of tetrahedral states

Merging two tetrahedral codes of distance L into a 2-tetrahelix code is described in Figure 4.6(a-b). Pairs of qubits from $\mathcal{P}_{1,2}$ are measured over the faces on which tetrahedral codes are merged and a correction is applied depending on the measurement outcomes. Since facets of a tetrahedral code have the structure of a triangular code (2D color code) of size L, in the absence of errors, measurements yield a binary codeword w of the corresponding classical code. This codeword can be written as the sum of an X stabilizer and an X logical operator of the 2D code with the same formalism used in Section 4.3.4 for the 3D code.

The appropriate correction then can be seen to be a 3D color code codeword whose restriction to the triangular code vertices gives w. This can be obtained by determining first the decomposition of w into facets of the triangular code (X stabilizer generators) and the logical operator X over the entire triangle (so that it is a logical operator of both the 2D and the 3D codes) before mapping the facets to cells to get a 3D code codeword

$$\begin{aligned} w &= s_{2D} + xL_{2D} \\ &\rightarrow s_{3D} + xL_{3D} = \mathrm{corr}(w) \end{aligned} \tag{4.24}$$

for $x \in \{0, 1\}$. Importantly, an X stabilizer of the tetrahedral code commutes with encoded T and CS-gates on the tetrahelix code since it does not change the structure of the codewords described in subsection 4.3.4. Since the circuit ends with X measurements this means that it is sufficient for our purpose to compute x and only apply the logical part of the correction. In other words, we only need to prepare tetrahelix encoded states up to tetrahedral codes X stabilizers.

If the two tetrahedral codes are not perfectly in their code space, or in the case of measurement errors, the measurement results deviate from w:

$$w_e = w + e_r + e_m. \tag{4.25}$$

Here, e_r accounts for the residual errors of the tetrahedral states preparation, and e_m stands for measurement errors. Because the preparation of encoded states in the Hadamard basis for the tetrahedral code is single-shot, the resulting errors follow a local stochastic noise model. This is also the case for measurement errors and hence decoding the triangular code yields the correct value of $\overline{Z}_1\overline{Z}_2$ with probability exponentially close to 1 in L. $\overline{Z}_1\overline{Z}_2$ is then set to $+1$ by applying or not $\overline{X}_1$.

A k-tetrahelix code encoded state can then be prepared in a similar manner simply by repeating the merging operation with additional tetrahedral codes, while always applying the logical correction on the tetrahedron for example on the left of the merge. This scheme can be seen as similar to preparing a GHZ state of size k from parity measurements and logical correction, with the difference that here, measurement errors are exponentially suppressed, thus giving Lemma 10.

Efficient decoding algorithms exist for 2D color codes [39, 134] and 3D color codes [23, 92] and are single-shot for the 3D case. These algorithms have a complexity polynomial in the code size. The different tetrahedral encoded states can be prepared in parallel. Parallel merge measurements followed by iterative computation of the associated correction then give a preparation of tetrahelix encoded states with polynomial in L and proportional to k classical

computation. We prove in Section 4.5 that we need a polylogarithmic number of qubits per code block which hence gives a polylogarithmic-time classical computation.

4.5 Application to sparse IQP circuits

In this section, we apply the results of the two previous sections to demonstrate the main result of this chapter stated in Theorem 12. We start by presenting the error model. Next, we show that the encoding of the circuit of Figure 4.2 is fault-tolerant by proving the existence of an error threshold. Finally, we provide an estimation of the space overhead of the scheme.

4.5.1 Error model

The coupling of the quantum system with the environment generates noise that can later induce errors in the computation. We use the *local stochastic quantum noise model* from [66] where the set of faulty locations is a random variable of a discrete space-time and local correlations are allowed. No assumption is made on a particular type of error operator. This makes the model general enough to cover a wide class of applications. In particular this captures commonly studied noise channels such as independent depolarizing and dephasing noise, or amplitude damping.

A noise model of parameter ε that satisfies the following two properties is said to be locally stochastic: (i) the faults are confined to a random set of space-time locations $A \subset \mathcal{V}$ with probability $p(A)$ and (ii) the probability that a set of faulty locations contains a specific set of A locations is upper bounded by $\varepsilon^{|A|}$.

$$\sum_{A' \supseteq A} p(A') \leq \varepsilon^{|A|}. \tag{4.26}$$

Final measurements are performed in the Hadamard basis and hence at the end of the circuit only Z-type errors induce errors on the classical output. Z errors can either be environmentally induced or generated during the propagation of X-type errors in the circuit. X errors can also arise due to the coupling with the environment but also from incorrect preparation of encoded states (recall that the preparation only includes X correction). Local stochastic errors propagate as such through the constant depth circuit but residual errors after

encoded states preparation are not necessarily local. We showed in Section 4.4 that their non-local representatives admit exponentially low probabilities which implies that the correction of local stochastic errors by the final decoding is sufficient to exponentially suppress the logical error rate.

More formally, residual errors after preparation of tetrahedral encoded states are characterised in [23] such that (i) correctable physical errors follow a local stochastic noise model $\mathcal{N}^{\varepsilon}_{\mathrm{T,loc}}$, (ii) non-correctable physical errors (that is to say errors whose correction attempt induces a logical error) are exponentially suppressed, we denote by $\mathcal{N}^{\tilde{\varepsilon}_1}_{\mathrm{T,nc}}$ the corresponding error channel. Using a similar notation we call $\mathcal{N}^{\tilde{\varepsilon}_2}_{\mathrm{M,nc}}$ the channel associated to logical errors due to unsuccessful merging, with probability exponentially suppressed in the code distance.

The encoded states preparation error channel then writes with $\tilde{\varepsilon}_1$ and $\tilde{\varepsilon}_2$ exponentially suppressed in ε:

$$\mathcal{N}^{\varepsilon}_{\mathrm{prep}} = \mathcal{N}^{\varepsilon}_{\mathrm{T,loc}} \circ \mathcal{N}^{\tilde{\varepsilon}_1}_{\mathrm{T,nc}} \circ \mathcal{N}^{\tilde{\varepsilon}_2}_{\mathrm{M,nc}}. \tag{4.27}$$

The two non-correctable terms contribute to the final logical error rate but are exponentially rare. In the following subsections we prove that low enough local stochastic noise is corrected by the tetrahelix code. Post-processing of final single qubit measurements in the form of the tetrahelix code decoding then yields the value of the logical measurement. In the following we analyze error configurations and describe an efficient decoder from 2D and 3D color code decoders.

4.5.2 Existence of a good decoder

In the 3D color code, the logical $\overline{Z}$ operator corresponds to strings of Pauli Z connecting the four boundaries of different colors. An extremal vertex of the tetrahedron belongs to three boundaries and a string connecting this vertex to the opposite face of the remaining color hence yields an example of a $\overline{Z}$ logical operator. Logical errors arise when more than half of the respective phases of any such path are flipped. Errors on the tetrahelix code have a similar origin except that error strings can jump between tetrahedra to connect boundaries of different colors as described in Figure 4.7. This means that we cannot individually decode tetrahedra and that we first need to split (in lattice surgery language) the chain to retrieve tetrahedral codes.

This can be performed in software after final single-qubit X measurements

by reconstructing the value of X stabilizers of the tetrahedral codes. Considering the example of the 2-tetrahelix code for simplicity, X stabilizers at the interface between the two tetrahedra were merged and hence, taken individually, do not stabilize the tetrahelix code. This means that they will initially not necessarily be in their $+1$ eigenspace even without errors. This can be fixed by applying Z stabilizers from $\mathcal{P}_{1,2}$ to set them to $+1$ while acting trivially on the code space of the tetrahelix code. This can be seen as preparing two triangular codes (2D color codes) logical states on the two facets by applying a Pauli operator of the form

$$Z(\sigma) \otimes Z(\varphi(\sigma)), \tag{4.28}$$

with $\sigma \subset B_1$ a set of vertices from the triangular facet on which the merge was performed, and φ the bijection between the two tetrahedra sets of vertices defined in Section 4.3. Note here that any potential logical error applied to one tetrahedron would also be applied to the second one.

In reality, errors arising on the support of tetrahedral codes X stabilizers prevents all such stabilizer to be set back to $+1$ by applying Z stabilizers from $\mathcal{P}_{1,2}$. Since in this scheme we aim at correcting errors at the next step during individual tetrahedral codes decoding we only need here to approach the tetrahedral code spaces. This can be done by minimizing the number of tetrahedral code X stabilizers with value -1 in the chain. For the k-tetrahelix code we start by X stabilizers merged between more than two tetrahedra, that is to say on tetrahedra vertices and edges, followed by those on the bulk of the facet on which tetrahedral codes are merged.

Once each tetrahedron is back to the tetrahedral code space (up to physical errors) it suffices to individually decode each code and multiply the logical values of the $\overline{X}_i$'s to recover the desired logical information (and hence pairs of tetrahedral codes logical errors possibly introduced at the splitting step cancel each other). For a low enough error rate we thus expect the logical error rate $\overline{\varepsilon}$ after such decoding to be proportional to the number of tetrahedra in the chain and to the logical error rate of a single tetrahedral code:

$$\overline{\varepsilon} = \mathcal{O}\big(kL^3\big)(\varepsilon/\varepsilon_{\text{th}})^{\mathcal{O}(L)}. \tag{4.29}$$

Here we have only used 2D and 3D color codes decoders and therefore the existence of efficient 2D and 3D color code decoders [39, 15, 134, 92] implies the

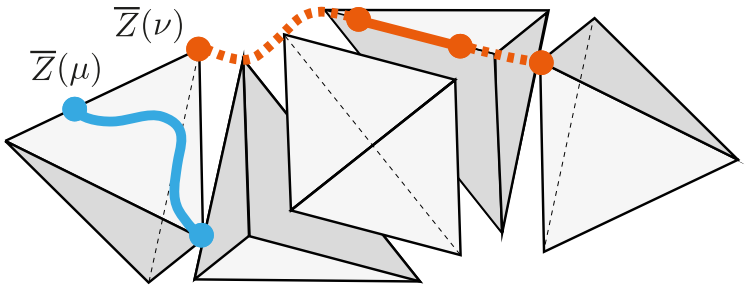

Figure 4.7: Representation of $\overline{Z}$ logical error configurations. Error strings are no longer restricted to a single tetrahedron (blue) but can also connect neighbouring tetrahedra (red). In the tetrahelix stacking, error strings can jump up to two tetrahedra at once.

existence of an efficient decoder for the tetrahelix code. The formal definition and analysis of such a decoder under the general noise model considered here is beyond the scope of this paper and in the following we will prove Theorem 12 by relying on existing results on quantum LDPC codes. To do so we show that the code admits a non-zero threshold independent of k so that for low enough noise the logical error can be made arbitrarily low by increasing L.

4.5.3 Existence of a threshold for minimum weight decoder and local stochastic noise

The tetrahelix code is a quantum LDPC code since its generators have a bounded weight and each qubit is involved in a bounded number of generators. It is known that a family of $[[\tilde{n}, \tilde{k}, \tilde{d}]]$ quantum LDPC codes, with $\tilde{n}$ and $\tilde{d}$ scaling to infinity, and experiencing local stochastic noise of parameter ε, admits a non-zero error threshold ε_{th} [66]. More precisely, below this threshold the logical error rate is exponentially suppressed as

$$\bar{\varepsilon} = \mathcal{O}\Big(\tilde{n}(\varepsilon/\varepsilon_{\text{th}})^{\tilde{d}/2}\Big), \tag{4.30}$$

using the *minimum weight decoder*.

In the case of tetrahelix code, k is in general an independent parameter from L the distance of the code. Applying directly the results of [66] would lead to a threshold dependent on the value of k. We take care of this issue by imposing $k = \mathcal{O}(L)$. Thus, the associated family of k-tetrahelix codes admits a number of physical qubits $\tilde{n} = \mathcal{O}(L^4)$, and such a $[[\tilde{n} = \Theta(kL^3), \tilde{k} = 1, \tilde{d} = \Theta(L)]]$ code

admits a non-zero threshold ε_{th} with L scaling to infinity. Note that, while the minimum-weight decoder is not efficient in general, we expect the efficient decoder of the previous subsection to present similar error suppression property.

In the following subsection, we show that imposing $k = \mathcal{O}(L)$ is compatible with the desired parallelization and fault-tolerance properties.

4.5.4 Proof of Theorem 12

When implementing sparse IQP circuits on N qubits on k-tetrahelix codes, k, L and N are related through three relations. First, as discussed in the previous subsection, to ensure the existence of threshold independent of k, we need to have

$$k = \mathcal{O}(L). \tag{4.31}$$

Second, an arbitrarily large fraction of sparse IQP circuits on N qubits are of depth $\Theta(\log N)$ and can hence be implemented on k-tetrahelix codes with:

$$k = \Theta(\log N), \tag{4.32}$$

Third, another relation between L and N results from the required code size to reach the target precision δ of the sparse IQP problem. A logical sparse IQP circuit D is implemented with a k-tetrahelix code by the circuit C_D. After the final decoding, one obtains samples from a distribution $p_{D,\bar{\varepsilon}}$. For a constant logical error rate $\bar{\varepsilon}$ per logical qubit, the union bound gives an upper bound to the distance between the noisy and the ideal probability distributions with respect to N:

$$\|p_D - p_{D,\bar{\varepsilon}}\|_{\text{TV}} \leq \mathcal{O}(N \cdot \bar{\varepsilon}). \tag{4.33}$$

Keeping the noisy probability distribution δ-close to the ideal distribution thus imposes a logical error rate $\bar{\varepsilon}$ at most $\mathcal{O}(\delta/N)$. Logical errors can arise both from local stochastic errors and remaining non-local residual errors induced by merging errors or tetrahedral encoded states preparation errors, all of which are exponentially suppressed in L below some threshold:

$$\bar{\varepsilon} = \left(\frac{\varepsilon}{\varepsilon_{\text{th}}}\right)^{\Theta(L)}, \tag{4.34}$$

where the polynomial dependency on L in equation (4.30) is absorbed by the exponential. From (4.33) and (4.34), we derive the third equation relating L and N,

$$L = \Omega\left(\frac{\log(N/\delta)}{\log(\varepsilon_{\text{th}}/\varepsilon)}\right). \tag{4.35}$$

For a given N, it is enough to take

$$L = \Theta\left(\frac{\log(N/\delta)}{\log(\varepsilon_{\text{th}}/\varepsilon)}\right) \quad \text{and} \quad k = \Theta(\log N), \tag{4.36}$$

which automatically also satisfy (4.32). The total number of qubits n of each code block for a sparse IQP circuit of width N then reduces to:

$$n = \Theta(kL^3) = \Theta\left(\frac{\text{polylog}(N/\delta)}{\text{polylog}(\varepsilon_{\text{th}}/\varepsilon)}\right). \tag{4.37}$$

This completes the proof of Theorem 12. We note that for the (arbitrarily small) fraction of sparse IQP circuits of super-logarithmic depth the overhead is at most polynomial since the depth of sparse IQP circuits is at most linear.

Chapter 5

Conclusion and perspectives

5.1 Conclusion

The work presented in this manuscript investigates proposals for near-term quantum devices operating in an intermediate regime between NISQ and fully universal fault-tolerant quantum computing, leveraging the capabilities offered by modern physical platforms.

In Chapters 2 and 3, we focus on superconducting platforms, which face significant overhead due to the control infrastructure required for each qubit. Since this overhead poses a major obstacle to scalability, we explore local decoding strategies that mitigate the need for long-range connectivity. In particular, we consider biased-noise qubits—such as cat qubits [112, 127] for which we propose designs for one-dimensional local quantum memories. In the short term, where equipping the device with additional local classical memory may be experimentally challenging, we give in Chapter 2 a quantum memory architecture based only on two rows of qubits with nearest-neighbor connectivity. Looking further ahead, we propose augmenting the system with local classical memory to enable asymptotic suppression of logical errors with system size. This approach builds on a new local decoder that interprets odd parities in the repetition code as defects, which interact via classical binary signal exchange that effectively attracts them toward one another. The hardware implementation of these proposals remains an open and intersting question. Since they involve only a very small number of bits, it is conceivable that they could be realized using classical microelectronics integrated with the superconducting device—either through superconducting digital circuits based on Single Flux Quantum (SFQ) logic, in the spirit of [148], or possibly using CMOS transistors. The dissipative nature of CMOS technology will however make it challenging to operate next to superconducting devices working at $\sim$ 20mK. An alternative idea is to use biased-noise qubits, such as cat qubits, as classical bits within the decoder lever-

aging a simplified coupling—potentially eliminating the need for measurement, since entropy could be removed through simple resets. An interesting question in this approach is to what extent, given that we do not intend to protect the phase of those 'decoder cat qubits', their construction can be simplified. This could make it feasible to equip each data cat qubit with a few 'decoder cat qubits' at low overhead cost. We hope our results will motivate further work on such hybrid quantum-classical memories.

In Chapter 4, we explore fault-tolerant protocols specifically designed for an alternative quantum architecture based on a reconfigurable array of atoms. By exploiting the inherent parallelism of this platform, we proposed a constant-depth, fault-tolerant implementation of sparse IQP circuits—offering a promising route toward near- or mid-term demonstrations of super-polynomial quantum advantage. Our scheme relies on a single round of classical feed-forward to prepare encoded states, followed by a single layer of transversal gates to implement the logical circuit. Central to this approach is the tetrahelix code, which supports single-shot preparation of logical $|+\rangle$ states and enables transversal implementation of IQP circuits. We believe the tetrahelix code is of independent interest, as it possesses a large transversal gate set and, in this construction, already demonstrates the potential of highly parallel quantum computation. The qubit overhead and classical computation time scale only polylogarithmically with the width of the original sparse IQP circuit. Notably, the experimental requirements of our protocol are nearly compatible with current NISQ hardware. We hope this work can bring within reach demonstration of robust super-polynomial advantage of quantum over classical computation.

5.2 Perspectives

5.2.1 Towards optimal local decoders

Given the large number of the decoders introduced in Chapter 3, we do not claim optimality of the decoders in terms of either performance or resource efficiency. The charge conservation property of those automata gives them a high degree of structure, a desirable property enabling to reduce the space size in which to look for better performing variants. While, to the best of our knowledge, charge conservation in cellular automata has not been studied extensively, a related construction is that of number-conserving automata that conserves a number of

particles, an example of which are lattice gases [154] widely used in physics, or the Nagel-Schreckenberg model [115] for street traffic. Significant work has been done to characterize such automata in low dimensions [76, 130, 155, 18, 19], that is to say to list all such automata for given state space and neighborhood. Extending the framework of number-conserving cellular automata to incorporate signed quantities (i.e., charges rather than particle counts) would prove particularly useful to look for charge-conserving decoder with good performances. A more general but promising approach to optimize the local update rule involves reinforcement learning [123]. Overall, we believe there remains significant room for performance improvement, which we leave for future work.

Regarding the asymptotic scaling, it appears that all known local decoders share a common limitation: they do not correct errors up to the full code distance d, but only up to a sublinear power of d. For example, Toom's rule corrects errors up to $d^{1/2}$, while Tsirelson's decoder achieves scaling of $d^{\log_3 2}$. It remains unclear whether such sublinear scaling reflects an inherent limitation of local decoders in finite dimension, or whether local decoders capable of correcting all errors of linear weight with the code distance exist. Note that this is the case in dimension n, where good qLDPC codes exist with distance linear in n and for which local decoders correct up to a constant fraction of the distance [101]. Going back to the finite dimension case, notably, if a local decoder is capable of correcting isolated errors within a space-time box whose size scales with the spatial extent of the error (as is the case for the signal-based decoders introduced in Chapter 3), then it becomes possible to construct error configurations of weight d^α for some $\alpha < 1$ that lead to a logical failure. These are obtained by recursively puncturing holes into a logical operator, leading to what we refer to in Chapter 3 as *fractal-like error configurations*. A natural follow-up question is then whether the ability to correct such configurations—despite their sublinear weight—is a necessary condition for a local decoder to exhibit a threshold. We leave this as an open problem.

5.2.2 Towards an efficient cellular automaton decoder for the surface code

Another interesting research direction is the extension of the rule to a 2D lattice, which is relevant for surface code decoding, the surface code remaining as of today the leading experimental platform [17, 1]. The biggest challenge when

generalising signal-based decoder to the surface code is that a natural analogous of point-like particle on a 1D lattice would be string-like wave-front on a 2D lattice. This poses the problem that, contrary to point-like particle, string-like particles cannot be erased locally and instantaneously. A possible fix would be to engineer a mechanism so that front-wave can be erased progressively from local interactions, at the cost of an increasingly complicated dynamics. An alternative construction, that we believe to be more promising, retains point-like particles that are this time sent in the four cardinal directions but introduces a new 'reflection' mechanism that causes defects to be attracted to each other via intermediary reflection points. Note that for the erasure mechanism to work in this new setting, the automaton must also keep track of particles that were reflected, so that anti-signals can follow those reflections, and the automaton is ensured to return to the zero configuration once the initial error is corrected.

Figure 5.1 illustrates the remaining challenges that an automaton, operating in the setting described in the previous paragraph, would need to address in order to function correctly. First, the transition rule should ensure that elliptic dynamics between defects are prohibited, where defects perform loop around each other but without actually getting closer from each other. Indeed, since in 2D there are only two bi-segment matching trajectories—vertical then horizontal, or vice versa—that yield the same logical outcome, it is crucial to prevent the two defects from locally choosing distinct matching trajectories. For example in Figure 5.1 (a), we do not want the bottom left defect to start the matching trajectory by going up while on the other hand the top right defect is going left. Preventing this could be achieved, for example, by allowing defects to move only in the south and west directions. Then, notice that contrary to the 1D case, in 2D defects can 'avoid' signals by moving in an orthogonal direction to the signal propagation direction. This is problematic in the phenomenological model because a single error that displaces a defect by one step in the wrong direction can delay the decoding process by $\mathcal{O}(d)$ as illustrated in Figure 5.1 (b). This typically prevents the decoder from having a threshold, since beyond a certain system size, defects are very unlikely to remain stationary throughout the entire error-correction process. We believe this caveat can be fixed by an appropriate hierarchical structure, so that signals sent from neighboring sites are regrouped at long distance into the same trajectories, similar to a multi-level road network with local streets, regional roads, and national highways. We leave this for future work.

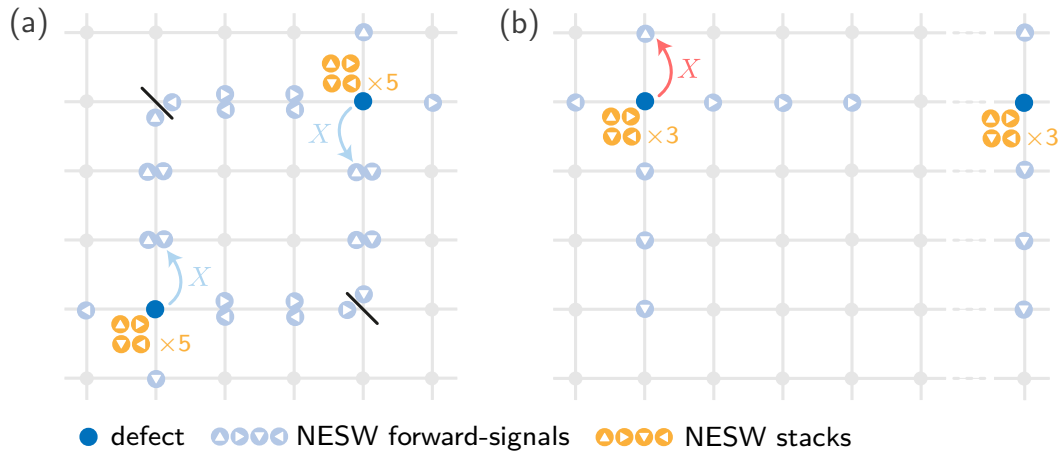

Figure 5.1: Naive candidate for a signal-based surface code decoder. Signals are sent in all four cardinal directions indicated by a white triangle (each with an associated stack), and reflect upon collision (black lines) on which site is counted the number of each type of reflections (not shown). (a) A naive automaton could induce elliptic dynamics between defects because sites with a defect that receive signals from two orthogonal directions need to chose locally in which direction to displace the defect. In the example, the defect on the top right could be displaced to the bottom, while the bottom left defect could be displaced upward (correction are shown in blue). (b) A single error (in red) can displace a defect by 1 in an orthogonal direction to the one pointing towards the target defect. In this case the accumulated stack decrements into anti-signals that will erase all forward-signal sent in this direction until then. In the worst case if the other defect with which it should be paired is at distance $\mathcal{O}(d)$, the process delays the error-correction process by the same amount, which is expected to prevent the memory lifetime to scale arbitrarily with sytem size.

5.2.3 On parallel implementation of commuting gates

In Chapter 4, we have proposed a fault-tolerant implementation of sparse IQP circuits, paving the way for demonstrations of super-polynomial quantum advantage in near or mid-term experiments. To do this we have introduced the tetrahelix code admitting single-shot preparation of encoded $|+\rangle$ states and transversal implementation of IQP circuits. The tetrahelix code has interesting properties in itself. The ability to implement many different non-Clifford unitaries in a transversal manner could potentially be leveraged in other settings. One can take inspiration from this construction to design other codes with large sets of transversal non-Clifford unitaries to locally trade depth for width in larger scale algorithms. The key ingredient of our approach is the commuting nature of sparse IQP gates that enables their parallelization, in the spirit of [36] for MBQC, but in a fault-tolerant manner. Note that a generalization of the construction to any set of commuting gate could have powerful applications, for example, it would be interesting to investigate the case of commuting CNOT circuits, an example of which is the powerful quantum fan-out gate [80].

Another interesting research direction concerns the apparent trade-off between transversal gates in the Z basis and the X-distance. While this trade-off is intuitive to some extent, it is not clear whether it can be asymptotically improved in finite-dimensional systems. This is however of particular practical interest, since the scheme of Chapter 4 only uses the X distance, so that balancing the X and Z distances out would contribute to making the scheme more accessible to current experiments.

Bibliography

[1] Quantum error correction below the surface code threshold. Nature, 638(8052):920–926, 2025.

[2] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In Proceedings of the forty-third annual ACM symposium on Theory of computing, pages 333–342, 2011.

[3] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pages 176–188, 1997.

[4] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 945–957, 2023.

[5] Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. Nature, 614(7949):676–681, 2023.

[6] Victor V Albert and Philippe Faist, editors. The Error Correction Zoo. 2025.

[7] Robert Alicki, Michal Horodecki, Pawel Horodecki, and Ryszard Horodecki. On thermal stability of topological qubit in kitaev's 4d model. Open Systems & Information Dynamics, 17(01):1–20, 2010.

[8] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. Nature, 574(7779):505–510, 2019.

[9] Alain Aspect, Jean Dalibard, and Gérard Roger. Experimental test of bell's inequalities using time-varying analyzers. Physical Review Letters, 49(25):1804, 1982.

[10] Shankar Balasubramanian, Margarita Davydova, and Ethan Lake. A local automaton for the 2d toric code. arXiv preprint arXiv:2412.19803, 2024.

[11] Boaz Barak, Chi-Ning Chou, and Xun Gao. Spoofing linear cross-entropy benchmarking in shallow quantum circuits. arXiv preprint arXiv:2005.02421, 2020.

[12] John S Bell. On the einstein podolsky rosen paradox. Physics Physique Fizika, 1(3):195, 1964.

[13] Elwyn R Berlekamp, John H Conway, and Richard K Guy. Winning ways for your mathematical plays, volume 4. AK Peters/CRC Press, 2004.

[14] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. In Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pages 11–20, 1993.

[15] Michael E Beverland, Aleksander Kubica, and Krysta M Svore. Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes. PRX Quantum, 2(2):020341, 2021.

[16] Dolev Bluvstein, Simon J Evered, Alexandra A Geim, Sophie H Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, et al. Logical quantum processor based on reconfigurable atom arrays. Nature, 626(7997):58–65, 2024.

[17] Dolev Bluvstein, Alexandra A. Geim, Sophie H. Li, Simon J. Evered, J. Pablo Bonilla Ataides, Gefen Baranes, Andi Gu, Tom Manovitz, Muqing Xu, Marcin Kalinowski, Shayan Majidy, Christian Kokail, Nishad Maskara, Elias C. Trapp, Luke M. Stewart, Simon Hollerith, Hengyun Zhou, Michael J. Gullans, Susanne F. Yelin, Markus Greiner, Vladan Vuletic, Madelyn Cain, and Mikhail D. Lukin. Architectural mechanisms of a universal fault-tolerant quantum computer, 2025.

[18] Nino Boccara and Henryk Fuks. Cellular automaton rules conserving the number of active sites. Journal of Physics A: Mathematical and General, 31(28):6007, 1998.

[19] Nino Boccara and Henryk Fukś. Number-conserving cellular automaton rules. Fundamenta Informaticae, 52(1-3):1–13, 2002.

[20] AH Boerdijk. Some remarks concerning close-packing of equal spheres. Philips Research Reports, 7:303–313, 1952.

[21] H Bombin and MA Martin-Delgado. Exact topological quantum order in d= 3 and beyond: Branyons and brane-net condensates. Physical Review B, 75(7):075103, 2007.

[22] Héctor Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. New Journal of Physics, 17(8):083002, 2015.

[23] Héctor Bombín. Single-shot fault-tolerant quantum error correction. Physical Review X, 5(3):031043, 2015.

[24] Hector Bombin and Miguel Angel Martin-Delgado. Topological quantum distillation. Physical Review Letters, 97(18):180501, 2006.

[25] Hector Bombin and Miguel-Angel Martin-Delgado. Topological computation without braiding. Physical Review Letters, 98(16):160502, 2007.

[26] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. Nature Physics, 15(2):159–163, 2019.

[27] Sergey Bravyi, David Gosset, Robert Koenig, and Marco Tomamichel. Quantum advantage with noisy shallow circuits. Nature Physics, 16(10):1040–1045, 2020.

[28] Sergey Bravyi and Jeongwan Haah. Magic-state distillation with low overhead. Physical Review A, 86(5):052329, 2012.

[29] Sergey Bravyi and Jeongwan Haah. Quantum self-correction in the 3d cubic code model. Physical Review Letters, 111(20):200501, 2013.

[30] Sergey Bravyi and Barbara Terhal. A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes. New Journal of Physics, 11(4):043029, 2009.

[31] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 467(2126):459–472, 2011.

[32] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. Physical Review Letters, 117(8):080501, 2016.

[33] Michael J Bremner, Ashley Montanaro, and Dan J Shepherd. Achieving quantum supremacy with sparse and noisy commuting quantum computations. Quantum, 1:8, 2017.

[34] Nikolas P Breuckmann, Kasper Duivenvoorden, Dominik Michels, and Barbara M Terhal. Local decoders for the 2d and 4d toric code. arXiv preprint arXiv:1609.00510, 2016.

[35] Benjamin J Brown, Daniel Loss, Jiannis K Pachos, Chris N Self, and James R Wootton. Quantum memories at finite temperature. Reviews of Modern Physics, 88(4):045005, 2016.

[36] Dan Browne, Elham Kashefi, and Simon Perdrix. Computational depth complexity of measurement-based quantum computation. In Theory of Quantum Computation, Communication, and Cryptography: 5th Conference, TQC 2010, Leeds, UK, April 13-15, 2010, Revised Selected Papers 5, pages 35–46. Springer, 2011.

[37] Ana Bušić, Nazim Fates, Jean Mairesse, and Irene Marcovici. Density classification on infinite lattices and trees. In Latin American Symposium on Theoretical Informatics, pages 109–120. Springer, 2012.

[38] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. Physical Review A, 54(2):1098, 1996.

[39] Christopher Chamberland, Aleksander Kubica, Theodore J Yoder, and Guanyu Zhu. Triangular color codes on trivalent graphs with flag qubits. New Journal of Physics, 22(2):023019, 2020.

[40] Christopher Chamberland, Kyungjoo Noh, Patricio Arrangoiz-Arriola, Earl T Campbell, Connor T Hann, Joseph Iverson, Harald Putterman, Thomas C Bohdanowicz, Steven T Flammia, Andrew Keller, et al. Building a fault-tolerant quantum computer using concatenated cat codes. PRX Quantum, 3(1):010329, 2022.

[41] Stefano Chesi, Beat Röthlisberger, and Daniel Loss. Self-correcting quantum memory in a thermal environment. Physical Review A—Atomic, Molecular, and Optical Physics, 82(2):022305, 2010.

[42] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. Reducing the number of qubits in quantum factoring. Cryptology ePrint Archive, 2024.

[43] Neng-Chun Chiu, Elias C. Trapp, Jinen Guo, Mohamed H. Abobeih, Luke M. Stewart, Simon Hollerith, Pavel Stroganov, Marcin Kalinowski, Alexandra A. Geim, Simon J. Evered, Sophie H. Li, Lisa M. Peters, Dolev Bluvstein, Tout T. Wang, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Continuous operation of a coherent 3,000-qubit system, 2025.

[44] BS Cirel'son. Reliable storage of information in a system of unreliable components with local interactions. In Locally Interacting Systems and Their Application in Biology: Proceedings of the School-Seminar on Markov Interaction Processes in Biology, Held in Pushchino, Moscow Region, March, 1976, pages 15–30. Springer, 2006.

[45] HSM Coxeter and JM Wills. Regular complex polytopes. Jahresbericht der Deutschen Mathematiker Vereinigung, 96(1):2–2, 1994.

[46] Alexander M Dalzell, Nicholas Hunter-Jones, and Fernando GSL Brandão. Random quantum circuits anticoncentrate in log depth. PRX Quantum, 3(1):010333, 2022.

[47] Guillaume Dauphinais and David Poulin. Fault-tolerant quantum error correction for non-abelian anyons. Communications in Mathematical Physics, 355:519–560, 2017.

[48] Nicolas Delfosse and Naomi H Nickerson. Almost-linear time decoding algorithm for topological codes. Quantum, 5:595, 2021.

[49] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. Journal of Mathematical Physics, 43(9):4452–4505, 2002.

[50] Benoît Douçot and Lev B Ioffe. Physical implementation of protected qubits. Reports on Progress in Physics, 75(7):072001, 2012.

[51] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. Physical Review Letters, 104(5):050504, 2010.

[52] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. Physical Review Letters, 102(11):110502, 2009.

[53] Jack Edmonds. Paths, trees, and flowers. Canadian Journal of mathematics, 17:449–467, 1965.

[54] Albert Einstein. Über einen die erzeugung und verwandlung des lichtes betreffenden heuristischen gesichtspunkt, 1905.

[55] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Constant overhead quantum fault tolerance with quantum expander codes. Communications of the ACM, 64(1):106–114, 2020.

[56] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. Physical Review A—Atomic, Molecular, and Optical Physics, 86(3):032324, 2012.

[57] Stuart J Freedman and John F Clauser. Experimental test of local hidden-variable theories. Physical Review Letters, 28(14):938, 1972.

[58] Keisuke Fujii and Tomoyuki Morimae. Commuting quantum circuits and complexity of ising partition functions. New Journal of Physics, 19(3):033003, 2017.

[59] Henryk Fuks-acute. Solution of the density classification problem with two cellular automata rules. Physical Review E, 55(3):R2081, 1997.

[60] Péter Gács. Reliable cellular automata with self-organization. Journal of Statistical Physics, 103:45–267, 2001.

[61] Péter Gács, Georgy L Kurdyumov, and Leonid Anatolevich Levin. One-dimensional uniform arrays that wash out finite islands. Problemy Peredachi Informatsii, 14(3):92–96, 1978.

[62] Craig Gidney. How to factor 2048 bit rsa integers with less than a million noisy qubits. arXiv preprint arXiv:2505.15917, 2025.

[63] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum, 5:433, 2021.

[64] Leslie Ann Goldberg and Heng Guo. The complexity of approximating complex-valued ising and tutte partition functions. computational complexity, 26:765–833, 2017.

[65] Daniel Gottesman. Stabilizer codes and quantum error correction. California Institute of Technology, 1997.

[66] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. arXiv preprint arXiv:1310.2984, 2013.

[67] Daniel Gottesman and Isaac L Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. Nature, 402(6760):390–393, 1999.

[68] Lawrence F Gray. A reader's guide to gacs's "positive rates" paper. Journal of Statistical Physics, 103:1–44, 2001.

[69] Lov K Grover. A fast quantum mechanical algorithm for database search. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 212–219, 1996.

[70] Thiago LM Guedes, Don Winter, and Markus Müller. Quantum cellular automata for quantum error correction and density classification. Physical Review Letters, 133(15):150601, 2024.

[71] Jérémie Guillaud and Mazyar Mirrahimi. Repetition cat qubits for fault-tolerant quantum computation. Physical Review X, 9(4):041053, 2019.

[72] Jeongwan Haah. Local stabilizer codes in three dimensions without string logical operators. Physical Review A—Atomic, Molecular, and Optical Physics, 83(4):042330, 2011.

[73] Dominik Hangleiter and Jens Eisert. Computational advantage of quantum random sampling. Reviews of modern physics, 95, 2023.

[74] James William Harrington. Analysis of quantum error-correcting codes: symplectic lattice codes and toric codes. PhD thesis, California Institute of Technology, 2004.

[75] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical Review Letters, 103(15):150502, 2009.

[76] Tetsuya Hattori and Shinji Takesue. Additive conserved quantities in discrete-time lattice dynamical systems. Physica D: Nonlinear Phenomena, 49(3):295–322, 1991.

[77] Michael Herold, Earl T Campbell, Jens Eisert, and Michael J Kastoryano. Cellular-automaton decoders for topological quantum memories. npj Quantum information, 1(1):1–8, 2015.

[78] Michael Herold, Michael J Kastoryano, Earl T Campbell, and Jens Eisert. Cellular automaton decoders of topological quantum memories in the fault tolerant setting. New Journal of Physics, 19(6):063012, 2017.

[79] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. New Journal of Physics, 14(12):123011, 2012.

[80] Peter Høyer and Robert Špalek. Quantum fan-out is powerful. Theory of computing, 1(1):81–103, 2005.

[81] Pavithran Iyer and David Poulin. Hardness of decoding quantum stabilizer codes. IEEE Transactions on Information Theory, 61(9):5209–5223, 2015.

[82] Stephen P Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V Isakov, Tanuj Khattar, and Ryan Babbush. Optimization by decoded quantum interferometry. arXiv preprint arXiv:2408.08292, 2024.

[83] Alastair Kay and Roger Colbeck. Quantum self-correcting stabilizer codes. arXiv preprint arXiv:0810.3557, 2008.

[84] A Yu Kitaev. Quantum computations: algorithms and error correction. Russian Mathematical Surveys, 52(6):1191, 1997.

[85] A Yu Kitaev. Fault-tolerant quantum computation by anyons. Annals of physics, 303(1):2–30, 2003.

[86] Emanuel Knill, Raymond Laflamme, and Wojciech H Zurek. Resilient quantum computation. Science, 279(5349):342–345, 1998.

[87] Jens Koch, Terri M Yu, Jay Gambetta, Andrew A Houck, David I Schuster, Johannes Majer, Alexandre Blais, Michel H Devoret, Steven M Girvin, and Robert J Schoelkopf. Charge-insensitive qubit design derived from the cooper pair box. Physical Review A—Atomic, Molecular, and Optical Physics, 76(4):042319, 2007.

[88] Anna Kómár, Olivier Landon-Cardinal, and Kristan Temme. Necessity of an energy barrier for self-correction of abelian quantum doubles. Physical Review A, 93(5):052337, 2016.

[89] Yasuhiro Kondo, Ryuhei Mori, and Ramis Movassagh. Quantum supremacy and hardness of estimating output probabilities of quantum circuits. In 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), pages 1296–1307. IEEE, 2022.

[90] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, et al. Realizing repeated quantum error correction in a distance-three surface code. Nature, 605(7911):669–674, 2022.

[91] Aleksander Kubica and Michael E Beverland. Universal transversal gates with color codes: A simplified approach. Physical Review A, 91(3):032330, 2015.

[92] Aleksander Kubica and Nicolas Delfosse. Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders. Quantum, 7:929, 2023.

[93] Aleksander Kubica and John Preskill. Cellular-automaton decoders with provable thresholds for topological codes. Physical Review Letters, 123(2):020501, 2019.

[94] Ethan Lake. Fast offline decoding with local message-passing automata. arXiv preprint arXiv:2506.03266, 2025.

[95] Mark Land and Richard K Belew. No perfect two-state cellular automata for density classification exists. Physical Review Letters, 74(25):5148, 1995.

[96] Andrew J. Landahl and Ciaran Ryan-Anderson. Quantum computing by color-code lattice surgery, 2014.

[97] Rolf Landauer. The physical nature of information. Physics letters A, 217(4-5):188–193, 1996.

[98] Olivier Landon-Cardinal, Beni Yoshida, David Poulin, and John Preskill. Perturbative instability of quantum memory based on effective long-range interactions. Physical Review A, 91(3):032303, 2015.

[99] Zaki Leghtas, Steven Touzard, Ioan M Pop, Angela Kou, Brian Vlastakis, Andrei Petrenko, Katrina M Sliwa, Anirudh Narla, Shyam Shankar, Michael J Hatridge, et al. Confining the state of light to a quantum manifold by engineered two-photon loss. Science, 347(6224):853–857, 2015.

[100] Anthony Leverrier and Gilles Zémor. Quantum tanner codes. In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), pages 872–883. IEEE, 2022.

[101] Anthony Leverrier and Gilles Zémor. Decoding quantum tanner codes. IEEE Transactions on Information Theory, 69(8):5100–5115, 2023.

[102] Simon Lieu, Yu-Jie Liu, and Alexey V Gorshkov. Candidate for a passively protected quantum memory in two dimensions. Physical Review Letters, 133(3):030601, 2024.

[103] Thomas Milton Liggett and Thomas M Liggett. Interacting particle systems, volume 2. Springer, 1985.

[104] Ting-Chun Lin, Hsin-Po Wang, and Min-Hsiu Hsieh. Proposals for 3d self-correcting quantum memory. arXiv preprint arXiv:2411.03115, 2024.

[105] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. Quantum, 3:128, 2019.

[106] Austin P Lund, Michael J Bremner, and Timothy C Ralph. Quantum sampling problems, bosonsampling and quantum supremacy. npj Quantum Information, 3(1):1–8, 2017.

[107] Atsushi Masumori, Lana Sinapayen, and Takashi Ikegami. Simulation of gacs' automaton. arXiv preprint arXiv:2405.04060, 2024.

[108] Robert J McEliece. A public-key cryptosystem based on algebraic. Coding Thv, 4244(1978):114–116, 1978.

[109] Rawad Mezher, Joe Ghalbouni, Joseph Dgheim, and Damian Markham. Fault-tolerant quantum speedup from constant depth quantum circuits. Physical Review Research, 2(3):033444, 2020.

[110] Kamil Michnicki. Towards self-correcting quantum memories. University of Washington, 2015.

[111] Kamil P Michnicki. 3d topological quantum memory with a power-law energy barrier. Physical Review Letters, 113(13):130501, 2014.

[112] Mazyar Mirrahimi, Zaki Leghtas, Victor V Albert, Steven Touzard, Robert J Schoelkopf, Liang Jiang, and Michel H Devoret. Dynamically protected cat-qubits: a new paradigm for universal quantum computation. New Journal of Physics, 16(4):045014, 2014.

[113] Edward F Moore et al. Gedanken-experiments on sequential machines. Automata studies, 34:129–153, 1956.

[114] Ramis Movassagh. The hardness of random quantum circuits. Nature Physics, 19(11):1719–1724, 2023.

[115] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. Journal de physique I, 2(12):2221–2229, 1992.

[116] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. Cambridge university press, 2010.

[117] Yingkai Ouyang. Permutation-invariant quantum codes. Physical Review A, 90(6):062317, 2014.

[118] Adam Paetznick and Ben W Reichardt. Universal fault-tolerant quantum computation with only transversal gates<? format?> and error correction. Physical Review Letters, 111(9):090505, 2013.

[119] Louis Paletta, Anthony Leverrier, Mazyar Mirrahimi, and Christophe Vuillot. Local automaton decoder for defect matching in 1d. arXiv, 2025.

[120] Louis Paletta, Anthony Leverrier, Alain Sarlette, Mazyar Mirrahimi, and Christophe Vuillot. Robust sparse iqp sampling in constant depth. Quantum, 8:1337, 2024.

[121] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical ldpc codes. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, pages 375–388, 2022.

[122] Kihong Park. Ergodicity and mixing rate of one-dimensional cellular automata. PhD thesis, Boston University Computer Science Department, 1997.

[123] Mincheol Park, Nishad Maskara, Marcin Kalinowski, and Mikhail D Lukin. Enhancing quantum memory lifetime with measurement-free local error correction and reinforcement learning. arXiv preprint arXiv:2408.09524, 2024.

[124] John Preskill. Quantum computing and the entanglement frontier. arXiv preprint arXiv:1203.5813, 2012.

[125] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, 2018.

[126] John Preskill. Beyond nisq: The megaquop machine. ACM Transactions on Quantum Computing, 6(3):1–7, 2025.

[127] Shruti Puri, Samuel Boutin, and Alexandre Blais. Engineering the quantum states of light in a kerr-nonlinear resonator by two-photon driving. npj Quantum Information, 3(1):18, 2017.

[128] Harald Putterman, Kyungjoo Noh, Connor T Hann, Gregory S MacCabe, Shahriar Aghaeimeibodi, Rishi N Patel, Menyoung Lee, William M Jones, Hesam Moradinejad, Roberto Rodriguez, et al. Hardware-efficient quantum error correction using concatenated bosonic qubits. arXiv preprint arXiv:2409.13025, 2024.

[129] Joel Rajakumar, James D Watson, and Yi-Kai Liu. Polynomial-time classical simulation of noisy iqp circuits with constant depth. In Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1037–1056. SIAM, 2025.

[130] Markus Redeker. Number conservation via particle flow in one-dimensional cellular automata. Fundamenta Informaticae, 187, 2022.

[131] Pedro Sales Rodriguez, John M Robinson, Paul Niklas Jepsen, Zhiyang He, Casey Duckering, Chen Zhao, Kai-Hsin Wu, Joseph Campo, Kevin Bagnall, Minho Kwon, et al. Experimental demonstration of logical magic state distillation. arXiv preprint arXiv:2412.15165, 2024.

[132] Wouter Rozendaal and Gilles Zémor. A worst-case analysis of a renormalisation decoder for kitaev's toric code. In 2023 IEEE International Symposium on Information Theory (ISIT), pages 625–629. IEEE, 2023.

[133] Diego Ruiz, Jérémie Guillaud, Anthony Leverrier, Mazyar Mirrahimi, and Christophe Vuillot. Ldpc-cat codes for low-overhead quantum computing in 2d. arXiv preprint arXiv:2401.09541, 2024.

[134] Kaavya Sahay and Benjamin J Brown. Decoder for the triangular color code by matching on a möbius strip. PRX Quantum, 3(1):010310, 2022.

[135] Claus-Peter Schnorr and Adi Shamir. An optimal sorting algorithm for mesh connected computers. In Proceedings of the eighteenth annual ACM symposium on Theory of computing, pages 255–263, 1986.

[136] Thomas Schuster, Chao Yin, Xun Gao, and Norman Y Yao. A polynomial-time classical algorithm for noisy quantum circuits. Physical Review X, 15(4):041018, 2025.

[137] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 465(2105):1413–1439, 2009.

[138] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science, pages 124–134. Ieee, 1994.

[139] Peter W Shor. Scheme for reducing decoherence in quantum computer memory. Physical review A, 52(4):R2493, 1995.

[140] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review, 41(2):303–332, 1999.

[141] Karthik Siva and Beni Yoshida. Topological order and memory time in marginally-self-correcting quantum memory. Physical Review A, 95(3):032324, 2017.

[142] Andrew M Steane. Error correcting codes in quantum theory. Physical Review Letters, 77(5):793, 1996.

[143] Larry J Stockmeyer. The polynomial-time hierarchy. Theoretical Computer Science, 3(1):1–22, 1976.

[144] Kristan Temme. Thermalization time bounds for pauli stabilizer hamiltonians. Communications in Mathematical Physics, 350:603–637, 2017.

[145] Barbara M Terhal. Quantum error correction for quantum memories. Reviews of Modern Physics, 87(2):307, 2015.

[146] Andrei Toom. Cellular automata with errors: problems for students of probability. Topics in contemporary probability and its applications, pages 117–157, 1995.

[147] Andrei L Toom. Stable and attractive trajectories in multicomponent systems. Multicomponent random systems, 6:549–575, 1980.

[148] Yosuke Ueno, Masaaki Kondo, Masamitsu Tanaka, Yasunari Suzuki, and Yutaka Tabuchi. Qecool: On-line quantum error correction with a superconducting decoder for surface code. In 2021 58th ACM/IEEE Design Automation Conference (DAC), pages 451–456. IEEE, 2021.

[149] Michael Vasmer, Dan E Browne, and Aleksander Kubica. Cellular automaton decoders for topological quantum codes with noisy measurements and beyond. Scientific reports, 11(1):2027, 2021.

[150] Anastasios Vergis, Kenneth Steiglitz, and Bradley Dickinson. The complexity of analog computation. Mathematics and computers in simulation, 28(2):91–113, 1986.

[151] John Von Neumann. The general and logical theory of automata. In Systems research for behavioral science, pages 97–107. Routledge, 2017.

[152] Christophe Vuillot. Fault-Tolerant Quantum Computation: Theory and Practice. PhD thesis, TU Delft, 2020.

[153] Gregor Weihs, Thomas Jennewein, Christoph Simon, Harald Weinfurter, and Anton Zeilinger. Violation of bell's inequality under strict einstein locality conditions. Physical Review Letters, 81(23):5039, 1998.

[154] Dieter A Wolf-Gladrow. Lattice-gas cellular automata and lattice Boltzmann models: an introduction. Springer, 2004.

[155] Barbara Wolnik, Anna Nenca, Jan M Baetens, and Bernard De Baets. A split-and-perturb decomposition of number-conserving cellular automata. Physica D: Nonlinear Phenomena, 413:132645, 2020.

[156] Hayata Yamasaki and Masato Koashi. Time-efficient constant-space-overhead fault-tolerant quantum computation. Nature Physics, 20(2):247–253, 2024.

[157] Beni Yoshida. Feasibility of self-correcting quantum memory and thermal stability of topological order. Annals of Physics, 326(10):2566–2633, 2011.

[158] Hengyun Zhou, Chen Zhao, Madelyn Cain, Dolev Bluvstein, Casey Duckering, Hong-Ye Hu, Sheng-Tao Wang, Aleksander Kubica, and Mikhail D Lukin. Algorithmic fault tolerance for fast quantum computing. arXiv preprint arXiv:2406.17653, 2024.

RÉSUMÉ

Un calcul quantique peut se concevoir comme l'évolution d'un ensemble de bits quantiques, ou qubits, selon les lois de la mécanique quantique — d'un état initial encodant l'entrée d'un problème, à sa solution représentée par l'état final. Une machine capable de réaliser cette tâche, un ordinateur quantique, doit être robuste aux erreurs liées aux imperfections inhérentes à la fabrication et au contrôle de systèmes aussi complexes. Cette thèse étudie la théorie du calcul quantique tolérant aux fautes, adaptée aux contraintes physiques des plateformes contemporaines, en particulier les qubits supraconducteurs et les réseaux d'atomes neutres.
Un calcul quantique tolérant aux fautes impose de corriger les erreurs en temps réel, au prix d'une couche architecturale dédiée dont l'intégration peut être source de bruit supplémentaire. Une simplification possible utilise un modèle de mémoire quantique où l'information est encodée dans une paire de configurations d'un système multi-qubits, stabilisé par l'application uniforme d'une règle locale simple s'inscrivant dans la théorie des automates cellulaires. On propose une solution de court terme, la règle de cisaillement, qui protège avec de bonnes performances un code de répétition unidimensionnel contre les erreurs, mais ne possède pas de seuil. Cette lacune est comblée en ajoutant de petites mémoires classiques locales, utilisées pour definir les automates règles à signaux, où la correction d'erreurs repose sur l'échange de signaux binaires. Combinés avec des qubits à bruit biaisé, comme les qubits de chats, ces automates définissent une mémoire quantique entièrement locale en une dimension.
La dernière partie du manuscrit traite de l'avantage des machines quantiques sur leurs homologues classiques, dont une démonstration pertinente doit tenir compte de la présence de bruit — par exemple, l'échantillonnage exact de la distribution de sortie de certains circuits aléatoires est considéré comme difficile classiquement, mais devient traitable si l'on accepte une solution approchée, reflétant les effets du bruit. Dans ce contexte, on propose un circuit quantique de profondeur constante qui résout un problème d'échantillonnage de manière tolérante aux fautes. La solution convient particulièrement aux plateformes à atomes neutres, et l'on espère qu'elle contribuera à la démonstration expérimentale d'un avantage quantique tolérant aux fautes.

MOTS CLÉS

information quantique, calcul quantique tolérant aux fautes, correction d'erreur quantique, automate cellulaire, décodeur local, avantage quantique, circuits aléatoires quantiques

ABSTRACT

A quantum computation can be described by the evolution of a collection of quantum bits, or qubits, according to the laws of quantum mechanics, from an initial state encoding the input of a problem, to its solution represented by the final state. A machine capable of performing this task, a quantum computer, must be robust against errors caused by the inherent imperfections in the fabrication and control of such complex systems. This thesis examines the theory of fault-tolerant quantum computation, tailored to the physical constraints of today's leading platforms, most notably superconducting circuits and neutral-atom arrays.
A fault-tolerant quantum computation requires to correct errors in real time, at the cost of a dedicated architectural layer whose integration may introduce additional noise. A possible simplification uses a quantum memory model where information is encoded in a pair of configurations of a multi-qubit system, stabilized by the uniform application of a simple local rule fitting within the theory of cellular automata. We propose a short-term solution, the shearing-rule which protects a one-dimensional repetition code against errors with good performance but lacks a threshold. This shortcoming is addressed by adding small local classical memories, used to define the signal-rules decoders, where error correction relies on exchanging binary signals. Combined with biased-noise qubits, such as cat qubits, these local decoders define a fully local one-dimensional quantum memory.
The final part of the manuscript addresses the advantage of quantum machines over their classical counterparts. We emphasize that any such demonstration must account for the presence of noise to be meaningful — for example, while exact sampling from the output distribution of circuits drawn from certain families is believed to be hard for classical computers, the task may become tractable if one allows for an approximate solution that reflects the presence of noise. In this context, we propose a constant-depth quantum circuit that solves a sampling problem in a fault-tolerant manner. Because our approach is particularly well-suited to the atom rearrangement capabilities of neutral-atom array platforms, we hope it will contribute to a future experimental demonstration of a fault-tolerant quantum advantage.

KEYWORDS

quantum information, fault-tolerant quantum computation, quantum error-correction, cellular automaton, local decoder, quantum advantage, quantum random circuits