Quantum message-passing algorithm for optimal and efficient decoding

Christophe Piveteau and Joseph M. Renes

Institute for Theoretical Physics, ETH Zürich, Switzerland

Recently, Renes proposed a quantum algorithm called belief propagation with quantum messages (BPQM) for decoding classical data encoded using a binary linear code with tree Tanner graph that is transmitted over a pure-state CQ channel [1], i.e., a channel with classical input and pure-state quantum output. The algorithm presents a genuine quantum counterpart to decoding based on the classical belief propagation algorithm, which has found wide success in classical coding theory when used in conjunction with LDPC or Turbo codes. More recently Rengaswamy *et al.* [2] observed that BPQM implements the optimal decoder on a small example code, in that it implements the optimal measurement that distinguishes the quantum output states for the set of input codewords with highest achievable probability. Here we significantly expand the understanding, formalism, and applicability of the BPQM algorithm with the following contributions. First, we prove analytically that BPQM realizes optimal decoding for any binary linear code with tree Tanner graph. We also provide the first formal description of the BPQM algorithm in full detail and without any ambiguity. In so doing, we identify a key flaw overlooked in the original algorithm and subsequent works which implies quantum circuit realizations will be exponentially large in the code dimension. Although BPQM passes quantum messages, other information required by the algorithm is processed globally. We remedy this problem by formulating a truly message-passing algorithm which approximates BPQM and has quantum circuit complexity $\mathcal{O}(\mathrm{poly}\, n, \mathrm{polylog}\, \frac{1}{\epsilon})$, where n is the code length and ϵ is the approximation error. Finally, we also propose a novel method for extending BPQM to factor graphs containing cycles by making use of approximate cloning. We show some promising numerical results that indicate that BPQM on factor graphs with cycles can significantly outperform the best possible classical decoder.

1 Introduction

Message-passing algorithms on graphical models have proven to be an integral computational tool for many fields such as statistics, information theory, machine learning, and statistical physics. *Belief propagation* (BP), which is perhaps the most well-known message-passing algorithm, approximates the marginal distributions of a given probabilistic graphical model. In the context of a spin model in statistical physics, for example, the single-site marginals yield the magnetization of the model. BP operates by sending messages across the edges of the graph in question. The nodes in the graph receive messages from their neighbors, perform a local computation, and then transmit the output to their neighbors. It has found tremendous success in the field of coding theory as a means of decoding data encoded into linear block codes and transmitted over noisy classical channels. For instance, certain low-density parity-check (LDPC) codes have been shown to approach the Shannon capacity of the channel when paired with BP decoding on the Tanner graph describing the parity-checks of the code [3]. The BP decoder works by estimating the bits of the codeword individually, which requires marginalization of the conditional probability distribution of the input codewords given the observed noisy channel output.

The task of decoding noisy channels with quantum output is quite different. Even for channels with classical input and quantum output, so-called CQ channels, there is generally no analog to

the conditional probability distribution of the inputs. Quantum information, famously, does not take specific values, and so there is no sensible notion of conditioning on quantum information in a probability distribution. Only if the output quantum states all commute can one find such a description. Instead, the decoding task is to implement a measurement of the quantum output which can reliably determine which codeword was input to the channel. The analogous situation for classical channels is to regard the decoding task as a procedure for distinguishing between the conditional distributions of the output given the input. But, due to the nature of classical information, it is not necessary to regard the decoding task in this manner since the opposite conditional distribution of the input given the output is available by Bayes' rule.

A simple approach for decoding CQ channels is to just measure the output quantum system for each transmitted codeword bit individually, which effectively transforms the CQ channel into a classical channel, and then proceed with classical decoding. Alas, this approach cannot generally achieve the capacity of the underlying CQ channel. This is even true for the simplest family of binary-input CQ channels, where the two possible output states are pure states. For instance, this channel arises when using Bosonic coherent states as signals over a pure-loss Bosonic channel. The maximal rate at which information can be transmitted over a CQ channel is given by the Holevo capacity χ, which in this case is simply $\chi = h_2(\frac{1}{2}(1-\cos\theta))$, where $\cos\theta$ is the fidelity (overlap) of the two pure state outputs and h_2 is the binary entropy function. Even using the optimal basis for measurement in the individual measurement strategy only results in a Shannon capacity of the effective classical channel of $1-h_2(\frac{1}{2}(1-\sin\theta))$, which is strictly smaller than χ for all $\theta \in (0, \frac{\pi}{2})$. Indeed the ratio of the latter to the former goes to zero as θ goes to zero; in the above example this implies that the individual measurement strategy will be especially suboptimal in the regime of small amplitude coherent states.

Recently, Renes proposed a decoding algorithm for just such pure state CQ channels which generalizes belief propagation decoding [1]. Like BP, it decodes the input codeword bitwise and operates by passing information, now quantum, among the nodes of cycle-free Tanner graphs. Unlike BP, though, it implements a measurement to distinguish between the quantum states associated to the two possible values of any chosen codeword bit. In the technical sense it is not strictly a belief propagation algorithm as it does not operate by marginalizing a probability distribution. But, like BP, it is an algorithm for inference. There are other quite distinct notions of belief propogation and other applications in the quantum setting, and we comment on these at the end of this section.

In fact, the algorithm of [1] performs the optimal distinguishing measurement, the so-called Helstrom measurement, which determines the value of any given codeword bit with optimal error probability. Like BP, this decoder is bitwise optimal on codes with tree Tanner graphs. In [2] Rengaswamy *et al.* proposed several simplifications of the algorithm and coined the name *belief propagation with quantum messages* (BPQM). Moreover, quite surprisingly, they also showed analytically that BPQM is blockwise optimal for a simple example code, i.e., it determines the entire input codeword with optimal error probability. This was unexpected, as bitwise optimality does not guarantee blockwise optimality, and indeed BP is provably not blockwise optimal for the classical binary symmetric channel (BSC) in all parameter regimes.

Both previous works on the subject of BPQM [1, 2] lack an explicit and formal description of the algorithm. Some details are only implicitly suggested or illustrated by example. A more precise description of the algorithm reveals that the original formulation is, contrary to what is claimed in the previous works, not truly a message-passing algorithm. While BPQM can be described as an algorithm acting on qubits which are passed along the edges of the Tanner graph, the operations performed by the nodes do not only depend solely on those qubits. Due to this problem, the resulting quantum circuit is not efficient as claimed, but in fact exhibits a runtime that scales exponentially in the number of encoded bits, k.

In this paper we address several of the limitations and open questions of BPQM raised in [1, 2]. We first list these in order of presentation before proceeding to sketch out the ideas and methods used for each.

- We give the first formally complete and concise description of BPQM. As part of that, we recognise a flaw in its original formulation which leads to a quantum circuit depth that grows exponentially in the code rank k (Section 3).

- We prove analytically that BPQM performs block-optimal decoding for any pure-state CQ

channel and any binary linear classical code with tree Tanner graph (Theorem 4.1 in Section 4).

- We provide a new algorithm, which we call *message-passing BPQM*. This new algorithm is truly a message-passing algorithm and does not exhibit the exponential cost present in BPQM. Message-passing BPQM can approximate BPQM to arbitrary precision. We provide analytical and numerical results characterizing the effect of the involved discretization errors (Section 5).
- We suggest a strategy for extending BPQM to linear codes with non-tree Tanner graphs by making use of approximate quantum cloning. We numerically investigate this approach and demonstrate that it achieves near-optimal decoding performance in a simple example (Section 6).

Further, we provide the source code to reproduce all numerical results and figures included in this work. [1]

1.1 Description of BPQM

Instead of working directly on the Tanner graph, BPQM utilizes what we call a *message-passing graph* (MPG), a binary tree derived from the Forney-style representation of the factor graph of the code. BPQM sequentially decodes one codeword bit at a time, and a separate MPG is required for each individual codeword bit. As an example, consider the 5-bit code with two parity-checks $X_1 + X_2 + X_4 = 0$ and $X_1 + X_3 + X_5 = 0$ (also considered in [2]). Its Tanner graph is depicted in Fig. 1a and the MPG associated to the bit X_1 is shown in Fig. 1b. Qubits are passed along the edges of the MPG, starting from the leaves and going towards the root of the graph. The qubits sent from the leaves are the output qubits obtained from the CQ channel. Every subsequent node receives two qubits and performs some processing in order to generate a single-qubit output which is then passed toward the root. There are two types of nodes, equality nodes denoted by '=' and check nodes denoted by '+', and the performed node operation depends on the corresponding type of the node. Ultimately a single qubit is produced by the node operation at the root, which is then projectively measured to obtain the codeword bit estimate.

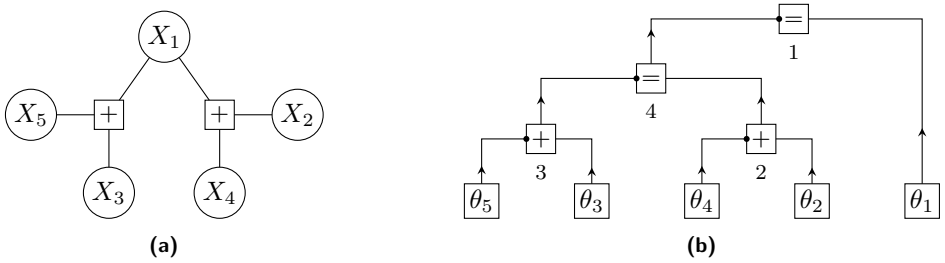


Figure 1: Tanner graph and message-passing graph (MPG) for a five-bit code. The MPG is centered on X_1.

However, the equality node operations must be uniformly controlled[2] on ancilla qubits which are generated by the node operations of all the preceding check nodes in the tree. That is, a different action must be taken on the qubits at the equality node for each of the different values of the ancilla qubits. These ancillas accumulate and their total number scales with the number of encoded bits k. Provided, as usual, that the universal set of gates supported by the quantum computer only contains gates acting on a bounded number of qubits, such uniformly-controlled gates generally need to be decomposed in a number of basic gates that scales exponentially in k. Previous works on BPQM do not explicitly address how to deal with this issue.

[1] The source code is available at https://github.com/ChriPiv/quantum-message-passing-paper.

[2] A gate acting on the system A and controlled by the m-qubit system B is said to be a *uniformly controlled gate* if the corresponding unitary acting on the joinst system AB is of the form $\sum_{i=0}^{2^m-1} U_A^{(i)} \otimes |i\rangle\langle i|_B$ where $U_A^{(0)}, \ldots, U_A^{(2^m-1)}$ is a collection of unitaries acting on A and $|i\rangle$ denotes a computational basis state on B.

To illustrate why BPQM achieves optimal bit-wise decoding (as also described in [1, 2]), we employ a formalism where classical and quantum states and channels can be represented by a variant of Forney factor graphs. This formalism is based on [4, 5] and is introduced in detail in Section 2.3. The procedure to decode the r-th codeword bit X_r can be understood by considering the factor graph representation of the CQ channel from X_r to the channel outputs $Y_1, \ldots, Y_n$ obtained from passing the codeword bits $X_1, \ldots, X_n$ through the CQ channel $W[\theta]$, for a random choice of the codeword. The operations performed in the BPQM algorithm can be modeled as adding additional nodes to the factor graph, and the operations in BPQM are carefully chosen so that the graph simplifies two qubits at a time until the codeword bit value is encoded in just a single qubit. This qubit can then be projectively measured in the appropriate basis to determine the codeword bit value.

On its face, this description seems to differ strongly from the classical BP algorithm, where marginalization is performed by merging the nodes of the graph appropriately, sometimes referred to as "closing the boxes" in Forney-style factor graph language [6]. We show in Appendix C that BP can also be formulated in a channel contraction formalism similar to BPQM, which makes clear that BPQM is a quantum analog of BP.

1.2 Block optimality

In order to decode the complete codeword, the single-codeword bit procedure is executed sequentially for k independent codeword bits. Ideally, each of these k procedures would have access to the original n channel output qubits. But since quantum information cannot be cloned, one must attempt to retrieve the original n-qubit state after some codeword bit has been decoded, to allow for the decoding for the subsequent codeword bits. This is nearly possible: The node operations involved in BPQM are all unitary, and can therefore be reversed. The only non-unitary part of the algorithm is the single-qubit projective measurement which produces the estimate for the codeword bit. Thus, for each codeword bit, the POVM elements associated to the two possible bit values are orthogonal projection operators.[3]

A central ingredient for the proof of optimality of the complete codeword decoder is the fact that an optimal decoder is realized by the so-called *pretty-good measurement* (PGM) [7, 8], sometimes also called square-root measurement (also pointed out in [2, 9]). This ensures the optimal measurement is a rank-one projective measurement, due to the linear structure of the code and earlier results on the optimality and orthogonality of the PGM [10–12]. Orthogonality of the optimal measurement then implies that optimal decoding can in principle be performed bitwise. Simply construct projection operators $\Lambda_{r,j}$ for the rth codeword bit to have the value j by summing the associated codeword projections $\Pi_{\vec{x}}$ over codewords $\vec{x}$ such that $x_r = j$. The intersection of the bitwise projection operators thus determines the codeword.

The proof is therefore reduced to demonstrating that BPQM realizes $\Lambda_{r,j}$. To show this, a precise analysis is required that characterizes how the BPQM unitary acts on all possible pure state inputs. The result of this analysis is captured by Lemma 4.4. It can be considered a generalization of the factor graph contraction argument used to demonstrate bitwise optimality, as the latter result can be recovered by simply averaging out the corresponding codeword bits. As a consequence of the structure of the projections, it follows that the (optimal) decoding performance of BPQM does not depend on how the k independent codeword bits are chosen nor in which order these k codeword bits are decoded.

1.3 Message-passing BPQM

As previously mentioned, BPQM is not truly a message-passing algorithm. Consider the qubit that is passed over some edge e of the MPG towards the root. The state of this qubit is a superposition of states $|Q(z,\varphi)\rangle := \cos(\varphi/2)\,|0\rangle + (-1)^z \sin(\varphi/2)\,|1\rangle, z \in \{0,1\}, \varphi \in (0,\pi)$ for different values of φ (but fixed z). The two states corresponding to the two possible values of z are related by a Pauli-Z operation on the qubit. The angle φ takes values in some set $\{\varphi_{\vec{j}}\}_{\vec{j}\in\{0,1\}^{k_e-1}}$ where $k_e - 1$ is the number of check nodes that precede e in the graph. The information specifying the angle

[3] Here orthogonality refers to the fact that the POVM elements E_i fulfill $E_i E_j = \delta_{ij} E_i$.

is stored in the ancilla qubits produced by all the check nodes that precede e in the MPG. This information is classical, as the states of the ancilla qubits corresponding to the different angles all commute. Equality node operations depend on the angle of the two incoming qubits, and for this reason they must be conditioned on the $k_e - 1$ ancilla qubits produced by the preceeding check nodes. This is in turn the cause for the non-message-passing nature of BPQM that exhibits an exponential quantum circuit complexity.

We address this problem by introducing a quantum register that directly stores the angle of the qubits passed over the edge e. The messages passed around the MPG now consist of a qubit as well as the angle register. This on-the-fly bookkeeping of the involved angles is the essence of message-passing BPQM. Were these new quantum registers able to represent all angles $\varphi \in (0, \pi)$ deterministically, they would need to be infinite-dimensional quantum systems. To overcome this issue, we instead store the angle in a discretized representation using only B qubits. In principle, the angle register could also be chosen to be a classical register, but in that case it is impossible to revert the action of BPQM at a later point as is required to decode multiple bits. Therefore, the quantum nature of the angle register is imperative for decoding the entire codeword.

We provide an in-depth analysis of how the discretization errors propagate throughout the execution of the algorithm and how much they reduce the probability of successful decoding. The understanding of this error propagation is critical as it fundamentally differs from its classical counterpart: If BPQM operates under the assumption that a qubit involved in a node operation has the angle $\tilde{\varphi}$ which differs from the true value φ, then the output of the node operation will generally no longer be in a state that exhibits the aforementioned Pauli-Z symmetry. The technical difficulty of analyzing the error is that the functions which need to be computed are not uniformly continuous. The central result is given by Theorem 5.2, which states an explicit upper bound on the error of the decoder. As a direct consequence, one has to choose $B = \mathcal{O}(n, \log \frac{1}{\epsilon})$ in order to achieve a desired decoding error ϵ. Consequently, the quantum circuit width and depth of message-passing BPQM are shown to scale as $\mathcal{O}(\operatorname{poly} n, \operatorname{polylog} \frac{1}{\epsilon})$. We also provide some numerical results on how the discretization errors effect the decoding error when decoding a single codeword bit of some 17-bit binary linear code. As expected, we observe in Fig. 15b that the decoding error decreases exponentially in B.

If one takes the value of B to be fixed, which corresponds to a model of computation in which classical scalar operations take constant time, then the circuit depth of message-passing BPQM scales as $\mathcal{O}(kn)$ where k is the number of encoded bits. However, if the topology of the MPG allows us to parallelize some of the node operations, this complexity can in some cases even be reduced to quasilinear time $\mathcal{O}(k \log n)$.

1.4 Codes with cycles

Finally, we also study the question of how the BPQM decoding algorithm can be extended to codes with cycles in their Tanner graphs. As a guiding principle, we consider how classical BP is heuristically extended from tree to non-tree factor graphs by defining a fixed message-passing schedule at each node [13]. In general, classical BP on non-tree factor graphs is not guaranteed to produce the correct marginals, let alone converge. Nevertheless, it has empirically been observed to give good approximate results in the case where the cycles are large. Similarly, when applying BPQM to non-tree graphs, one should not expect to achieve optimal decoding.

Fixing the message-passing schedule for non-tree graphs can be thought of as transforming the original decoding problem, including the actually observed output $\vec{y}$ as well as the descriptions of both the code $\mathcal{C}$ and the channel, to a new decoding problem involving a new output $\vec{y}'$, a tree code $\mathcal{C}'$, and a new channel model. The code $\mathcal{C}'$ in question is obtained by "unrolling" the Tanner graph of $\mathcal{C}$ around the variable X_r for a certain number of steps, giving each variable node a unique name. The obtained graph is sometimes also referred to as *computation tree*. For example, consider decoding X_1 of the (6,3) binary linear code with Tanner graph depicted in Fig. 2a. By unrolling the Tanner graph around X_1 for $h = 2$ steps, we obtain a Tanner graph defining the 9-bit code $\mathcal{C}'$, shown in Fig. 2b. A codeword $\vec{X}$ of $\mathcal{C}$ specified by $X_1, \ldots, X_6$ can be transformed to a codeword $\vec{X}'$ of $\mathcal{C}'$ by taking $X'_j = X_j$ for $j = 1, \ldots, 6$, and $X'_7 = X_4$, $X'_8 = X_3$, and $X'_9 = X_6$. This is depicted in Fig. 2c. The same duplication procedure can be applied to the channel output $\vec{y}$, and the channel parameters for duplicated bits can be taken to be the same as their original values.

Then, BP decoding of the resulting $\vec{y}'$ using the duplicated channel model will give an estimate of the original input X_1. One can check that running classical BP on $\mathcal{C}$ to decode X_r given a noisy channel output $\vec{y} \in \mathbb{F}_2^n$ for h time steps is equivalent to running classical BP on $\mathcal{C}'$ given the noisy channel output $\vec{y}'$.

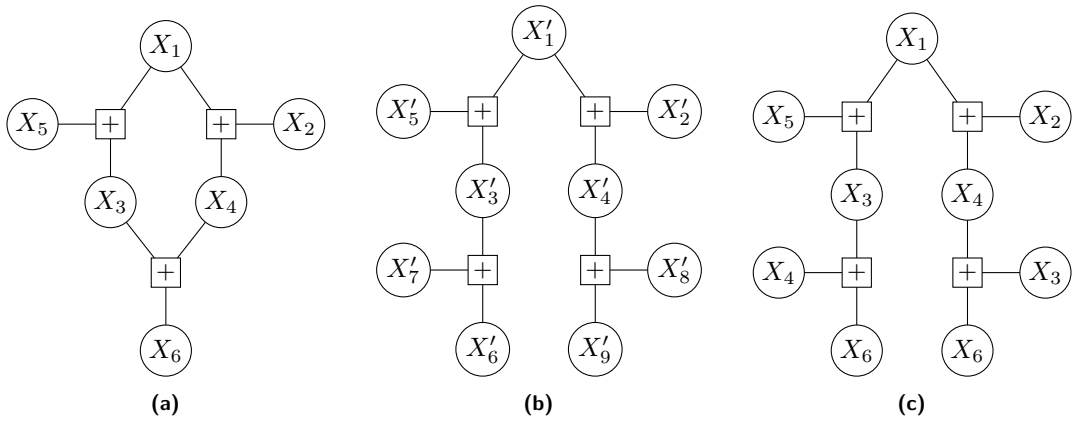


Figure 2: Unrolling the Tanner graph of a code $\mathcal{C}$ to generate a new code $\mathcal{C}'$ and computation tree of $\mathcal{C}$. (a) The Tanner graph of a 6-bit code $\mathcal{C}$. (b) Tanner graph of the 9-bit code $\mathcal{C}'$ defined by unrolling the original Tanner graph around X_1 to depth 2. (c) $h = 2$ computation tree of $\mathcal{C}$, obtained by identifying the variables of $\mathcal{C}'$ with those of $\mathcal{C}$.

Returning to the case of a CQ channel, we encounter a major problem if we try to apply the same procedure: It is impossible to copy the channel output qubits due to the no-cloning theorem. We address this problem by employing approximate cloning, which generates imperfect clones of the channel output qubits. Choosing the channel parameters of the duplicated outputs in accordance with the cloning procedure, numerical results for an 8-bit code in Section 6.1 show that the resulting decoder can significantly outperform the best possible classical decoder[4] using this strategy. In fact, it nearly reaches optimal decoding performance.

Note that due to the approximate cloning issue, BPQM decoding on non-tree graphs exhibits a strongly different characteristic to its classical counterpart: Increasing the number of unrolling steps h does not always improve the result. After some point, decoder performance deteriorates, as a larger value of h implies that more approximate cloning operations must be performed and therefore the quality of the involved qubits decreases.

We note that the described unrolling strategy is very unlikely to be the best choice to generate $\mathcal{C}'$ for any possible code $\mathcal{C}$, especially for codes with many small cycles, where the number of approximate cloning operations would increase very quickly with the depth h of the unrolling. Most likely, different families of codes will require different strategies to generate the best possible choice of $\mathcal{C}'$. Similarly, which kind of codes lend themselves best to BPQM decoding is an interesting question for future research. LDPC codes are a promising family of codes, as they have proven to exhibit excellent performance for transmission over classical channels and their Tanner graphs exhibit few and large cycles.

1.5 Other notions of quantum belief propagation

The notion of belief propagation has been applied in the quantum setting in many different previous works; here we comment on their relation to the present work. Most closely related is the use of belief propagation for decoding quantum information subject to Pauli errors, as studied by Poulin in [14, 15], as well as many others since [16–23]. Here, however, classical BP is sufficient: The task is to infer which error occured from the classical syndrome information, which only involves the classical conditional probability of the former given the latter. Conversely, [24] proposes decoding LDPC codes over a classical channel by means of quantum annealing. This does not involve

[4]Here the best classical decoding refers to measuring the channel outputs in the $|\pm\rangle$ basis and doing classical maximum likelihood decoding.

decoding information from quantum systems, but instead using quantum systems to perform a classical computation. Slightly further afield is the work of Leifer and Poulin on developing quantum graphical models and belief propagation algorithms for marginalization of density operators [25]. This issue does not immediately arise here, as essentially only the leaves of Forney factor graphs we consider involve quantum systems. Finally, BP is also related to the Bethe-Peirels approximation of the free energy in statistical mechanics [26, 27]. In this approximation one computes the free energy from the probability distribution using only the marginals describing single sites and neighboring sites. As first observed by Yedidia, Freeman, and Weiss, the fixed points of BP are stationary points of the approximation of the free energy [28–30]. Hastings and Poulin have constructed quantum belief propagation algorithms for approximating the free energy in quantum statistical mechanics [31, 32]. Further variations of this approach were investigated by Cao and Vontobel [5, 33].

2 Preliminaries

2.1 Factor graphs and classical belief propagation decoding

Factor graphs are a graphical representation of the factorization of a function of several variables, e.g., a joint probability distribution, into a product of separate terms each involving only a subset of the variables. The standard variant of factor graphs [34] uses a bipartite graph consisting of two types of nodes: variable and factor nodes. Variable nodes are usually represented with circles and factor nodes with rectangular boxes. Consider a factorizable function

$$f(x_1, \ldots, x_n) = \prod_{j \in J} f_j(Z_j) \tag{1}$$

where J is some finite index set and the f_j are functions having some subset Z_j of $\{x_1, \ldots, x_n\}$ as arguments. The corresponding factor graph consists of n variable nodes $X_1, \ldots, X_n$ as well as a factor node f_j for each $j \in J$. A variable node X_i and a factor node f_j are connected by an edge if and only if $x_i \in Z_j$.

Consider an (n, k) binary linear code $\mathcal{C}$, i.e., a linear subspace of $\mathbb{F}_2^n$ of dimension k. A parity-check matrix H of this code directly tells us how to represent the membership indicator function

$$I_{\mathcal{C}} : \mathbb{F}_2^n \to \{0, 1\}\,,\; I_{\mathcal{C}}(x) := \begin{cases} 1 \text{ if } x \in \mathcal{C} \\ 0 \text{ else} \end{cases} \tag{2}$$

in terms of such a factor graph. The parity-checks manifest as check factor nodes (denoted by '+') in the factor graph, which represent the function

$$x_1, \ldots, x_m \mapsto \begin{cases} 1 \text{ if } x_1 + \cdots + x_m \equiv 0 \pmod 2 \\ 0 \text{ else} \end{cases} \tag{3}$$

where m is the number of edges connected to the check node. The resulting factor graph is commonly called a Tanner graph of the code. Generally a Tanner graph is not unique, as there exist different parity-check matrices H associated with the code $\mathcal{C}$. Consider again the $(5, 3)$ code with Tanner graph in Fig. 1a. It has the following parity-check and generator matrices:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{4}$$

Throughout the majority of this work, it will prove advantageous to work with a different graphical representation of factorizable functions, namely the so-called *Forney-style factor graphs* [6, 35]. Here variables are represented by the edges of the graph whereas the factor terms of the function are represented by the nodes of the graph. Similarly, an edge and a node of the graph are connected if and only if the corresponding variable is part of the corresponding factor term. The Forney-style representation of the Tanner graph of the 5-bit code is depicted in Fig. 3a. When translating a

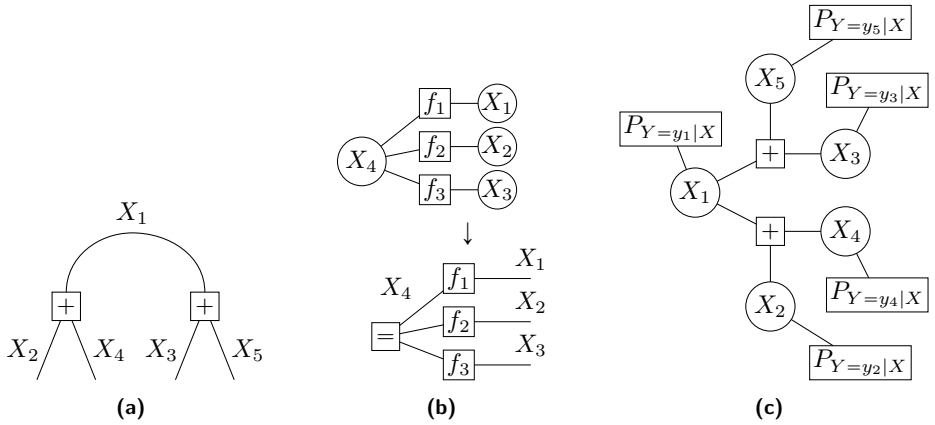


Figure 3: (a) Forney-style representation of the Tanner graph of the 5-bit code. (b) The graph on top is an example of a (standard) factor graph containing a variable X_4 that is connected to more than two factor nodes. The corresponding Forney-style representation of the factor graph, depicted on the bottom, correspondingly contains an equality node. (c) (Standard) factor graph characterizing the joint distribution of the codeword bits of the 5-bit code given that some channel output $\vec{y}$ was observed.

factor graph to the Forney-style representation, one might encounter the issue that some variables are involved in more than two factors, as depicted in the example in Fig. 3b. In that case, equality factor nodes must be introduced in the Forney-style factor graph. Equality nodes are depicted by the symbol = and represent the function that returns 1 if all connected variables have the same value and 0 otherwise. Thus, the Forney-style representation of the Tanner graph of any binary linear code contains only check nodes and equality nodes.

Given a multivariate function $f(x_1, \ldots, x_n)$, consider the task of finding the marginal w.r.t. the variable x_i, i.e., the function

$$f_i(x_i) := \sum_{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n} f(x_1, \ldots, x_n)\,. \tag{5}$$

In general this task is very expensive as the runtime of the brute force computation scales exponentially in n. However, if we are given some additional information about the factorizability of f in form of a factor graph, then the belief propagation (BP) algorithm can utilize this additional structure to approximate the marginal at a reduced complexity. BP is a message-passing algorithm, i.e., it can be realized as a sequence of messages being passed between nodes of the factor graphs.[5] Each node receives some input messages from its neighbors, performs a certain computation given that input, and produces output messages that it passes to its neighbors. Under the assumption that the factor graph is a tree, BP is able to compute the exact marginals, otherwise it will generally only return an approximate result. For a more complete description of factor graphs and classical belief propagation, we refer the reader to [6, 13].

BP can be directly applied to the task of communicating classical information over a noisy classical channel described by the transition probability $P_{Y|X}$ which maps some bit $x \in \mathbb{F}_2$ into some output alphabet $\mathcal{Y}$. Reliable communication is achieved by coding our message using some (n, k) binary linear code. Assume that the transmitted codeword is chosen uniformly randomly. If we observe the output of the channel to be some vector $\vec{y} \in \mathcal{Y}^n$, that new knowledge about the joint probability distribution of the codeword bits is reflected by adding new factor nodes to the codeword bits in the factor graph, as depicted in Fig. 3c. Marginalizing this new joint probability distribution with respect to the i-th codeword bit X_i using BP gives the conditional probability density $\mathbb{P}_{X_i|\vec{Y}=\vec{y}}$. From this we can easily compute the bitwise maximum a-posteriori (bit-MAP) estimate for the i-th bit, defined by $\arg\max_{x\in\{0,1\}} \mathbb{P}_{X_i|\vec{Y}=\vec{y}}(x)$.

[5] Belief propagation can be formulated either for the standard factor graphs or for Forney-style factor graphs.

2.2 Commmunication over a pure state classical-quantum channel

We consider the task of achieving reliable communication of classical information over a CQ channel, i.e., a channel with classical input and quantum output. We restrict ourselves to the study of pure-state CQ channels $W[\theta]$ that produce qubit states of the form

$$\{0,1\} \ni x \mapsto |Q(x,\theta)\rangle := \cos\frac{\theta}{2}|0\rangle + (-1)^x \sin\frac{\theta}{2}|1\rangle \tag{6}$$

for some angle $\theta \in (0,\pi)$. Note that any pure-state CQ channel taking a single bit as input can be brought into this form by performing a corresponding unitary transformation on the output of the channel, since the two possible pure-state outcomes span a two-dimensional subspace of the total Hilbert space.

To achieve reliable communication of classical information over this channel, we can redundantly encode our information using a binary linear code, completely analogous to the procedure for transmitting classical information over a noisy classical channel. The only difference is that the channel outputs $Y_1,\dots,Y_n$ now denote two-dimensional quantum systems (i.e. qubits) instead of classical random variables. The question now arises, how one can decode the original codeword given access to this quantum system.

A first naive approach is to convert the CQ channel to a classical channel, by estimating for each channel output separately if it is more likely in the state $|Q(0,\theta)\rangle$ or $|Q(1,\theta)\rangle$. This can be done optimally by the Helstrom measurement [36], which in this case corresponds to measuring the corresponding qubit in the $|\pm\rangle$ basis. The resulting effective classical channel is a binary symmetric channel with error probability $p = \frac{1-\sin\theta}{2}$ and capacity $C = 1 - h(p)$. We refer to this decoding strategy as a *classical decoder*, since we can make use of any classical decoding algorithm and we do not make use of the fact that the channel output is a quantum system.

However, a classical decoder cannot realize the best possible decoding performance. This is illustrated by the celebrated Holevo-Schumacher-Westmoreland (HSW) theorem which states that the maximal possible rate of classical information that can be reliably transmitted is given by the Holevo capacity of the channel, which for $W[\theta]$ turns out to be $\chi = h_2(\frac{1+\cos\theta}{2})$ where h_2 is the binary entropy function. It can be easily checked that χ can be significantly larger than C, and in fact one find that $\chi/C \to \infty$ in the limit $\theta \to 0$. To approach the Holevo capacity, it is therefore imperative to design truly quantum decoders which explicitly make use of the quantum nature of the channel output. See Guha [37] for more on this point.

When the Tanner graph of the considered code is a tree, classical BP can achieve bitwise optimal decoding (i.e. bit-MAP decoding) of information transmitted over a classical channel. Message passing on such tree graphs is the foundation of belief propagation in general, though codes with cycle-free Tanner graphs are known to perform badly, since they exhibit a distance $d \leqslant 2$ as soon as the rate k/n is greater or equal one half [38]. Interestingly, when decoding codes with cycle-free Tanner graphs transmitted over pure-state CQ channels with BPQM, one does not only attain bitwise optimal decoding, but even the block optimal decoding.

2.3 Representing states and channels using Forney-style factor graphs

This section explains how we use the Forney-style factor graph formalism to represents classical and quantum states and channels throughout this work. The introduced formalism uses the language of Forney-style factor graphs, but at its heart it is very closely related to the tensor network formalism (for an introduction, see [39]). The main advantage is that the formalism naturally represents states and operations that involve both classical and quantum systems. The concepts introduced in this section are necessary to understand the central ideas in the proof of bit optimality of BPQM in Section 3.2, but they have no further importance for the rest of this manuscript.

For the rest of this section we will always restrict ourselves to (random) variables over finite alphabets as well as finite-dimensional quantum systems. Consider a Forney-style factor graph representing the factorization of some function in $m+l$ variables $f(V_1,\dots,V_m,H_1,\dots,H_l)$ where one subset of the variables $V_1,\dots,V_m$, called the 'visible' variables, are represented by half-edges, i.e., they are each connected to exactly one factor node. The other variables $H_1,\dots,H_l$ are 'hidden' inside the graph, i.e., they are each connected to exactly two factor nodes. We define the *exterior*

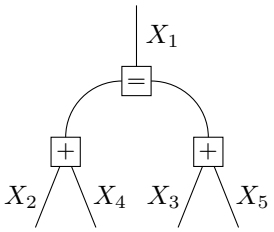


Figure 4: Factor graph representing the distribution of a uniformly random codeword of the 5-bit code.

function of the factor graph to be the joint marginal of the visible variables, i.e.,

$$v_1, \ldots, v_m \mapsto \sum_{h_1,\ldots,h_l} f(v_1, \ldots, v_m, h_1, \ldots, h_l)\,. \tag{7}$$

We define the *partition function* of the factor graph to be the sum over all variables, i.e.,

$$\sum_{v_1,\ldots,v_m,h_1,\ldots,h_l} f(v_1, \ldots, v_m, h_1, \ldots, h_l)\,. \tag{8}$$

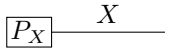
Consider some rank-k tensor[6] $T_{i_1,\ldots,i_k}$ where the indices i_j take values in the alphabets $\mathcal{I}_j$. We say that a factor graph G represents the tensor T if G has k visible variables $I_1, \ldots, I_k$ defined on the alphabets $\mathcal{I}_1, \ldots, \mathcal{I}_k$ and the exterior function of G is exactly equal to

$$i_1, \ldots, i_k \mapsto T_{i_1,\ldots,i_k}\,. \tag{9}$$

The contraction of two tensors can be represented by the factor graph obtained by connecting half-edges of the factor graphs representing the two original tensors. Consider for instance two factor graphs G and G' representing the tensors $T_{i_1,\ldots,i_k}$ and $T'_{j_1,\ldots,j_{k'}}$ where the indices i_l and j_l run over the same alphabets for $l = 1, \ldots, m$, for $m \leqslant \min\{k, k'\}$. Consider the factor graph $\tilde{G}$ defined by connecting the half-edges for I_l and J_l between G and G' for $l = 1, \ldots, m$. $\tilde{G}$ then represents the tensor $\tilde{T}$ obtained by contracting T and T':

$$\tilde{T}_{i_{m+1},\ldots,i_k,j_{m+1},\ldots,j_{k'}} = \sum_{k_1,\ldots,k_m} T_{k_1,\ldots,k_m,i_{m+1},\ldots,i_k} T'_{k_1,\ldots,k_m,j_{m+1},\ldots,j_{k'}}\,. \tag{10}$$

The central quantities and operations in probability theory can be expressed in terms of tensors and tensor contractions. For instance, a probability distribution P_X of some random variable X defined over the alphabet $\mathcal{X}$ can be simply thought of as a rank-1 tensor where the index takes values in $\mathcal{X}$. The following graph is a trivial example of a factor graph representing the distribution P_X:



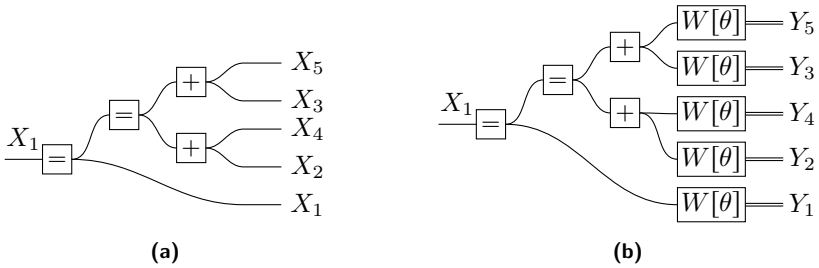
. (11)

As another example, consider the Forney-style representation of the Tanner graph of a binary linear (n, k) code where every variable is represented by a half-edge, as in Fig. 4 for the 5-bit code. Since the constructed factor graph corresponds to the membership indicator function, it represents the probability distribution of a uniformly randomly chosen codeword $(X_1, \ldots, X_n)$, up to a normalization constant (1/8 in the case of the 5-bit code).

In general, when we represent probability distributions (and also channels as we will see later) with factor graphs in this manuscript, we do not bother to properly normalize the tensors. This is done to simplify the notation. The correct normalization can always be inferred by context, e.g., the entries of a tensor representing a probability distribution should always sum up to one.

Channels are linear maps that are uniquely represented by the matrix containing all the transition probabilities, and therefore they can also thought of as a tensor. Moreover, the application

[6]Note that we use the term *tensor* in the sense of a multidimensional array (as is common in machine learning or tensor network theory) and not of a multilinear map. The rank of a tensor is its number of indices, e.g., a rank-1 tensor is a vector and a rank-2 tensor is a matrix.

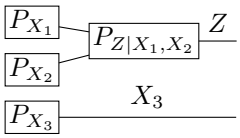


(a) **(b)**

Figure 5: (a) Factor graph describing the classical channel from X_1 to X_1, X_2, X_3, X_4, X_5 for the 5-bit code. (b) Factor graph describing the CQ channel from X_1 to the CQ channel outputs Y_1, Y_2, Y_3, Y_4, Y_5 for the 5-bit code.

of a channel on some state consists of a contraction of the two involved tensors which can accordingly be represented using our factor graphs. Note that the tensors corresponding to probability distributions and channels contain only non-negative entries. Correspondingly, the contraction of two tensors with non-negative entries is itself again a tensor with non-negative entries.

Consider as an example three random variables X_1, X_2, X_3 distributed as $P_{X_1}, P_{X_2}, P_{X_3}$ as well as a channel $P_{Z|X_1,X_2}$ that maps X_1, X_2 to a random variable denoted by Z. The joint state of X_3 and Z is represented by the following factor graph:



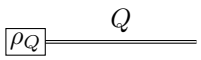
. (12)

For further illustration, we provide a non-trivial example of a factor graph representation of a channel that will play an important role later in this work. Consider the Forney-style representation of the Tanner graph of some (n, k) binary linear code, such that the half-edges consist exactly of the codeword bits $X_i, i = 1, \ldots, n$, and one of the codeword bits X_r for $r \in \{1, \ldots, n\}$ has two half-edges (which can be achieved by the use of an equality node). Figure 5a depicts this factor graph for the 5-bit code and $r = 1$. This factor graph represents the channel from X_r to $X_1, \ldots, X_n$, i.e., the channel that given some fixed value $X_r = x_r$ generates one of the 2^{k-1} possible codewords $\vec{z} \in \mathcal{C}$ with $z_r = x_r$ uniformly randomly.[7] Here again, the tensor represented by the factor graph differs from the channel by a normalization constant.

Forney-style factor graphs have also been generalized to describe quantum mechanical systems [4, 5]. Here the central quantities of interest are mixed states (density matrices) and quantum channels, which are again tensors and thus representable with our factor graph formalism. Consider for instance a density matrix ρ on some joint quantum system $Q_1 Q_2 \ldots Q_m$ described by the Hilbert space $\mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_m$ where $d_i := \dim(H_i)$. One possibility to represent ρ with a tensor works as follows: We identify ρ with a rank-$2m$ tensor T defined as

$$T_{i_1^+, i_1^-, \ldots, i_m^+, i_m^-} = \left(\langle i_1^+|_1 \otimes \cdots \otimes \langle i_m^+|_m\right) \rho \left(|i_1^-\rangle_1 \otimes \cdots \otimes |i_m^-\rangle_m\right) , \tag{13}$$

where for all $j = 1, \ldots, m$ we have implicitly made some choice of basis $\{|0\rangle_j, \ldots, |d_j - 1\rangle_j\}$ of $\mathcal{H}_j$. Notice that the indices of T always appear in pairs I_j^+, I_j^- that run over the same alphabet. Following the notation introduced by [5], we depict both random variables with a single double-edged wire that we simply denote with the symbol I_j. For instance, the state ρ_Q of a quantum system Q can be trivially described by the factor graph



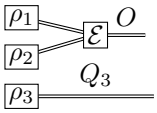
. (14)

Since quantum channels are linear maps on the space of endomorphisms of the involved Hilbert spaces, they can also be represented by tensors. Fully analogous to the case of classical channels,

[7] Note that here we implicitly assume that there exists some codeword $z \in \mathcal{C}$ where $z_r \neq 0$.

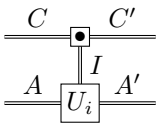
an application of a quantum channel on some quantum state can be represented by a contraction of the two corresponding factor graphs. In fact, by correctly identifying half-edges of a factor graph describing some channel, one can verify that the tensor described by the factor graph is the Choi matrix of the involved channel. Therefore the fundamental quantities of quantum information theory (density matrices and Choi matrices) are positive semi-definite operators, in contrast to the non-negative tensors in classical probability theory. The contraction of two positive semi-definite operators can again be shown to be itself a positive semi-definite operator.

In analogy to the example in the classical case presented previously, consider three quantum systems Q_1, Q_2, Q_3 in the separable state $\rho_1 \otimes \rho_2 \otimes \rho_3$. A quantum channel $\mathcal{E}$ takes the systems Q_1, Q_2 as input and produces an output described by the quantum system O. The joint state of $Q_3 O$ is then described by the factor graph



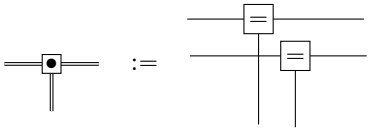
. (15)

We illustrate one special family of a quantum channels that will be encountered later in this work: uniformly-controlled gates, which apply the unitary $\sum_{i=0,\ldots,2^m-1} |i\rangle\langle i|_C \otimes (U_i)_A$ on the m-qubit control system C and the target system A, where the $U_i, i = 0, \ldots, 2^m - 1$, are some unitaries on the system A. The Following factor graph can be shown to describe the corresponding channel



, (16)

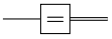
where I is a m-qubit system and the factor node $\bullet$ stands for an equality node for each separate edge, i.e.,



, (17)

and U_i is the quantum channel corresponding to the isometry $\sum_i \langle i|_I \otimes (U_i)_A$ acting on the system IA. Note that the subscript i in the factor node U_i is chosen to indicate which system (here I) acts as the control for choosing which unitary is applied. Moreoever, for unitaries we slightly abuse the convention of labels of nodes corresponding to channels and simply label the node by the (name of the) unitary, say U, rather than the corresponding channel $\mathcal{U} : \rho \to U\rho U^\dagger$.

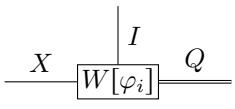
Finally, this factor graph formalism can be straightfowardly generalized to also describe CQ and QC channels, following the same procedure to describe classical and quantum objects introduced above. In this case, the corresponding factor nodes will have a mix of single-edged and double-edged wires. For instance, the factor node



(18)

represents the CQ channel which embeds a bit in the computational basis (i.e. $0 \mapsto |0\rangle\langle 0|$ and $1 \mapsto |1\rangle\langle 1|$), or when read from the other side as the QC channel corresponding to non-selectively measuring a qubit. Similarly, the CQ channel $W[\theta]$ can be represented by a corresponding factor node with a single-edged and double-edged wire. For example, Fig. 5b depicts the channel from the codeword bit X_1 to the CQ channel outputs $Y_1, \ldots, Y_5$ for the 5-bit code, i.e., the channel that given some value $X_1 = x_1$ generates a uniformly random codeword $\vec{z} \in \mathcal{C}$ such that $z_1 = x_1$ and then passes that codeword $\vec{z}$ through five instances of the channel $W[\theta]$.

To end this section, we introduce a generalization of the channel $W[\theta]$ that will play an important role in later sections. We consider the CQ channel, that takes some bit $x \in \{0, 1\}$ as well as a bit string $i \in \{0, 1\}^m$ as input for some $m \geqslant 0$. The channel output is a qubit Q prepared in the state $|Q(x, \varphi_i)\rangle$ where the angle is chosen from some pre-determined list of angles $\varphi_0, \ldots, \varphi_{2^m-1}$ according to the value of i. Visually, we will depict this channel with following factor graph node:



. (19)

Note that we can recover the original channel $W[\theta]$ by choosing $m = 0$. The subscript i in the channel node $W[\varphi_i]$ is used to indicate that the system I controls the chosen angle.

3 Description of BPQM

Consider a (n, k) binary linear code $\mathcal{C}$ that exhibits a Tanner graph that is a tree. A codeword $(X_1, \ldots, X_n)$ is chosen uniformly randomly from $\mathcal{C}$. For $i = 1, \ldots, n$, the ith codeword bit X_i is passed through the CQ channel $W[\theta_i]$ for some $\theta_i \in (0, \pi)$ and the output qubit of the channel is denoted by Y_i. BPQM is an algorithm that takes as input the qubits $Y_1, \ldots, Y_n$ as well as the channel parameters $\theta_1, \ldots, \theta_n$, and its objective is to correctly guess the original codeword $(X_1, \ldots, X_n)$.

To decode the full codeword $(X_1, \ldots, X_n)$, it suffices to decode k independent codeword bits from which the value of the remaining codeword bits can be uniquely determined. By relabelling, we can assume that $X_1, X_2, \ldots, X_k$ form such an independent set. The BPQM decoder of [1] consists of k sequentional single-bit decoding operations for the codeword bits $X_j, j = 1, \ldots, k$, each specified by a unitary V_j acting on the qubits $Y_1, \ldots, Y_n$ followed by a single-qubit measurement in the $|\pm\rangle$ basis. The measurement outcome is the estimate of the corresponding codeword bit. After the bit X_j has been decoded, the decoding operations are rewound by applying $V_j^\dagger$. For example, the quantum circuit realizing the codeword decoding for the previously introduced 5-bit code is depicted in Fig. 6.

Section 3.1 is concerned with describing an algorithm that generates a sequence of operations on $Y_1, \ldots, Y_n$, which can be represented using a quantum circuit, that realize the single-bit decoding unitary V_j. That is, the single-bit BPQM decoding algorithm consists of two parts, the *quantum circuit* realization of V_j and the *compiler* which generates the quantum circuit description. Many steps involved in the single-bit decoding of BPQM might seem somewhat arbitrary at first glance. In section Section 3.2, the action of the single bit decoding procedure will be analyzed using the factor graph formalism from Section 2.3, and by doing so it becomes naturally clear why optimal bit decoding is realized.

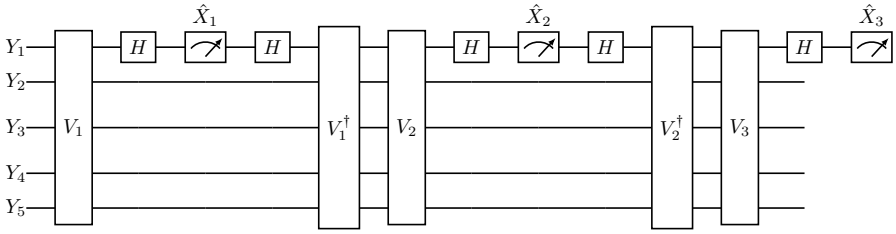


Figure 6: Quantum circuit for decoding the complete codeword for the 5-bit code. The input of the circuit are the 5 channel output qubits $Y_1, \ldots, Y_5$. The classical output of this circuit are the estimates $\hat{X}_1, \hat{X}_2, \hat{X}_3$ of the bits X_1, X_2, X_3.

On a simple 5 bit code, it has been observed in [2] that the quantum circuits for decoding subsequent bits can be simplified by constructing them from an updated factor graph where the information of previously decoded codeword bits is incorporated. We do not address this optimization here and leave its analysis as an open question for further research.

3.1 Single-bit decoding

3.1.1 Message passing graph

The BPQM algorithm to decode a single bit X_r for some $r \in \{1, \ldots, k\}$ is most easily expressed in terms of what we call a *message-passing graph* (MPG) for the code $\mathcal{C}$ w.r.t. X_r. It is a binary tree associated to a codeword bit X_r which simplifies the bookkeeping of all the conditional operations required in the quantum circuit implementing the decoder. It can be constructed from the Tanner graph as follows. First, determine the Forney-style description of the Tanner graph of $\mathcal{C}$, adding

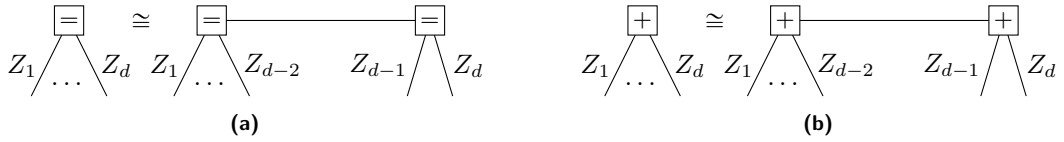


Figure 7: Decomposition of degree $d \geqslant 4$ equality and check nodes into multiple nodes of lower degree. The equality $\cong$ can be understood in terms of the exterior functions of the two factor graphs. By iteratively applying these two rules, one can obtain a graph where every equality and check has exactly degree three.

equality nodes as necessary so that to each codeword bit is associated with a half-edge and there is no other half-edge in the graph. The resulting graph consists of check and equality nodes of varying degrees. Using the decomposition rules depicted in Fig. 7, one adds additional nodes and edges to replace any node of degree larger than three by nodes of degree three. Observe that doing so does not alter the tree structure of the graph. Any node of degree two can be removed, as the two involved random variables are essentially equal. The resulting factor graph therefore only has nodes of degree three. Connect additional "channel" nodes to the half-edges, labelled by channel parameter θ_j for the codeword bit X_j. Place an equality node on the edge corresponding to X_r, and label this node as the root of the graph. Give a direction to each edge by pointing to its incident node which is closer to the root. For each node, choose an ordering of its two directly preceding nodes as first and second. The ordering is indicated on the graph by adding a dot to the incident edge from the node chosen to be first. Either choice is valid, but the particular choice will have implications for the quantum circuit. However, the decoding performance of BPQM is unaffected by this choice. Finally, name the check and variable nodes arbitrarily with integers $1, \ldots, n-1$.

The resulting graph is defined as an MPG of $\mathcal{C}$ w.r.t. X_r. Note that an MPG is not unique, as the original Tanner graph from which it is derived is generally also not unique. As an example, Fig. 1b depicts an MPG of the 5-bit code w.r.t. X_1. An MPG is always a binary tree with n leaves (channel nodes) which is full, meaning every node has either two or zero predecessors.[8] The number of check and equality nodes must therefore be $n-1$, because a full binary tree with n leaves always has $2n-1$ nodes.[9]

Out of these $n-1$ nodes, precisely $k-1$ are check nodes and $n-k$ are equality nodes. One way to show this fact is with a degree of freedom counting argument, as follows. For this purpose, let us again interpret the MPG without channel nodes as a Forney-style factor graph. It contains precisely $2n-2$ edges, meaning $2n-2$ binary variables are involved. But the number of degrees of freedom is smaller than $2n-2$, as the check and equality nodes impose linear constraints on these binary variables. More precisely, every check node imposes a linear constraint on the three involved variables, whereas every equality node imposes two linear constraints on the three involved variables. If n_c denotes the number of check nodes, the number of equality nodes is thus $n-1-n_c$. Observe that the root equality node only has degree 2 and therefore only imposes a single linear constraint. All these constraints can easily be seen to be linearly independent,[10] since the factor graph in question is a tree. The final number of degrees of freedom is thus

$$n_{\mathrm{dof}} = (2n-2) - n_c - 2(n-1-n_c-1) - 1 = n_c + 1\,. \tag{20}$$

Since the MPG under consideration here was derived from a factor graph encapsulating the constraints of a linear code of dimension k, there must be k independent random variables corresponding to the encoded bits, so $n_{\mathrm{dof}} = k$. This directly implies that $n_c = k-1$.

[8] As opposed to descendants, since the edges are directed toward the root.

[9] This can be seen by induction, since adding a degree three node increases the number of leaves by one and the root equality node has degree two.

[10] Put differently: The matrix representing this homogeneous system of linear equations has the rank $n_c + 2(n-1-n_c-1)+1$.

Node	Check node list	Branch list
θ_5	$\varnothing$	$(\varnothing, \theta_5, 1)$
θ_3	$\varnothing$	$(\varnothing, \theta_3, 1)$
θ_4	$\varnothing$	$(\varnothing, \theta_4, 1)$
θ_2	$\varnothing$	$(\varnothing, \theta_2, 1)$
θ_1	$\varnothing$	$(\varnothing, \theta_1, 1)$
3	(3)	$(i, \theta_5 \boxed{*}_i \theta_3, p_{\boxed{*}}(\theta_5, \theta_3, i))_{i=0,1}$
2	(2)	$(j, \theta_4 \boxed{*}_j \theta_2, p_{\boxed{*}}(\theta_4, \theta_2, j))_{j=0,1}$
4	(3, 2)	$(ij, (\theta_5 \boxed{*}_i \theta_3) \circledast (\theta_4 \boxed{*}_j \theta_2), p_{\boxed{*}}(\theta_5, \theta_3, i) p_{\boxed{*}}(\theta_4, \theta_2, j))_{i,j=0,1}$
1	(3, 2)	$(ij, (\theta_5 \boxed{*}_i \theta_3) \circledast (\theta_4 \boxed{*}_j \theta_2) \circledast \theta_1, p_{\boxed{*}}(\theta_5, \theta_3, i) p_{\boxed{*}}(\theta_4, \theta_2, j))_{i,j=0,1}$

Table 1: Node lists computed by BPQM for the MPG depicted in Fig. 1b. The check and equality nodes are referred to by their name (integer ranging from 1 to n) which they are given in the MPG. To shorten the expressions, we use the notation $p_{\boxed{*}}(\varphi_1, \varphi_2, l) := (1 + (-1)^l \cos\varphi_1 \cos\varphi_2)/2$ for $\varphi_1, \varphi_2 \in (0, \pi)$ and $l \in \{0, 1\}$. The branch list of the root node 1 contains four elements.

3.1.2 Check and branch lists

The MPG will be a central ingredient to define the quantum operations that constitute the unitary V_r. But first, additional classical information needs to be generated from it. This is the compilation step, which keeps track of conditional operations that need to be performed in the circuit. More precisely, the BPQM quantum circuit compiler generates two lists for each node of the MPG. The first is a length-m list of distinct integers in $\{1, \ldots, n-1\}$, where $m \in \{0, \ldots, k-1\}$ is the number of check nodes in the set containing the node in question as well as all its precedessors in the MPG. We refer to this first list as the check node list. The second is a length-2^m list of triples (s, θ, p), where $s \in \mathbb{Z}_2^m$, $\theta \in (0, \pi)$, and $p \in [0, 1]$. We refer to this as the branch list. The final entry in the triple, a probability, will not play a role here in specifying the decoding quantum circuit, but will be useful in analyzing block optimality later.

At the leaves, $m = 0$ and the check node list is empty, denoted by $\varnothing$, and the branch list contains the single entry $(\varnothing, \theta_j, 1)$ at the jth leaf node. The check node and branch lists of the remaining nodes are computed iteratively, as for any node the two lists are defined from those of their direct predecessors. Let L and L' be the check node lists of the two predecessors, of lengths m and m', respectively. Furthermore, let $(s_i, \varphi_i, p_i)_i$ and $(s'_j, \varphi'_j, p'_j)_j$ be the respective branch lists. At equality nodes, the check node list is simply the length-$(m + m')$ list $L \parallel L'$, where '$\parallel$' denotes concatenation. The branch list contains the $2^{m+m'}$ triples $(s_i \parallel s'_j\,, \varphi_i \circledast \varphi'_j\,, p_i p'_j)$ for all i and j, where

$$\varphi \circledast \varphi' := \arccos\left(\cos\varphi \cdot \cos\varphi'\right)\,. \tag{21}$$

Meanwhile for check nodes the lists have lengths $m + m' + 1$ and $2^{m+m'+1}$, respectively. The check node list is simply $t \parallel L \parallel L'$ where t is the name of the check node in question. To construct the branch list, first define, for $l \in \mathbb{Z}_2$,

$$\varphi \boxed{*}_l \varphi' = \arccos\left(\frac{\cos\varphi + (-1)^l \cos\varphi'}{1 + (-1)^l \cos\varphi \cos\varphi'}\right)\,, \tag{22}$$

$$p_{\boxed{*}}(\varphi, \varphi', l) := \tfrac{1}{2}(1 + (-1)^l \cos\varphi \cos\varphi')\,. \tag{23}$$

Then the branch list contains the $2^{m+m'+1}$ triples

$$\left(l \parallel s_i \parallel s'_j,\ \varphi_i \boxed{*}_l \varphi'_j,\ p_i p'_j \cdot p_{\boxed{*}}(\varphi_i, \varphi'_j, l)\right)\,, \tag{24}$$

where l ranges over $\{0, 1\}$ and again i and j range over all their respective values. As an example, Table 1 gives the computed lists for the MPG of the 5-bit code w.r.t X_1 depicted in Fig. 1b.

Observe that the functions $\circledast$ and $\boxed{*}_0$ are symmetric in their arguments, but $\boxed{*}_1$ is not. In fact, $\beta \boxed{*}_1 \alpha = \pi - \alpha \boxed{*}_1 \beta$. Therefore, the choice of which of the two predecessor nodes corresponds to the first argument results in different angle entries in the branch list.

Note that the first entries s of the triples range over all elements in $\mathbb{Z}_2^m$. Thus, it is not strictly necessary to include s in the triple, as it could be specified by order of the triple (now pair) in the

branch list. However, we have opted for this redundancy to avoid having to explicitly deal with the ordering in the description of the compiliation step.

3.1.3 Quantum circuit

Using the MPG and the generated check and branch lists, we can now easily construct a quantum circuit to realize V_r. The circuit can be thought of as the process of passing qubits along the edges of the MPG. First, every leaf node θ_j passes the channel output qubit Y_j to its immediate successor. Once a check or equality node has received a qubit from both of its immediate predecessors, it performs some unitary gate on these two qubits, resulting in two output qubits. One of these qubits is then passed to the immediate successor of the node, whereas the other remains at the node. We call the former *data qubit* and the later *ancilla qubit*. The operation that a node performs on its two input qubits depends on whether the node is an equality or a check node. We call the node operation *equality node operation* or *check node operation* correspondigly. The final data qubit produced by the root of the graph is precisely the qubit that is to be measured in the $|\pm\rangle$ basis in order to obtain the estimate $\hat{X}_r$ of X_r. To fully specify the BPQM algorithm and the quantum circuit implementation of V_r, it thus suffices to specify the quantum gates realizing the check and equality node operations.

The check node operation is simply a CNOT gate. More specifically, choose the control qubit to be the input data qubit coming from the first predecessor in the MPG, as indicated by the dot in the MPG, and the second data qubit to the be target. The control qubit is passed on further as a data qubit, while the target is an ancilla qubit and remains at the check node.

The equality node operation is more involved. It consists of a two-qubit unitary applied to the incoming data qubits, uniformly controlled by the ancilla qubits produced by the predecessor check nodes in the MPG. The unitary in question is given by[11]

$$U_{\circledast}(\alpha,\beta) := \begin{pmatrix} a_+ & 0 & 0 & a_- \\ -a_- & 0 & 0 & a_+ \\ 0 & b_- & b_+ & 0 \\ 0 & b_+ & -b_- & 0 \end{pmatrix}, \tag{25}$$

$$a_\pm := \frac{1}{2}\frac{\cos(\frac{\alpha-\beta}{2}) \pm \cos(\frac{\alpha+\beta}{2})}{\left|\cos(\frac{\alpha\circledast\beta}{2})\right|}, \tag{26}$$

$$b_\pm := \frac{1}{2}\frac{\sin(\frac{\alpha+\beta}{2}) \pm \sin(\frac{\alpha-\beta}{2})}{\left|\sin(\frac{\alpha\circledast\beta}{2})\right|}, \tag{27}$$

for parameters $\alpha,\beta \in (0,\pi)$. Again, the first qubit is chosen to be the qubit arriving from the first predecessor in the MPG. At the output of the gate, the first qubit sent to the next node as the data qubit while the second is the ancilla which is retained. The check and branch lists of the MPG specify the precise conditional $U_{\circledast}$ operation that is required. For a given equality node, denote its check list by L and its branch list by $(s_i,\varphi_i,p_i)_i$. Then the equality node operation is the uniformly-controlled unitary

$$\sum_{i,\ell} U_{\circledast}(\varphi_i,\varphi'_\ell) \otimes |s_i\rangle\langle s_i| \otimes |s'_\ell\rangle\langle s'_\ell| \ , \tag{28}$$

where $U_{\circledast}$ acts on the data qubits and the projectors act on ancilla qubits. Precisely which are the ancilla qubits is recorded in the check node lists, as one ancilla qubit has been generated at each check node passed thus far in the algorithm.

The above algorithm is not truly a message-passing algorithm, since the equality node operation does not act on the two received data qubits alone. Furthermore, the uniformly-controlled $U_{\circledast}$ gates can generally only be decomposed into a number of two-qubit gates scaling with the number of different control patterns, i.e. 2^{k-1} [40]. Hence, for codes of finite rate such that k scales linearly

[11] Note that this is a slight variation of the unitary appearing in [1, 2]; the second row is negated. Additionally, b_+ and b_- are interchanged, for reasons which will become apparent in Section 5.2.1.

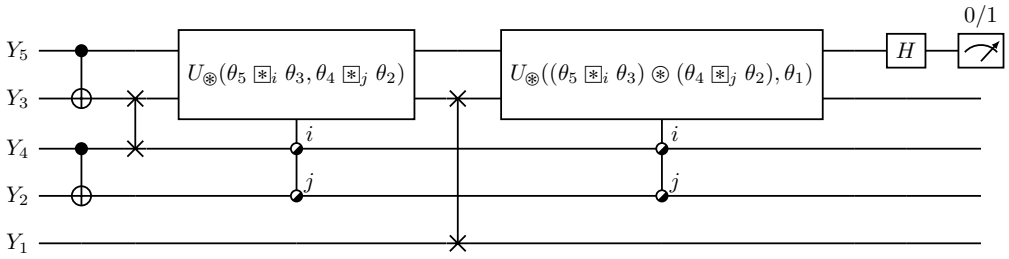


Figure 8: Quantum circuit implementing BPQM decoding of X_1 for the 5-bit code. Without the Hadamard gate and measurement at the end, this circuit realizes the unitary V_1. The symbol ◕ denotes the qubits on which a gate is uniformly controlled.

with n, the resulting quantum circuit depth will be exponential in the blocklength of the code. In fact the exponential overhead already shows up before executing the circuit, in the compilation, which requires an exponential amount of (classical) memory to store the branch list.

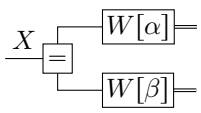
This significant drawback was overlooked in previous literature, mainly because the exponential runtime of the circuit can be avoided in the case where one only decodes a single codeword bit. In this case the ancilla qubits can simply be measured as soon as they are produced, and the appropriate angle arguments for equality node gates $U_{\circledast}$ can be generated for the specific observed cases. However, this hybrid classical-quantum approach is not useful when decoding the complete codeword, as it prevents the node operations from being rewound (which can normally be done by executing $V_r^\dagger$).

3.2 Bit optimality of BPQM

To establish that the single-bit decoding operation for X_r realizes the Helstrom measurement, and thus constitutes the optimal bit decoding, consider the CQ channel from some codeword bit X_r to the output quantum systems $Y_1, \ldots, Y_n$ obtained by passing a randomly-chosen codeword with the prescribed value of X_r through the channel $W[\theta_1] \otimes W[\theta_2] \otimes \cdots \otimes W[\theta_n]$. Optimality is the statement that the decoding quantum circuit realizes the Helstrom measurement for the two output states of the channel.

A convenient means to show this is provided by the Forney-style factor graph representation of the CQ channel and the decoding operations. Such a factor graph of the channel can be constructed by taking the binary tree of the MPG and adding a half-edge to the root equality node and double edges to the channel nodes at the leaves. For example, a factor graph representing this CQ channel for the 5-bit code and $r = 1$ is depicted in Fig. 5b. To this factor graph we may append the factor graph representation of the gates involved in the decoding operation. The resulting graph can be successively simplified by making use of two contraction identities, each of which reduces the number of channel nodes by one. The final simplified graph will be a CQ channel with a single-qubit output, for which the Helstrom measurement is a simple projection operation onto the eigenbasis of σ_x.

Equality node contraction Let $\alpha, \beta \in (0, \pi)$. Consider the CQ channel described by the factor graph



(29)

that produces two-qubit pure quantum states of the form

$$[W[\alpha] \circledast W[\beta]](x) := |Q(x,\alpha)\rangle\langle Q(x,\alpha)| \otimes |Q(x,\beta)\rangle\langle Q(x,\beta)| \tag{30}$$

given the input bit $x \in \{0, 1\}$.

Applying the unitary $U_{\circledast}(\alpha, \beta)$ from (25) as per BPQM yields

$$U_{\circledast}(\alpha,\beta) \cdot |Q(x,\alpha)\rangle \otimes |Q(x,\beta)\rangle = |Q(x, \alpha \circledast \beta)\rangle \otimes |0\rangle \,, \tag{31}$$

using the $\circledast$ angle function from (21). Importantly, the first output qubit, the data qubit, is the output of a pure state channel W for appropriate choice of angle, namely $\alpha \circledast \beta$. That is, the information about the input x has been compressed into a single qubit, in accordance with the fact that the two possible pure output states in (30) only span a two-dimensional space. This is made especially clear in the factor graph contraction identity depicted in Fig. 9a. By linearity, the contraction identity can be extended to the case that the channels $W[\alpha]$ and $W[\beta]$ depend on additional classical information which determines their angle parameters. The extended contraction identity is depicted in Fig. 9b.

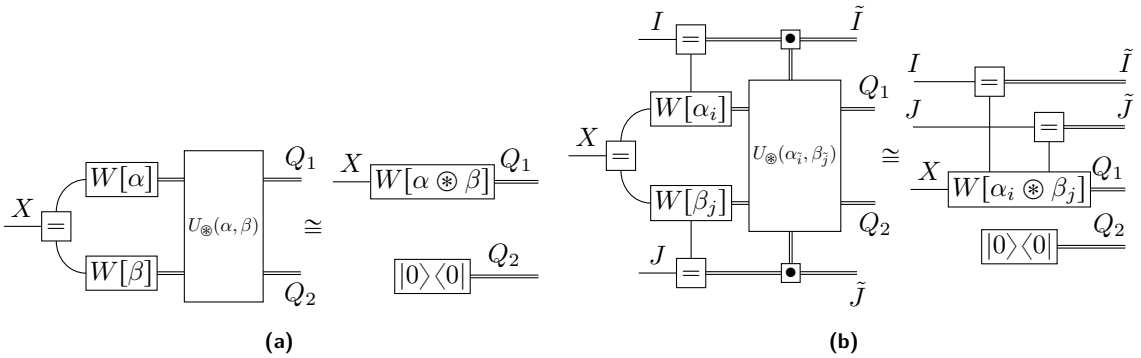
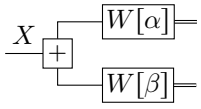


Figure 9: (a) Graphical depiction of equality node contraction identity using Forney-style factor graphs. The equivalence $\cong$ is to be understood as equality of the exterior functions of the two factor graphs. (b) represents a generalizations of this identity, where the channel parameters are classically conditioned on some n_I-bit and n_J-bit systems I and J for some $n_I, n_J \geqslant 0$. $\{\alpha_i\}_{i=0,\ldots,2^{n_I}-1}$ and $\{\beta_j\}_{j=0,\ldots,2^{n_J}-1}$ are some sets of angles between 0 and π.

Check node contraction Let $\alpha, \beta \in (0, \pi)$. Consider the CQ channel described by the factor graph



(32)

which, given the input $x \in \{0, 1\}$, produces mixed quantum states of the form

$$[W[\alpha] \boxplus W[\beta]](x) := \frac{1}{2} |Q(x,\alpha)\rangle\langle Q(x,\alpha)| \otimes |Q(0,\beta)\rangle\langle Q(0,\beta)| + \frac{1}{2} |Q(1-x,\alpha)\rangle\langle Q(1-x,\alpha)| \otimes |Q(1,\beta)\rangle\langle Q(1,\beta)| \,. \tag{33}$$

Applying a CNOT gate on the two involved qubits results in the state

$$\text{CNOT} \cdot [W[\alpha] \boxplus W[\beta]](x) \cdot \text{CNOT}^\dagger = \sum_{l=0,1} p_l \, |Q(x, \alpha \circledast_l \beta)\rangle\langle Q(x, \alpha \circledast_l \beta)| \otimes |l\rangle\langle l| \,, \tag{34}$$

where $p_{0/1} := \frac{1}{2}(1 \pm \cos\alpha \cos\beta)$ and using $\alpha \circledast_l \beta$ from (22). Again the data qubit of the output is the output of a pure state channel. Now, however, there are two possible values for the angle parameter of the channel, and the information about the particular case is stored (classically) in the ancilla qubit. In this case the information about the input is not compressed to a single qubit, but rather to a CQ state of two qubits. The corresponding factor graph contraction identity is depicted in Fig. 10a. It can also be extended by linearity to the case that the parameters of the channels $W[\alpha]$ and $W[\beta]$ are determined by further random variables, as depicted in Fig. 10b.

Return now to the factor graph describing the effective channel from X_r to $Y_1, \ldots, Y_n$ followed by the operations of V_r. Since the former graph is a binary tree, the contraction identities depicted in Figures 9a and 10a can be used to simplify the full factor graph at the channel nodes. Simplifying a given channel pair reduces the number of channel nodes in the graph by one and converts the

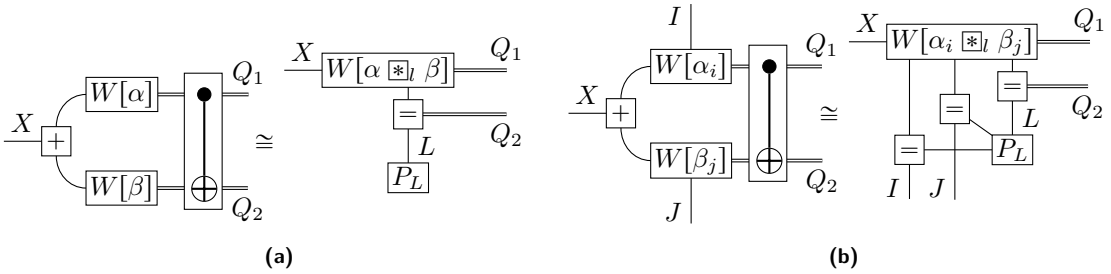


Figure 10: (a) Graphical depiction of check node contraction identity using Forney-style factor graphs. The equivalence $\cong$ is to be understood as equality of the exterior functions of the two factor graphs. (b) represents a generalizations of this identity, where the channel parameters are classically conditioned on some n_I-bit and n_J-bit systems I and J for some $n_I, n_J \geqslant 0$. $\{\alpha_i\}_{i=0,\ldots,2^{n_I}-1}$ and $\{\beta_j\}_{j=0,\ldots,2^{n_J}-1}$ are some sets of angles between 0 and π. In (a) L denotes a 1-bit system which is distributed as $P_L(0) = (1+\cos\alpha\cos\beta)/2, P_L(1) = (1-\cos\alpha\cos\beta)/2$. The corresponding 1-bit system L in (b) is distributed as $P_L(0) = (1+\cos\alpha_i\cos\beta_j)/2, P_L(1) = (1-\cos\alpha_i\cos\beta_j)/2$.

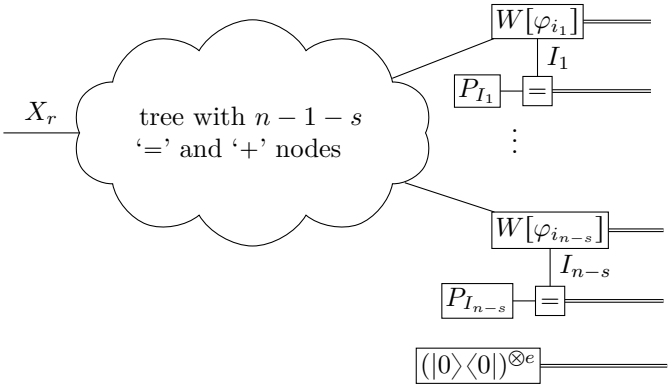
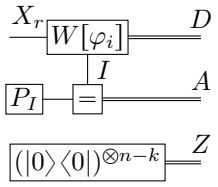


Figure 11: Intermediate state after s contraction steps have been performed, where $s \in \{0,\ldots,n-1\}$. Here I_j is a n_j-bit system for $j = 1,\ldots,n-s$, for some $n_j \geqslant 0$. The P_{I_j} are distributions defined over the system I_j. We denote by e the number contraction steps that involved equality nodes, and therefore $s-e$ is the number of contraction steps that involved check nodes. It must hold that $\sum_j n_j = s-e$.

remaining channel node to a classically conditioned channel node. The extended identities of Figures 9b and 10b can then be used to further simplify these channel nodes in an iterative fashion.

Iteratively simplifying the factor graph in this manner s times to remove s channel nodes results in a factor graph of the form depicted in Fig. 11. After $n-1$ steps, the final simplified factor graph describing the CQ channel from X_r to the output of BPQM is therefore of the following form



(35)

where I is a $(k-1)$-bit random variable, P_I is some distribution defined on I, $(\varphi_0,\ldots,\varphi_{2^{k-1}-1})$ is some set of angles $\in (0,\pi)$, Z is the system of $n-k$ zero qubits produced by equality node operations, A is the system of $k-1$ ancilla qubits produced by check node operations and D is the data qubit produced by the root node. We illustrate this contraction process on the 5-bit example in Fig. 12. A step-by-step version of this example which includes the intermediate steps of the contraction is provided in Appendix A.

The final step in the single bit decoding procedure for the codeword bit X_r is to measure the

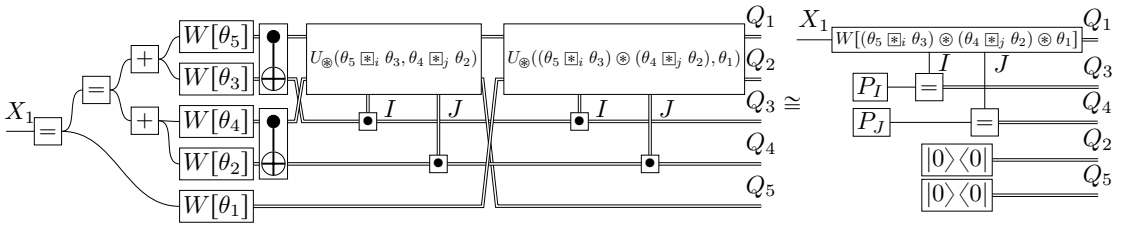


Figure 12: Contraction of the Forney-style factor graph representing the effect of the BPQM algorithm on the 5-bit code. Both factor graphs depict the quantum state of the system after executing V_1 on the channel output. By using the contraction identities from Figs. 9 and 10, this state can be characterized with the simpler factor graph on the right-hand side, which allows for a simple identification of the Helstrom measurement. P_I and P_J are identical distributions of a single-bit system defined by $P_I(0) = P_J(0) = (1+\cos^2\theta)/2$ and $P_I(1) = P_J(1) = (1-\cos^2\theta)/2$. A step-by-step version that illustrates the intermediate steps of this contraction is provided in Appendix A.

remaining data qubit in the $|\pm\rangle$ basis. It is easy to see that measurement in this basis is optimal for distinguishing the outputs of $W[\theta]$ for all values of θ, as the two states are symmetrically arranged about the z-axis in the Bloch sphere no matter the angle between them. Therefore, a $|\pm\rangle$ measurement is also optimal for channels whose angle parameter is set by additional inputs, as in (35). Since all operations in V_r are unitary and thus reversible, BPQM decoding of X_r must be optimal, i.e., it must realize the Helstrom measurement.

4 Block optimality of BPQM

Now we turn to the blockwise performance of the BPQM decoder. Consider again some (n, k) binary linear code $\mathcal{C}$ that exhibits a tree Tanner graph.

Theorem 4.1. *The BPQM algorithm performs block optimal decoding.*

The first step in the proof of this statement is the fact that optimal decoding is realized by the pretty-good measurement (PGM) [7, 8]. Therefore it will suffice to show that BPQM realizes a measurement equivalent to the PGM.

Recall that the PGM for a set of pure states $\mathcal{S} = \{|\phi_1\rangle, \ldots, |\phi_m\rangle\}$ with associated probabilities $p_i, i = 1, \ldots, m$ is given by the POVM with elements rank-1 positive operators $E_i := p_i \rho^{-1/2} |\phi_i\rangle\langle\phi_i| \rho^{-1/2}$, where $\rho = \sum_{i=1}^m p_i |\phi_i\rangle\langle\phi_i|$, and the matrix inverse is taken on its support. When $\sum_{i=1}^m E_i < \mathbb{1}$ the POVM formally requires an additional operator $E_{m+1} = \mathbb{1} - \sum_{i=1}^m E_i$, though note that the probability of observing E_{m+1} is zero for all states in $\mathcal{S}$. Under certain conditions, the PGM is the optimal measurement for determining which state is actually present, averaged over the prior probabilities p_i. Relevant here is geometric uniformity of the states.

Lemma 4.2 (Optimality of the PGM [10–12])**.** *If the $\{|\phi_i\rangle\}_i$ are equally probable, linearly independent and form a geometrically uniform state set, i.e., $\{|\phi_i\rangle\}_i = \{U|\phi\rangle | U \in \mathcal{G}\}$ for some state $|\phi\rangle$ and some finite Abelian group $\mathcal{G}$, then the PGM elements are orthogonal and the PGM distinguishes the states $|\phi_i\rangle$ with the highest possible average probability of success.*

To apply the statement to our setting, denote by $|\Psi_{\vec{x}}\rangle$ the joint pure state output of the quantum channels when the codeword $\vec{x} \in \mathcal{C}$ is transmitted, i.e.,

$$|\Psi_{\vec{x}}\rangle := |Q(x_1, \theta_1)\rangle \otimes \cdots \otimes |Q(x_n, \theta_n)\rangle \,. \tag{36}$$

Then $\mathcal{S}$ is the set of $|\Psi_{\vec{x}}\rangle$, of cardinality $m = 2^k$, and the prior probability of each state is simply $1/m$.

Lemma 4.3. *$\mathcal{S} = \{|\Psi_{\vec{x}}\rangle\}_{\vec{x}}$ is a geometrically uniform state set.*

Proof. This follows rather directly from the realization that we can write $|\Psi_{\vec{x}}\rangle = \sigma_Z^{\vec{x}} |\Psi_{\vec{0}}\rangle$ for any codeword $\vec{x} \in \mathcal{C}$ where $\vec{0} \in \mathcal{C}$ is the all-zero codeword and $\sigma_Z^{\vec{x}} := \sigma_Z^{x_1} \otimes \sigma_Z^{x_2} \otimes \cdots \otimes \sigma_Z^{x_n}$. Since the codewords $\vec{x}$ form an Abelian group under addition, this implies that the unitaries $\{\sigma_Z^{\vec{x}} | \vec{x} \in \mathcal{C}\}$ form an Abelian group under multiplication. □

Since $|Q(0,\theta)\rangle$ and $|Q(1,\theta)\rangle$ are linearly independent, the set $\{|\Psi_{\vec{x}}\rangle\}_{\vec{x}}$ consisting of tensor products of these two vectors must also be linearly independent. The PGM POVM elements therefore consist of the projectors onto the states $|f_{\vec{x}}\rangle := \frac{1}{\sqrt{2^k}}\rho^{-1/2}|\Psi_{\vec{x}}\rangle$ for $\rho := \sum_{\vec{x}} \frac{1}{2^k}|\Psi_{\vec{x}}\rangle\langle\Psi_{\vec{x}}|$.

Before we show that BPQM realizes a measurement equivalent to the PGM, we need to get a better understanding of how the single codeword bit decoding operation acts on the channel outputs.

Lemma 4.4. *Consider a codeword $\vec{x} \in \mathcal{C}$ and the unitary V_r that describes the action of the BPQM node operations corresponding to some MPG G of $\mathcal{C}$ with respect to X_r for $r \in \{1,\dots,k\}$. Then*

$$V_r|\Psi_{\vec{x}}\rangle = \sum_{\vec{j}\in\mathbb{Z}_2^{k-1}} c_{\vec{j}}(x_1,\dots,x_k)\sqrt{p_{\vec{j}}}\,|Q(x_r,\theta_{\vec{j}})\rangle_D \otimes |\vec{j}\rangle_A \otimes |0^{n-k}\rangle_Z\,, \tag{37}$$

where the probabilities and angles $\{(p_{\vec{j}},\theta_{\vec{j}}) : \vec{j}\in\{0,1\}^{k-1}\}$ correspond exactly to the angle and probability entries of the branch list of the root node of G. The $c_{\vec{j}}$ are certain functions $\{0,1\}^k \to \{+1,-1\}$ which fulfill the "orthogonality" property

$$\sum_{\tilde{x}_r} c_{\vec{j}_1}(x_1,\dots,x_k)c_{\vec{j}_2}(x_1,\dots,x_k) = 2^{k-1}\delta_{\vec{j}_1,\vec{j}_2} \quad \forall x_r\in\{0,1\}, \forall \vec{j}_1,\vec{j}_2\in\{0,1\}^{k-1} \tag{38}$$

where $\tilde{x}_r$ denotes all free variables except x_r, i.e., $x_1,\dots,x_{r-1},x_{r+1},\dots,x_k$

Here D denotes the qubit to be measured after V_r, A denotes the $k-1$ ancilla qubits produced by check node operations, and Z denotes the $n-k$ zero qubits arising from equality node operations. The lemma uses the notation $|\vec{l}\rangle := |l_1\rangle\otimes\cdots\otimes|l_m\rangle$ for $\vec{l}\in\{0,1\}^m$ where each ket in the tensor product is a computational basis state of a qubit. The symbol $0^m \in \{0,1\}^m$ denotes the all-zero vector. Due to its length, we delegate the proof to Appendix B.

The lemma can be thought of as a generalization of the contraction process described in Section 3.2. Both describe the state of the system after the BPQM node operations have been applied on the channel output, but from different perspectives. In Lemma 4.4 we assume that we know the full codeword $\vec{x}$ that was transmitted, whereas in the contraction argument we assume all codeword bits except X_r to be uniformly random. This difference makes it impossible to make the same contraction-style argument for Lemma 4.4, and instead a more careful approach has to be used in the proof. A central ingredient is the form of the output of a check node operation, the pure state version of (34):

$$\textsc{cnot}\cdot|Q(y_1,\alpha)\rangle\otimes|Q(y_2,\beta)\rangle = \sum_{j=0}^{1}(-1)^{jy_2}\sqrt{\frac{1+(-1)^j\cos(\alpha)\cos(\beta)}{2}}\,|Q(y_1\oplus y_2,\alpha\,\boxed{*}_j\,\beta)\rangle\otimes|j\rangle\,, \tag{39}$$

where $y_1,y_2\in\{0,1\}$, $\alpha,\beta\in(0,\pi)$ and $\oplus$ denotes addition modulo 2. The proof proceeds by iteratively applying this identity (which is derived in Appendix B) and (31) to $|\Psi_{\vec{x}}\rangle$.

Observe that Lemma 4.4 allows us to retrieve the factor graph contraction result by marginalizing over all inputs except X_r. Consider the state

$$\tau_r := V_r\left(\frac{1}{2^{k-1}}\sum_{\tilde{x}_r}|\Psi_{\vec{x}(x_1,\dots,x_k)}\rangle\langle\Psi_{\vec{x}(x_1,\dots,x_k)}|\right)V_r^\dagger\,, \tag{40}$$

where $\vec{x}(x_1,\dots,x_k)$ is the (unique) extension of the tuple $(x_1,\dots,x_k)$ to a codeword in $\mathcal{C}$. Using Lemma 4.4 we obtain

$$\tau_r = \tfrac{1}{2^{k-1}}\sum_{\vec{j}_1,\vec{j}_2\in\mathbb{Z}_2^{k-1}}\sum_{\tilde{x}_r} c_{\vec{j}_1}(x_1,\dots,x_k)c_{\vec{j}_2}(x_1,\dots,x_k)\sqrt{p_{\vec{j}_1}p_{\vec{j}_2}}\,|Q(x_r,\theta_{\vec{j}_1})\rangle\langle Q(x_r,\theta_{\vec{j}_2})|_D\otimes|\vec{j}_1\rangle\langle\vec{j}_2|_A\otimes|0^{n-k}\rangle\langle 0^{n-k}|_Z\,. \tag{41}$$

The off-diagonal $\vec{j}_1\neq\vec{j}_2$ terms cancel due to the orthogonality property of the $c_{\vec{j}}$ terms, leaving

$$\tau_r = \sum_{\vec{j}\in\mathbb{Z}_2^{k-1}} p_{\vec{j}}\,|Q(x_r,\theta_{\vec{j}})\rangle\langle Q(x_r,\theta_{\vec{j}})|_D\otimes|\vec{j}\rangle\langle\vec{j}|_A\otimes|0^{n-k}\rangle\langle 0^{n-k}|_Z\,, \tag{42}$$

which corresponds exactly to the fully-contracted factor graph encountered in Section 3.2.

We now have all ingredients necessary to prove Theorem 4.1. The main idea is to consider the $|f_{\vec{x}}\rangle$ under the basis transformation defined by the unitary V_r. The above argument implies that for any $r \in \{1, \dots, k\}$ we can write

$$\rho = V_r^\dagger \cdot \sum_{\vec{j} \in \mathbb{Z}_2^{k-1}} p_{\vec{j}} \left(\sum_{z=0,1} \frac{1}{2} |Q(z, \theta_{\vec{j}})\rangle\langle Q(z, \theta_{\vec{j}})|_D \right) \otimes |\vec{j}\rangle\langle\vec{j}|_A \otimes |0^{n-k}\rangle\langle 0^{n-k}|_Z \cdot V_r \,. \tag{43}$$

Furthermore, by Lemma 4.4 we have

$$|\Psi_{\vec{x}}\rangle = V_r^\dagger \sum_{\vec{j}' \in \mathbb{Z}_2^{k-1}} c_{\vec{j}'}(x_1, \dots, x_k) \sqrt{p_{\vec{j}'}} \, |Q(x_r, \theta_{\vec{j}'})\rangle_D \otimes |\vec{j}'\rangle_A \otimes |0^{n-k}\rangle_Z \,. \tag{44}$$

Then, by making use of

$$\sum_z \tfrac{1}{2} |Q(z, \theta_{\vec{j}})\rangle\langle Q(z, \theta_{\vec{j}})| = \cos^2 \tfrac{\theta_{\vec{j}}}{2} |0\rangle\langle 0| + \sin^2 \tfrac{\theta_{\vec{j}}}{2} |1\rangle\langle 1| \,, \tag{45}$$

and inserting (43) and (44) into $|f_{\vec{x}}\rangle = \frac{1}{\sqrt{2^k}} \rho^{-1/2} |\Psi_{\vec{x}}\rangle$, we thus obtain

$$|f_{\vec{x}}\rangle = V_r^\dagger \cdot \frac{1}{\sqrt{2}} (|0\rangle_D + (-1)^{x_r} |1\rangle_D) \otimes \left(\frac{1}{\sqrt{2^{k-1}}} \sum_{\vec{j} \in \mathbb{Z}_2^{k-1}} c_{\vec{j}}(x_1, \dots, x_k) \, |\vec{j}\rangle_A \right) \otimes |0^{n-k}\rangle_Z \,. \tag{46}$$

Now consider the space $\mathcal{H}$ spanned by the vectors $\{|\Psi_{\vec{x}}\rangle\}_{\vec{x}}$, which by the form of the PGM is exactly the space spanned by the orthonormal basis $\{|f_{\vec{x}}\rangle\}_{\vec{x}}$. By (46) it holds that on $\mathcal{H}$ the sequence of V_r, the projection on the qubit D described by $H |m_r\rangle\langle m_r| H$ for $m_r \in \{0, 1\}$, and $V_r^\dagger$ acts identically to the projector

$$\Pi_{x_r = m_r} := \sum_{\vec{x} \in \mathcal{C} : x_r = m_r} |f_{\vec{x}}\rangle\langle f_{\vec{x}}| \,. \tag{47}$$

Since

$$|f_{\vec{m}}\rangle\langle f_{\vec{m}}| = \prod_{i=1}^{k} \Pi_{x_i = m_i} \tag{48}$$

due to the orthonormality of the PGM, this implies that BPQM realizes a measurement equivalent to the PGM when acting any state $|\Psi_{\vec{x}}\rangle$ for $\vec{x} \in \mathcal{C}$.

Let us also comment that while optimal decoding can be performed bitwise in the classical case, it is not necessarily the same as optimal bitwise decoding. Given an optimal decoder, which is a deterministic function $\vec{y} \mapsto \vec{x}'(\vec{y})$ of the observed output, the blockwise-optimal bitwise output for the rth codeword bit is simply the rth bit of $\vec{x}'(\vec{y})$. However, bitwise optimal decoding does not always yield the same result. Take the BSC with crossover probability p and the five-bit code from Fig. 1a as an example.[12] For the output $\vec{y} = 00011$, the optimal decoder uniquely returns the codeword 10011 for all $p \in [0, \frac{1}{2}]$, so the estimate X_1' for the first bit is $X_1' = 1$. The optimal bitwise decoder, however, is only concerned with the joint distribution of X_1 and $Y_1, \dots Y_5$, i.e., the probability of all codewords with $X_1 = 1$ versus those with $X_1 = 0$. In this case the ratio of these probabilities is $(1 - 2p + 2p^2)^2/4(1 - p)^3 p$, which implies that for $p \gtrapprox 0.228$ the estimate will be $X_1' = 0$. The reason that bitwise-optimal decoding and blockwise-optimal decoding coincide in the considered quantum setting can be traced back to the particular structure of the blockwise-optimal decoding (i.e. it is a rank-1 projective measurement) implied by Lemmas 4.2 and 4.3.

[12] This example is due to H. D. Pfister (personal communication).

5 Message-passing BPQM

The goal of this section is to introduce a message-passing algorithm called *message-passing BPQM* which can approximate BPQM to arbitrary precision and which does not require a compilation step and whose quantum circuit implementation only scales polynomially with the blocklength. In BPQM the check nodes store ancilla qubits which indicate the angle arguments needed to perform subsequent equality node operations. The idea of the message-passing algorithm is to circumvent this issue by passing the angle information itself along with the data qubits. Appendix C discusses how this same issue plays out in the case of classical BP.

5.1 Decoding a single codeword bit

5.1.1 Hybrid classical-quantum algorithm

Consider first the simplified case of decoding only a single codeword bit X_r for $r \in \{1, \dots, n\}$. It then suffices to measure the ancilla qubits as they are created at check nodes, as it will not be necessary to rewind the entire decoding operation. The measurement result can then be used to update the angle information.

In more detail, the hybrid classical-quantum message-passing algorithm operates on the MPG for X_r as follows. Messages are passed from the leaves inward into the tree until they reach the root. Each message consists of one qubit and one real number in the range $(0, \pi)$. The initial messages transmitted by the ith leaf nodes consist of the corresponding channel output qubit Y_i as well as the corresponding channel parameter θ_i. As in BPQM, the final qubit output by the root is measured in the $|\pm\rangle$ basis, yielding the decoder's estimate $\hat{X}_r$ of the value of X_r.

The algorithm is fully specified by explaining how equality and check nodes process their two received messages into an output message to be sent up the tree. Denote the first and second data qubits of the two input messages of a node operation by Q_1 and Q_2, respectively, and denote the angle part of both input messages as φ_1, φ_2. The equality node operation consists of performing the unitary $U_{\circledast}(\varphi_1, \varphi_2)$ on the system $Q_1 Q_2$. The output message then consists of the qubit Q_1 as well as the angle $\varphi_1 \circledast \varphi_2$. The check node operation consists of performing a CNOT gate from Q_1 to Q_2 and then measuring the qubit Q_2 in the computational basis. Denote the measurement outcome by $l \in \{0, 1\}$. The output message of the check node contains the qubit Q_1 as well as the angle $\varphi_1 \circledast_l \varphi_2$. Note that in contrary to BPQM, here the node operations do not only consist of quantum gates, but also contain some classical computation.

5.1.2 Optimality of bitwise decoding

Note that in this hybrid quantum-classical algorithm, every message passed over some edge e will be of the form $(|Q(z, \varphi)\rangle, \varphi)$ form some $z \in \{0, 1\}$ and $\varphi \in (0, \pi)$. Put differently, we know the angle of all data qubits passed around the MPG, but not the value of z. This can be verified with a simple inductive argument. It clearly holds for the initial messages sent by the leaves. Suppose $(|Q(z_1, \varphi_1)\rangle, \varphi_1)$ and $(|Q(z_2, \varphi_2)\rangle, \varphi_2)$ are the messages input to either a check or an equality node. At check nodes the required form is a consequence of (39). At equality nodes it follows from the MPG that the state is such that $z_1 = z_2$,[13] so the desired consequence follows directly from (31). Thus, the hybrid classical-quantum message-passing algorithm implements the Helstrom measurement for determining the codeword bit.

5.2 Decoding the complete codeword

To construct a useful codeword decoder by sequentially decoding the individual codeword bits, it suffices to implement the above algorithm coherently. In particular, the angle information can be stored in a quantum register and the measurement of the final qubits (which produce the estimates for the decoded bits) can be replaced by a CNOT gate with a target ancilla qubit initialized in the state $|0\rangle$. Then the now-quantum angle register can be updated at check and equality nodes.

[13] This follows directly from Lemma B.1 in the proof of Lemma 4.4.

Nominally this approach requires an infinite-dimensional quantum angle register to allow for a deterministic differentiation between any possible $\theta \in (0, \pi)$. A natural solution to this problem is to use a discretized representation of the angle using B qubits. In order to simplify the analysis at a later point, we choose to represent an angle θ by its cosine $c := \cos\theta \in (-1, 1)$. Since θ lies in $(0, \pi)$ both representations contain the same information. More precisely, we restrict ourselves to only represent angles with their cosine being $c \in \mathcal{A}_B := \{-1 + 2\frac{1+k}{2^B+1} | k \in \{0, 1, \ldots, 2^B - 1\}\}$ with spacing

$$\delta := \frac{2}{2^B + 1} \tag{49}$$

between the individual representable values. Mathematically, this means that we choose the angle part of the message to be a quantum system living in a 2^B-dimensional Hilbert space with some orthonormal basis denoted by $\{|\text{ϰ}(c)\rangle | c \in \mathcal{A}_B\}$.

The goal of this section is to formalize the algorithm precisely and to show that the resulting decoder approaches the ideal performance of BPQM as B goes to infinity. First we formally introduce the fully coherent message-passing BPQM algorithm that allows for decoding of the complete codeword. Then we will analyze and bound the effect of the discretization errors and determine their effects on the performance of the decoder.

5.2.1 Quantum message-passing algorithm

The decoding of a single message bit X_r can be regarded as an algorithm on an MPG of the code with respect to X_r. Messages are passed from the leaves inward into the tree until they reach the root. Each message consists of a data qubit and a B-qubit register. The latter represents an angle cosine in $\mathcal{A}_B$. The data qubit of the final message generated by the root is measured in the $|\pm\rangle$ basis, producing the decoder estimate for the corresponding message bit. All message processing operations save the final measurement, are unitary and can therefore be undone after the value of X_r has been estimated. This procedure is repeated for all codeword bits to be decoded, just as in BPQM.

The message-passing algorithm is fully specified by describing how equality and check node operations take two messages, process them with some unitary operation, and produce an output message. This is visually depicted with the quantum circuits in Fig. 13. The two data qubits are denoted by D_1, D_2 and the angle registers are denoted by C_1, C_2 in the representation. The output message contains the data qubit D' and the angle register C'. The gates

$$c_1, c_2 \mapsto R_B(c_1 \cdot c_2) \text{ and } c_1, c_2 \mapsto R_B\left(\frac{c_1 + (-1)^l c_2}{1 + (-1)^l c_1 c_2}\right) \tag{50}$$

represent quantum circuits that reversibly compute the corresponding classical functions $\mathcal{A}_B \times \mathcal{A}_B \to \mathcal{A}_B$ in the computational basis, where $R_B : (-1, 1) \to \mathcal{A}_B$ denotes the rounding function

$$R_B(c) = \underset{\tilde{c} \in \mathcal{A}_B}{\arg\min} |c - \tilde{c}| . \tag{51}$$

Since it is understood how to implement these functions on a classical computer, we can simply translate these efficient classical circuits to an efficient quantum circuit. The quantum circuit will generally require a_e, respecitvely a_c, additional "scratch" qubits initialized to $|0\rangle$ for the realization of the computation as a reversible circuit. By uncomputing the classical circuit, we can ensure that these scratch qubits return to $|0\rangle$ after execution of the node operation [41]. This is essential in allowing us to reuse the qubits at a later point without losing coherence of our state.

Notice that each node produces quantum information in C_1, C_2 and D_2 that it does not pass on to the next node. These registers are stored at each node for unwinding the decoding operations at a later point.

It remains to specify the implementation of the uniformly-controlled $U_{\circledast}$ gate depicted in Fig. 13, as the naive implementation would again require a quantum circuit size exponential in B. As shown in Fig. 14a, the $U_{\circledast}(\varphi_1, \varphi_2)$ gate can be decomposed into CNOT gates and single-qubit R_y rotations. The angles of these rotations are given by

$$\alpha = -\arccos(a_+) - \arccos(b_+) \pmod{2\pi} , \tag{52}$$

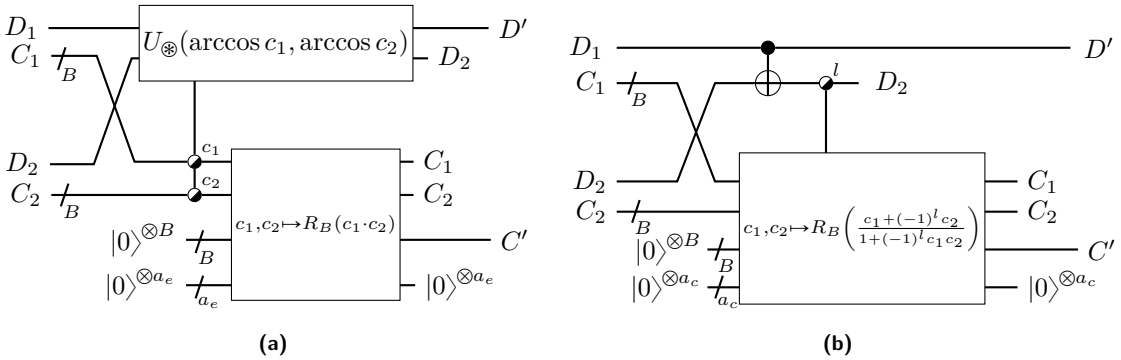


Figure 13: Quantum circuit representation of message-passing processing operations for equality and check node operations. Both types of node operations take two messages (D_1, C_1), (D_2, C_2) and produce an output message (D', C'). The symbol ◕ depicts on which qubits a gate is uniformly controlled.

$$\beta = -\arccos(a_+) + \arccos(b_+) \pmod{2\pi}\,, \tag{53}$$

where, as before,

$$a_\pm := \frac{1}{\sqrt{2}} \frac{\cos(\frac{\arccos c_1 - \arccos c_2}{2}) \pm \cos(\frac{\arccos c_1 + \arccos c_2}{2})}{\sqrt{1 + c_1 c_2}}, \tag{54}$$

$$b_\pm := \frac{1}{\sqrt{2}} \frac{\sin(\frac{\arccos c_1 + \arccos c_2}{2}) \pm \sin(\frac{\arccos c_1 - \arccos c_2}{2})}{\sqrt{1 - c_1 c_2}} \tag{55}$$

and $c_1 := \cos\varphi_1, c_2 := \cos\varphi_2$. For convenience, we choose α, β to both lie in the range $[0, 2\pi]$. We delegate the (short) proof of this identity to Appendix D.

Now only the single-qubit R_y gates need to be controlled by the angle registers. This is easily done by first computing α and β, storing the results in two further B-qubit registers, and then sequentially performing B controlled-R_y rotations, as depicted in Fig. 14b. More specifically, first apply a quantum circuit implementing the classical functions $c_1, c_2 \mapsto \tilde{R}_B(\alpha(c_1, c_2))$ and $c_1, c_2 \mapsto \tilde{R}_B(\beta(c_1, c_2))$, where $\tilde{R}_B : [0, 2\pi] \to \tilde{A}_B := \{2\pi\frac{k}{2^B - 1} | k = 0, \ldots, 2^B - 1\}$ is the rounding function $\tilde{R}_B(\varphi) := \arg\min_{\tilde{\varphi} \in \tilde{A}_B} |\varphi - \tilde{\varphi}|$, and store the result in a two ancilla B-qubit registers. Here again we require a certain number of scratch qubits to realize the classical computation, which we denote by a_{ang}. The R_y rotations can now be uniformly controlled by these ancilla registers.

In contrary to a general uniformly-controlled gate, the two remaining unformly-controlled R_y rotations can be realized efficiently, as described in the following. Consider a R_y rotation gate on the qubit T that is uniformly controlled by the angle φ stored digitally in the B qubits $C_1, \ldots, C_B$, i.e., the state $|b_1\rangle_{C_1} \otimes \cdots \otimes |b_B\rangle_{C_B}$ for $b_i \in \{0, 1\}$ represents the angle $\varphi = 2\pi \sum_{i=1}^n b_i 2^{-i}$. This uniformly-controlled gate can be realized with B controlled R_y gates (i.e. two-qubit gates) where the i-th rotation is targeted on the qubit T and controlled by the qubit C_i and has the angle $2\pi 2^{-i}$.

As we only use a finite number of qubits to represent the angles α and β, this implementation of a uniformly-controlled $U_{\circledast}$ gate exhibits discretization errors, which can in principle be made arbitrarily small by letting B go to infinity.

5.2.2 Performance of the algorithm

For the rest of this section, we focus on bounding the scaling of the discretization error in B so as to determine how large B has to be chosen to reach a desired decoding accuracy. By Lemma 4.4, the state after performing the BPQM unitary V_r according to some MPG G w.r.t X_r on the state $|Q(x_1, \theta_1)\rangle \otimes \cdots \otimes |Q(x_n, \theta_n)\rangle$ (but before performing the projective measurement in the $|\pm\rangle$ basis) for some $\vec{x} \in \mathcal{C}$ can be written in the form

$$\sum_{\vec{j} \in \{0,1\}^{k-1}} \gamma_{\vec{j}} \, |Q(x_r, \theta_{\vec{j}})\rangle_D \otimes |0^{n-k}\rangle_Z \otimes |\vec{j}\rangle_A\,, \tag{56}$$

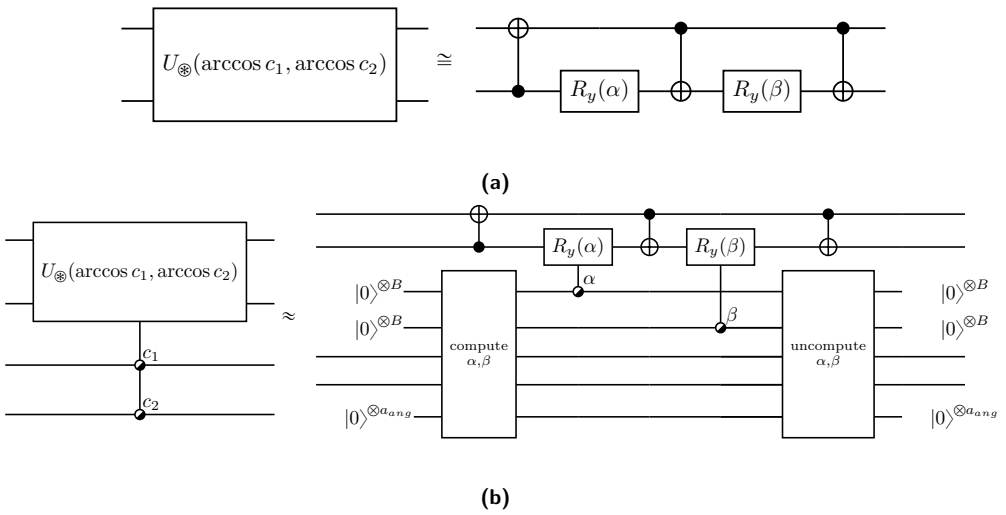


Figure 14: (a) Decomposition of equality node unitary into single-qubit and CNOT gates. The angles of rotation α and β are introduced in the main text in (52) and (54). (b) Efficient realization of a uniformly-controlled equality node unitary. The $\approx$ symbol illustrates that the right-hand side is not perfectly equal to the left-hand side, since the angles α, β are only computed to an accuracy of B bits. However, this discretization error can be made arbitrarily small.

where the angles $\theta_{\vec{j}} \in (0, \pi)$ and squares of the amplitudes $\gamma_{\vec{j}} \in \mathbb{R}$ correspond precisely to the angles and probabilities in the branch list of the root node of G. Here again, we denote by D the output qubit on which the measurement is to be performed, Z the system of ancilla qubits produced by equality node operations and A the system of qubits produced by check node operations. Similarly, we now want to determine how the final state after the execution of message-passing BPQM on the identical MPG and input state looks like:

Lemma 5.1. *After executing the message-passing variant of BPQM according to the MPG G w.r.t. X_r on the state $|Q(x_1, \theta_1)\rangle \otimes \cdots \otimes |Q(x_n, \theta_n)\rangle$ where $\vec{x} \in \mathcal{C}$, one obtains a state of the form*

$$\sum_{\vec{j} \in \{0,1\}^{k-1}} \tilde{\gamma}_{\vec{j}} \, |\tilde{\varphi}_{\vec{j}}\rangle_{DZ} \otimes |\vec{j}\rangle_A \otimes |\measuredangle(\tilde{c}_{\vec{j}})\rangle_C \otimes |\tilde{s}_{\vec{j}}\rangle_S \tag{57}$$

for some $(n-k+1)$-qubit states $|\tilde{\varphi}_{\vec{j}}\rangle$, $\tilde{\gamma}_{\vec{j}} \in \mathbb{R}$, $\tilde{c}_{\vec{j}} \in \mathcal{A}_B$ and some $(n-1)2B$-qubit states $|\tilde{s}_{\vec{j}}\rangle$ such that

1. $\displaystyle\sum_{\vec{j} \in \{0,1\}^{k-1}} \gamma_{\vec{j}}^2 |\cos\theta_{\vec{j}} - \tilde{c}_{\vec{j}}| \leqslant (2^{n+1} - 3)\delta$

2. $\displaystyle\sum_{\vec{j} \in \{0,1\}^{k-1}} |\gamma_{\vec{j}}^2 - \tilde{\gamma}_{\vec{j}}^2| \leqslant 2^{n+\frac{1}{2}} \pi \sqrt{\delta} \tfrac{1}{3} 26^n$

3. $\displaystyle\sum_{\vec{j} \in \{0,1\}^{k-1}} \gamma_{\vec{j}}^2 \left\| |Q(x_r, \theta_{\vec{j}})\rangle \otimes |0^{n-k}\rangle - |\tilde{\varphi}_{\vec{j}}\rangle \right\| \leqslant 2^{n+\frac{1}{2}} \pi \sqrt{\delta} 26^n$

Note that the state is defined on a larger system than before: besides D, Z, and A, there are new systems C and S present in this expression. We denote by C the B-qubit register that contains the angle-part of the message generated by the root node of G. Furthermore, S denotes the system of all qubits produced in the node operations which are not passed on to the next node and are not part of Z or A. By consulting Fig. 13, one can quickly see that there are exactly two such B-qubit registers (denoted C_1 and C_2 in the figure) per node operation. The precise states $|\tilde{s}_{\vec{j}}\rangle$ will not be of any importance, but we cannot discard these qubits, as they are required to undo the node operations at a later point.

Note that $|\tilde{\varphi}_{\vec{j}}\rangle$ describes the joint state of the output data qubit as well as the ancilla qubits produced by equality nodes operations. So, ideally, if discretization errors are small, then we should

hope to find $|\tilde{\varphi}_{\vec{j}}\rangle \approx |Q(x_r, \theta_{\vec{j}})\rangle \otimes |0^{n-k}\rangle$. Similarly, the B-qubit system C should well represent the angle of the data qubits, i.e., $\tilde{c}_{\vec{j}} \approx \cos\theta_{\vec{j}}$. Lemma 5.1 tells us how the discretization errors propagate through the message-passing algorithm. Due to the length and the technical nature, we delegate the proof to Appendix E. We merely note here that the central idea of the lemma is that we do not directly make a statement about the errors $|\cos\theta_{\vec{j}} - \tilde{c}_{\vec{j}}|$ and $\left\| |Q(x_r, \theta_{\vec{j}})\rangle |0^{n-k}\rangle - |\tilde{\varphi}_{\vec{j}}\rangle \right\|$ in each branch of the wave function, but rather make some sort of probability-averaged statement. This is necessary, because the classical computation of the function

$$c_1, c_2 \mapsto \frac{c_1 + (-1)^l c_2}{1 + (-1)^l c_1 c_2} \tag{58}$$

is numerically unstable in the regime where $(-1)^l c_1 c_2$ approaches -1, as the right-hand side is not uniformly continuous in that neighborhood, so some branches of the wave function might experience large errors caused by the discretization. Fortunately, using some algebraic tricks we can show that in the probability-averaged[14] examination of the problem as formulated in Lemma 5.1, the errors remain small, so only branches with small probability amplitudes suffer from strong numerical errors.

Lemma 5.1 can be applied to make a statement about the decoding performance of message-passing BPQM and how it compares to the ideal decoder, which is realized by BPQM.

Theorem 5.2. *Denote by $p^{(mp)}$ and $p^{(ideal)}$ the probabilities of sequentially decoding all the codeword bits $X_1, \ldots, X_k$ correctly using the message-passing BPQM and BPQM algorithms. One has*

$$p^{(ideal)} - p^{(mp)} \leqslant \frac{2^{9/4}\sqrt{\pi}}{\sqrt{3}} n 2^{n\cdot\left(\frac{3}{2} + \frac{1}{2}\frac{\log 26}{\log 2}\right) - \frac{1}{4}B}. \tag{59}$$

This immediately implies that we can choose $B = \mathcal{O}(n + \log\frac{1}{\epsilon})$ to guarantee that the probability of successful decoding deviates by at most ϵ from the ideal value. Due to its verbosity, we delegate the proof to Appendix F. The main idea of the proof is to use Lemma 5.1 iteratively to capture the error caused by the correponding message-passing BPQM operations, as well as the rewinding thereof. In order for Lemma 5.1 to be applicable, it is of central importance to argue that the intermediate states obtained after decoding a certain number of codeword bits are close to a state in the subspace spanned by the PGM basis elements. This is required, since Lemma 5.1 only makes a statement about how the action of BPQM and message-passing BPQM differ on states of the form $|Q(x_1, \theta_1)\rangle \otimes \cdots \otimes |Q(x_n, \theta_n)\rangle$ and not on general states.

5.3 Circuit complexity

In this section we discuss the quantum circuit complexity of message-passing BPQM. In the case where we decode only a single codeword bit as described in Section 5.1, one requires exactly n qubits, one for each channel output. The circuit depth scales with the number $n-1$ of node operations, and each node operation can be implemented in constant depth. The calculations done by the classical co-processor, which keeps track of the involved angles, are considered negligible.

The analysis is a bit more involved for the codeword decoder. The number of node operations now scales as $(n-1)k = \mathcal{O}(n^2)$ as we have to decode k codeword bits. The complexity of every individual node operation depends on B as well as the precise implementation of the involved classical functions. For classical computers the best known algorithm to evaluate these functions is based on the arithmetic-geometric mean operation [42] and when used in conjunction with the Harvey-Hoeven algorithm [43] that allows for multiplication of B-bit numbers in $\mathcal{O}(B\log B)$ time, the time complexity of the classical operation would be $\mathcal{O}(B\log^2 B)$.[15]

It is not clear if and how these classical algorithm can be realized in a quantum circuit of the same asymptotic complexity. The current research on implementing transcendental functions, like trigonometric functions, on quantum computers is sparse [44–47]. Wang *et al.* recently proposed

[14] The probability average is taken over the possible ancilla states $\vec{j} \in \{0,1\}^{k-1}$ with probabilities $\gamma_{\vec{j}}^2$.

[15] It should be noted that the mentioned algorithms are only optimal when considering asymptotic runtimes for very large B. In practice, different algorithm would likely be more efficient when considering realistic values of B.

an algorithm that can realize our desired classical computation using $\mathcal{O}(B^3)$ operations and $\mathcal{O}(B^2)$ qubits [47]. Using this result, message-passing BPQM exhibits a circuit depth that scales as $\mathcal{O}(nkB^3)$.

We require $(1+B)n$ qubits to store the n initial messages and each of the $(n-1)$ node operations consumes a B-qubit register, as seen in Fig. 13. Next to these $(1+B)n + B(n-1)$ qubits, we also require $\max\{a_e, a_c, a_{ang}\}$ additional qubits as scratch register to realize the classical functions involved in every node operation, which scales as $\mathcal{O}(B^2)$ using the result from Wang *et al.*. These qubits can be reused at every node. Therefore, the message-passing algorithm exhibits a circuit width that scales as $\mathcal{O}(B^2 + nB) = \mathcal{O}(n, B^2)$.

Again, we emphasize that the exact asympotic scaling in B strongly depends on the realization of the classical computation, so no large importance should be attributed to the exact polynomial scaling factors used here. It should also be noted that the circuit depth could potentially scale more favorably for some certain families of codes. For instance, if the MPG is close to being a balanced binary tree, then the node operations can be performed in parallel. If we additionally assume a model of computation in which the classical scalar operations take constant time (or equivalently we just take B to be fixed), then our circuit depth instead asymptotically scales as $\mathcal{O}(k \log n)$.

If instead we want to consider a model of computation that takes into account that B has to be chosen larger to reach certain accuracies, we can directly apply Theorem 5.2: If we want to achieve a fixed decoding error ϵ, we require $B = \mathcal{O}(n + \log \frac{1}{\epsilon})$. Taking that into account, the resulting circuit complexities can be summarized as follows:

Theorem 5.3. *Consider the message-passing BPQM algorithm applied to a code of length n such that the probability of successful decoding differs by at most ϵ from the ideal value. The circuit depth and width satisfy*

$$\text{depth} = \mathcal{O}(n^5, \log^3 \tfrac{1}{\epsilon}), \qquad \text{width} = \mathcal{O}(n^2, \log^2 \tfrac{1}{\epsilon})\,. \tag{60}$$

5.4 Numerical results

We present some numerical results on how the number of qubits B used to represent the angle-part of the messages impacts the decoder performance. As an example, we consider a (17,11) code for which the decoding of X_1 can be realized with the MPG depicted in Fig. 15a. This code was chosen because it exhibits multiple subsequent equality and check node operations. Figure 15b depicts the suboptimaility ϵ of the decoder, i.e., how much the probability of successfully decoding X_1 differs compared to the ideal Helstrom decoder, when applied on the all-zero codeword. As we increase B, the suboptimality is seen to decrease exponentially quickly, which is in accord with the prediction from Theorem 5.2. However, the observed error is many orders of magnitude smaller than the values suggested by the bound in Theorem 5.2. The bound should mostly be considered as an analytic proof about the asymptotic scaling and does not accurately reflect the real number of qubits B required to reach a certain decoding accuracy.

Note that message-passing BPQM involves more than $n(B+1)$ qubits, so a brute-force simulation of the circuit is infeasible for interesting values of B. Due to the structure of the algorithm, a more sophisticated simulation procedure can significantly reduce the memory requirements. For more details about the chosen linear code and the simulation procedure, we refer to Appendix G.

6 BPQM on non-tree graphs

Classical BP is commonly formulated as a message-passing algorithm on the Tanner graph, where each variable and check node receives messages, processes them and transmits the output to its neighbors. When the Tanner graph in question is a tree, there is a single natural way to define this algorithm: every node waits for all inputs to be received, and only then it generates the output message. This results into a deterministic and optimal marginalization.

When the Tanner graph contains cycles, however, it is no longer possible to wait for all inputs to be obtained before generating an output message. For this reason, one has to carefully define a schedule according to which the message processing of the nodes operates. For the purposes of this work, we will follow the convention from [13]: Message passing is performed in a sequence of

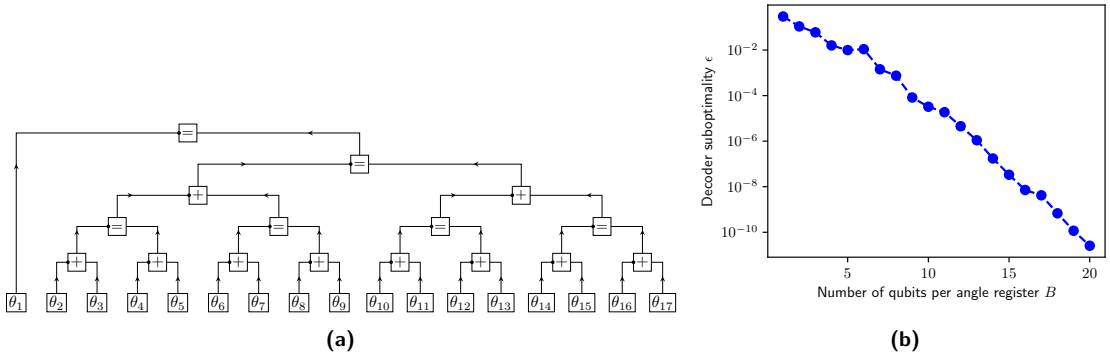


Figure 15: (a) MPG of a (17,11) code w.r.t. X_1. The labels of the edges are not depicted. (b) Decoder suboptimality ϵ of the (17,11) code obtained through numerical simulations. The size B of the angle registers is varied. The codeword to be decoded is the all-zero codeword.

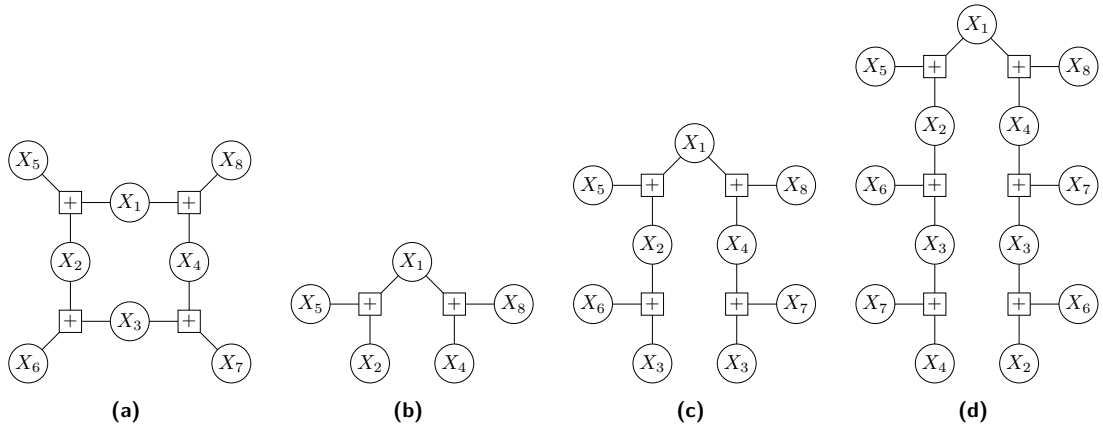


Figure 16: Tanner graph of the (8,4) code $\mathcal{C}$ and associated X_1 computation trees for $h = 1, 2, 3$.

rounds, where each round starts with the check nodes processing their inputs and sending their outputs to the variable nodes, upon which the variable nodes process their inputs and output the result among all their edges. Before the first round is executed, all variable nodes simply send the messages received from the channel output to their edges.

Consider a Tanner graph containing cycles on which classical BP is executed for h rounds in order to decode the codeword bit X_r. The output of the algorithm is obtained from local node operations, so we can unroll the Tanner graph into a tree of depth h that exactly represents this computation. We refer to this as the *depth-h computation tree of X_r*. Consider as an example the (8,4) linear code with Tanner graph depicted in Fig. 16a. The associated computation trees of X_1 for $h = 1, 2, 3$ are depicted in Figs. 16b to 16d. The depth-h computation tree exactly represents the computations required to obtain the output of BP after h time steps.

But it can also be interpreted in a different manner: As discussed in the introduction and depicted in Fig. 2, by associating a unique variable to each node we can consider the unrolled depth-h graph to be the Tanner graph of an (n', k') code, which we denote by $\mathcal{C}'$. In this sense, performing classical BP on the original non-tree factor graph of $\mathcal{C}$ for h time steps is equivalent to performing classical BP on the tree factor graph of $\mathcal{C}'$. Note that for this purpose, certain bits of the noisy channel output must be duplicated, since they appear at multiple locations. For example, in the $h = 3$ case of the 8-bit code, the variables X_2, X_3, X_4, X_6, X_7 all appear twice in the computation tree, so the corresponding channel outputs must be appropriately duplicated before decoding.

We translate this approach to the setting of BPQM by making use of approximate cloning both in the transformation of the actual channel output to something suitable for a $\mathcal{C}'$ decoder and in fixing the channel parameter inputs, i.e. angles, to the BPQM algorithm. Let us now

formalize the ideas outlined above. Consider the task of decoding the n-qubit channel output $|\Psi_{\vec{x}}\rangle := |Q(x_1,\theta_1)\rangle \otimes \cdots \otimes |Q(x_n,\theta_n)\rangle$ for input $\vec{x}$ a codeword of an (n,k) binary linear code $\mathcal{C}$. This can be done by choosing some new (n',k') binary linear code $\mathcal{C}'$ as well as a map $\xi : \{1,\dots,n'\} \to \{1,\dots,n\}$ such that

$$\begin{aligned} \lambda : \mathbb{F}_2^n &\to \mathbb{F}_2^{n'} \\ (x_1,\dots,x_n) &\mapsto (x_{\xi(1)},\dots,x_{\xi(n')}) \end{aligned} \tag{61}$$

maps codewords in $\mathcal{C}$ to codewords in $\mathcal{C}'$ by re-arranging, duplicating, and possibly even deleting codeword bits. The map ξ partially specifies how to transform the n-qubit channel output $|\psi_{\vec{x}}\rangle$ to an n'-qubit state ρ'. Define $n_i := |\{i' \in \{1,\dots,n'\} | \xi(i') = i\}|$ for $i \in \{1,\dots,n\}$. If $n_j = 1$, then there exists exactly one $j' \in \{1,\dots,n'\}$ such that $\xi(j') = j$ and the j'th qubit of ρ' is simply the j-th qubit of $|\psi\rangle$. If $n_j = m$ for some $m > 1$, then there exist m distinct indices $j'_1,\dots,j'_m \in \{1,\dots,n'\}$ such that $\xi(j'_l) = j$. Then the j-th qubit of $|\psi\rangle$ has to be approximately cloned by an appropriate operation, resulting in n_j approximate clones. The qubits $j'_1,\dots,j'_m$ of ρ' are then taken to be these clones. Choosing the cloning operation then fully specifies the transformation of $|\psi_{\vec{x}}\rangle$ to $\rho'_{\vec{x}}$. Finally, it remains to specify a choice of angle parameters θ'_j for $j = 1 \dots n'$, for the BPQM algorithm.[16] $\mathcal{C}'$ and ξ are fully specified from the Tanner graph by the computational unrolling strategy for a given number of time steps h. There is however still freedom in the choice of approximate cloning operation as well as the channel parameters θ'_j.

A very natural (but not necessarily optimal) choice of approximate cloner for $n_j = 2$ is to take the adjoint equality node unitary $U_{\circledast}(\theta',\theta')^\dagger$ for $\theta' = \arccos\sqrt{\cos\theta}$, which maps the state $|\pm\theta\rangle|0\rangle$ to $|\pm\theta'\rangle|\pm\theta'\rangle$. We denote this strategy as *ENU cloner* where ENU stands for equality node unitary. The main advantage of the ENU cloner is that it produces a product state with two qubits with known angle θ'. It is therefore naturally clear how to choose the associated θ'_j. Note that the ENU cloner can also be easily generalized for $n_j \geqslant 3$ by mapping the state $|\pm\theta\rangle|0\rangle^{\otimes n}$ to $|\pm\theta'\rangle^{\otimes n}$ where $\theta' = \arccos(\cos(\theta)^{1/n_j})$. One can quickly check, that this generalized cloner can be realized by a sequence of n_j two-qubit unitaries $U_{\circledast}$ with appropriate angles.

In Section 6.2 we will consider a more refined choice for an approximate cloner.

Just as in standard BPQM, it is necessary to reverse all the decoding operations for a given codeword bit, including any cloning operations but excluding the final measurement, before proceeding to decode the next codeword bit. Note that this reversibility of the cloning does not imply that the cloning process must necessarily be unitary. By implementing a general approximate cloner through its Stinespring dilation [48], we can revert its action at a later point by keeping track of the produced ancilla states.

6.1 Numerical results

We now present numerical results on decoding the 8-bit code introduced above using BPQM. We generate the tree factor graph used for decoding by computationally unrolling around the target codeword bit for h rounds. In cases where the unrolling is deep enough such that certain variables appear multiple times, we use the ENU to approximately clone the corresponding qubits.

Figures 17a and 17b depict the probability of successful decoding of the bits X_1 and X_5, respectively, as a function of the channel parameter θ, identical for all channels. Figure 17c depicts the probability of successfully decoding the complete codeword. Included for comparison are the optimal classical bit-MAP and block-MAP (codeword) decoders, applied after performing the optimal Helstrom measurement for the CQ channel on each quantum channel output individually. The performance of the optimal quantum decoders can be obtained using the formulation as a semidefinite program (see e.g. [48, §3.1]). In the bit-MAP case it is the Helstrom measurement of the corresponding channel from input codeword bit to quantum outputs, whereas in the codeword case it is realized by the PGM.

The numerics reveal that our approach can significantly outperform the best possible classical decoder in any of the analyzed cases. Furthermore we see that it is not beneficial to make the

[16] Recall that the input of the BPQM algorithm is not only a set of qubits $Y_1,\dots,Y_{n'}$ but also a set of angles $\theta'_1,\dots,\theta'_n$ that represent channel parameters.

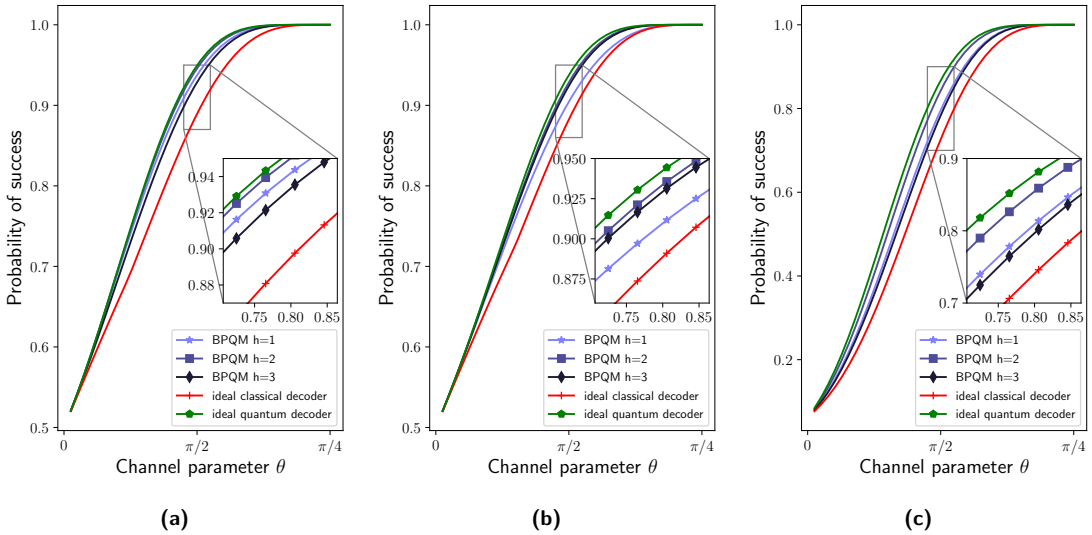


Figure 17: Numerical results from decoding X_1 (a), X_5 (b) and the complete codeword (c) in the 8-bit code depicted in Fig. 16a.

number of rounds h as large as possible—at some point the benefits of considering more rounds is smaller than the drawback incurred by having to do more approximate cloning. This is a strongly different characteristic compared to the classical belief propagation, where increasing the number of rounds generally improves the quality of the result.

One might wonder whether cloning is really necessary to achieve the best possible performance. For instance, it may be possible to simply employ BPQM defined for a spanning tree of the original Tanner graph. In Appendix H we investigate such alternative decoders and argue that it is unlikely that any such decoder can reach the same performance as the ENU+BPQM decoder.

6.2 Optimal approximate cloner

While the ENU cloner is conceptually very simple, there is a priori no reason why it should be the best choice of an approximate cloner. In this section we restrict ourselves to the simplest case in which the cloning operation produces just two approximate copies of the input. Bruß *et al.* [49] characterized the operation which maps $|Q(0,\theta)\rangle|0\rangle$ and $|Q(1,\theta)\rangle|0\rangle$ to some two-qubit states $|\phi_0\rangle$ and $|\phi_1\rangle$, respectively, which maximizes the average global (squared) fidelity

$$\tfrac{1}{2}\left(|\langle Q(0,\theta)|\langle Q(0,\theta)|\cdot|\phi_0\rangle|^2+|\langle Q(1,\theta)|\langle Q(1,\theta)|\cdot|\phi_1\rangle|^2\right). \tag{62}$$

It turns out that the optimal cloner unitarily embeds the two input states into the span of the ideal cloned states. A particularly simple implementation is actually just given by $U_{\circledast}(\theta,\theta)^\dagger$, i.e. the ENU cloner for the "wrong" angle.

When using this unitary to clone the channel outputs, it is not immediately clear what angle parameter to assign to the associated channels in the BPQM algorithm. We address this problem numerically. Consider decoding X_1 in the 8-bit code using $h=3$ for a fixed channel parameter θ. Several of the qubits have to be cloned to produce two approximate clones. Numerical simulation of the decoding success probability is shown in Fig. 18 for a range of channel parameters θ' used in the BPQM algorithm for the approximate clones. Interestingly, the optimal decoder can outperform the ENU cloner if θ' is chosen suitably. Furthermore, it would seem that the optimal value of θ' angle is very close to the value of $\arccos\sqrt{\cos\theta}$, the value used by the ENU cloner.

7 Future directions

In this section we explore remaining open questions as well as possible avenues for future research directions related to the BPQM algorithm. We should note that our focus here is on theoretical

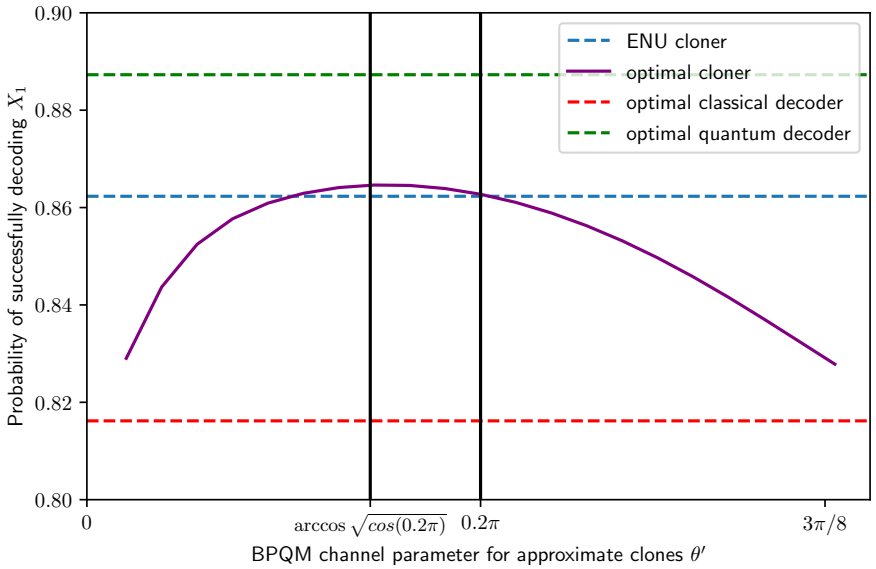


Figure 18: Decoding performance of optimal cloner on the 8-bit code with number of rounds $h = 3$. The angle θ' is varied. The channel parameter is fixed at $\theta = 0.2\pi$.

aspects of the algorithm, and we do not address the also interesting issues of hardware implementations. For more discussion of such issues, including the prospects for surpassing the performance of decoders which first measure the output qubits individually, see [2].

7.1 Potential improvements and extensions

An immediate question is how well BPQM fares when decoding LDPC codes, the usual application of BP decoding in the classical case. It might even be possible to show that LDPC codes paired with BPQM decoding can achieve the Holevo capacity, particularly the "spatially-coupled" variant which achieves the classical capacity [3]. A central question to be answered for this is how much the approximate cloning deteriorates the quality of the decoder for these families of codes.

Another aspect to be analyzed when choosing appropriate codes is the speed of the decoder. Assuming a fixed accuracy B, we have shown in Section 5.3 that the decoding circuit depth scales as $\mathcal{O}(kn)$ at worst for an arbitrary (n, k) binary linear code. However, if the topology of the MPG allows us to parallelize some of the node operations, the circuit depth could be reduced up to $\mathcal{O}(k \log n)$ in the best case. It would therefore be interesting to study which families of codes would allow such decoding in quasilinear time.

There is also another potential optimization to reduce the quantum circuit size, which we haven't explored in this work. In some cases it might be possible that certain operations in subsequent circuit parts cancel out. For instance, the final node operations in $V_j^\dagger$ might cancel with the first node operations in V_{j+1} for some $j \in \{1, \ldots, k-1\}$. To make use of this optimization, the order in which we decode the codeword bits suddenly becomes relevant. It would be interesting to study in future work what families of codes could benefit most from this optimization.

It should also be noted that the analysis of the discretization errors in this work includes many crude bounds. It cannot be excluded that a more refined analysis and/or some tweaks to the message-passing algorithm itself might exhibit a more favorable scaling, for instance the required B to reach a fixed accuracy ϵ might scale sublinearly in n.

Generally, given some code $\mathcal{C}$ with non-tree Tanner graph, there is considerable freedom in how exactly one decodes that code with BPQM. As explained in Section 6, one has to make some choice of (n', k') code $\mathcal{C}'$ with tree Tanner graph, a map $\xi : \{1, \ldots, n'\} \to \{1, \ldots, n\}$ that determines how to map codewords from $\mathcal{C}$ to $\mathcal{C}'$, some choice of channel parameters θ'_j as well as an approximate cloning procedure. It is not clear what the best choice is, and in fact the optimal strategy is likely to vary strongly depending on the considered code. For instance, computational unrolling is

unlikely to yield good results when the Tanner graph contains many small cycles.

Furthermore, it should be noted that our framework is unlikely to be the most general approach to deal with Tanner graphs that contain cycles. The approximate cloning procedure is executed purely before the message-passing on the MPG generated by $\mathcal{C}'$. However, it is plausible that the approximate cloning could also be performed interleaved with the message-passing operations, e.g., the output of some node operation could be cloned as to be passed to two successors insted of just one. In this case, our picture with the MPG would not be applicable anymore, since the data flow caould not be described by a tree.

Finally, it would be very interesting to extend BPQM to more general settings. One possibility would be to study general (non-pure) CQ channels. In that case, we expect BPQM to generally lose its block-optimality, since the noisy channel outputs generally do not form a geometrically uniform set (which was a crucial ingredient in Section 4). This would not be too surprising, as classical BP generally also does not perform block-optimal decoding, but only bitwise optimal decoding (assuming a code with tree Tanner graph).[17] As a further step, it would also be interesting to see if BPQM could even be generalized to decode quantum information transmitted over some form of a quantum channel.

7.2 MPGs not from the Tanner graph

The BPQM decoding algorithm is constructed using a tree MPG describing the codewords of the code, which we have shown can be constructed from a tree Tanner graph. However, it is not necessary to have a tree Tanner graph to have a tree MPG representation. For instance, the (8,5) code associated with the 8 channel leaves θ_2 through θ_9 in Fig. 15a has three parity-checks, leading to 168 possible parity-check matrices, none of which has a tree Tanner graph.

Moreover, MPGs need not come from the Tanner graph. An important instance comes from successive cancellation decoding of Reed-Muller or polar codes [50], as mentioned in [1]. Let us describe how to apply BPQM in this case more precisely. Successive cancellation decoding proceeds by sequential decoding the individual bits of the message (not codeword) as input to the encoding circuit. For each message bit, the encoding circuit itself defines a "synthesized" channel from that bit to all outputs of the noisy channel, as well as all preceding message inputs. Again the goal is to implement the Helstrom measurement to distinguish the two possible channel outputs. In the case of Reed-Muller and polar codes, the synthetic channels have a perfect binary tree structure consisting of repeated channel convolution of two different types, which we might call "better" and "worse". These are very closely related to the check and variable node channel convolutions depicted in (29) and (32).

Indeed, the worse channel convolution is precisely the check convolution of (32). The better convolution is ultimately equivalent to the equality convolution for symmetric channels such as the pure state output channel under consideration here. The better channel convolution of two channels W_1 and W_2 is $W'(x) = \sum_{y\in\mathbb{Z}_2} |y\rangle\langle y| \otimes W_1(x) \otimes W_2(x \oplus y)$. When W_2 is symmetric in the sense that $W_2(x \oplus y) = U_y W_2(x) U_y^\dagger$ for an appropriate unitary U_y, then a control-U_y operation from the classical output $|y\rangle\langle y|$ transforms the output to $\frac{1}{2}\mathbb{1} \otimes [W_1 \circledast W_2](x)$, the equality node convolution.

Due to the structure of the encoding circuit for Reed-Muller and polar codes, the additional classical outputs $|y\rangle\langle y|$ arising from the worse convolution are precisely the preceding input message bits of the synthesized channels. The output of the synthesized channels corresponding to the preceding already-decoded input messages can therefore be decoupled from the outputs of the noisy channel, at which point the decoding task is precisely to perform the Helstrom measurement to distinguish the two outputs of a channel described by a perfect tree MPG. Hence BPQM can be used for this purpose.

For a general CQ channel, an efficient implementation of the successive cancellation decoder is not known [51, 52]. Guha and Wilde examine the pure state case considered here in [53], but also do not construct an efficient decoder. In [1] it is claimed that BPQM leads to an efficient decoding algorithm for this case, but as we have seen above, this is more correctly a statement

[17] Note that in the classical setting, the block-optimal solution could instead be obtained using the max-product or min-sum algorithm.

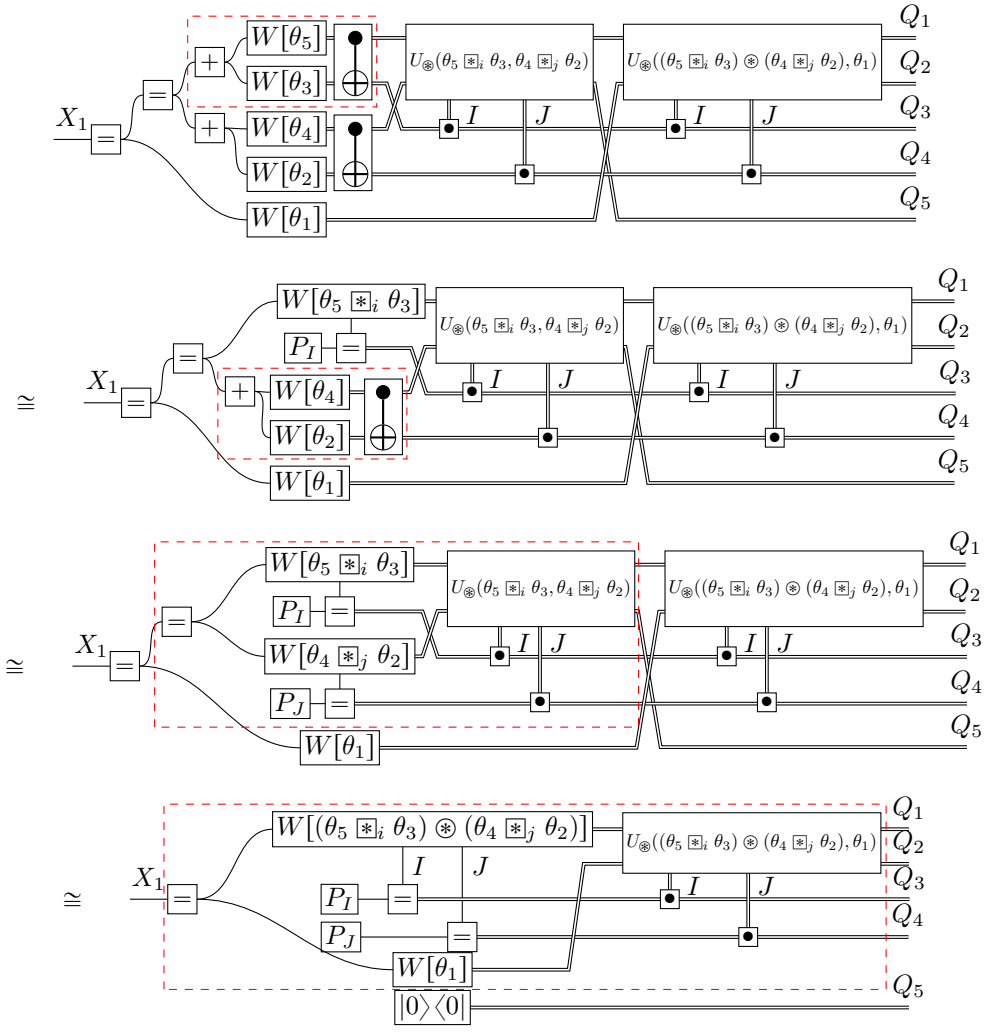
about the message-passing approximation of BPQM instead. Nevertheless, since the synthetic channels "polarize", i.e., become reliable in the large blocklength limit, and BPQM implements the optimal Helstrom measurement, the argument about the total error in decoding the entire message presented in [1] goes through, ensuring that the resulting BPQM successive cancellation decoder achieves capacity for the pure state channel.

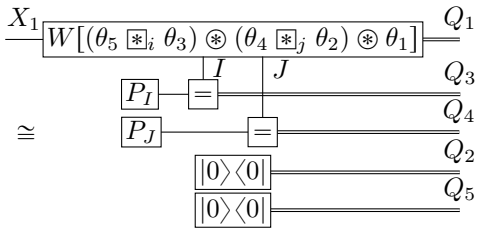
In light of the block optimality of BPQM, it would be interesting to investigate whether BPQM gives an optimal decoder for polar or Reed-Muller codes and pure state channels. However, one major difference to the MPGs generated from the Tanner graph is that each message bit requires a completely different MPG, not one obtained by moving the degree-two root node to the appropriate location on a fixed graph. Another case worth examination is if BPQM can be applied to decoding of turbo codes using the associated state FFG [13, Chapter 6].

Acknowledgments
This work was supported by the Swiss National Science Foundation, through the National Center of Competence in Research "Quantum Science and Technology" (QSIT) and through grant number 20QT21_187724. The authors are grateful to Narayanan Rengaswamy, Henry D. Pfister, and Andru Gheorghiu for useful discussions.

A Example of factor graph contraction on 5-bit code

In this section we illustrate step-by-step the contraction process described in Fig. 12 on the 5-bit code to decode the codeword bit X_1. The red dashed box indicates where the contraction takes place at each intermediate step.





B Proof of Lemma 4.4

We first show (37), by giving the form of the state at all the intermediate steps in the algorithm. Then we show (38).

For this proof, we have to again consider the factor graph from which the MPG was originally derived. Every edge e in the factor graph corresponds to some binary random variable, which is uniquely determined when given the values of $X_1, \dots, X_k$. Since the factor graph only contains equality and check nodes, the random variable in question must be of the form $g_{e,1}X_1 + g_{e,2}X_2 + \cdots + g_{e,k}X_k$ for some $\vec{g}_e \in \mathbb{F}_2^k$. Since the value $\vec{d} := (x_1, \dots, x_k)$ of $(X_1, \dots, X_k)$ is given in the setting of Lemma 4.4, the value z_e of the random variable associated to e is known to be $z_e = g_{e_1}x_1 + \cdots + g_{e_k}x_k$.

Furthermore, to slightly simplify the notation for this proof, we extend the MPG by adding a half-edge to the root, with the direction pointing away from the root. The final data qubit produced by the root node is then said to be passed over that half-edge.

Lemma B.1. *Consider an MPG G for the binary linear code $\mathcal{C}$ with respect to the codeword bit X_r, for $r \in 1, \dots, k$. Then the joint state of the qubit passed by BPQM over any edge e together with all ancilla qubits produced by check and variable nodes preceding e can be written as*

$$|\Xi(e)\rangle = \sum_{\vec{j} \in \{0,1\}^{k_e - 1}} c_{e,\vec{j}}(\vec{d}) \sqrt{p_{e,\vec{j}}} \, |Q(z_e, \theta_{e,\vec{j}})\rangle_{D_e} \otimes |\vec{j}\rangle_{A_e} \otimes |0^{n_e - k_e}\rangle_{Z_e} \,, \tag{63}$$

where n_e and $k_e - 1$ denote the number of channel and check nodes preceding e, respectively, $c_{e,\vec{j}}(\vec{d})$ are some maps $\mathbb{F}_2^k \to \{+1, -1\}$, and the probabilities and angles $\{(\theta_{e,\vec{j}}, p_{e,\vec{j}}) | \vec{j} \in \{0,1\}^{k_e - 1}\}$ correspond exactly to the probability and angle entries of the branch list of the node directly preceding e in G. Furthermore, D_e denotes the qubit system transmitted over the edge e, A_e denotes all ancilla qubits produced by check nodes preceding e, and Z_e denotes all ancilla qubits produced by equality nodes preceding e.

By choosing e to be the half-edge that we added to the MPG, Lemma B.1 directly implies (37).

Proof. The proof is by induction. The claim holds immediately for edges e connected to leaf nodes in G, since the qubit in question is simply in the state $|Q(z_e, \theta)\rangle$. Therefore we only need to show that if the claim holds for two input edges e, e' to a check or equality node, then it also holds for the output edge $\hat{e}$.

Now we assume the states of edges e and e' incident to a check or equality node are $|\Xi(e)\rangle$ and $|\Xi(e')\rangle$, respectively. We deal with the two cases separately.

Equality node Due to the equality constraint, one has $z_e = z_{e'} =: z$. As described in (28), BPQM applies the unitary

$$\sum_{\vec{j}, \vec{j}'} U_{\circledast}(\theta_{e,\vec{j}}, \theta_{e',\vec{j}'})_{D_e, D_{e'}} \otimes |\vec{j}\rangle\langle\vec{j}|_{A_e} \otimes |\vec{j}'\rangle\langle\vec{j}'|_{A_{e'}} \tag{64}$$

to $|\Xi(e)\rangle \otimes |\Xi(e')\rangle$. By (31), the resulting state is

$$\sum_{\vec{j}, \vec{j}'} c_{e,\vec{j}}(\vec{d}) c_{e',\vec{j}'}(\vec{d}) \sqrt{p_{e,\vec{j}} p_{e',\vec{j}'}} \, |Q(z, \theta_{e,\vec{j}} \circledast \theta_{e',\vec{j}'})\rangle_{D_{\hat{e}}} \otimes \left(|\vec{j}\rangle |\vec{j}'\rangle\right)_{A_{\hat{e}}} \otimes |0^{n_{\hat{e}} - k_{\hat{e}}}\rangle_{Z_{\hat{e}}} \,, \tag{65}$$

which is again of the form of (63) (because $c_{e,\vec{j}} \cdot c_{e',\vec{j}'}$ is again also a map $\mathbb{F}_2^k \to \{+1, -1\}$ and because the probabilites and angles are combined according to the same rules as in the construction of the branch list entries in Section 3.1). Note that here we have $n_{\hat{e}} = n_e + n_{e'}$ and $k_{\hat{e}} = k_e + k_{e'} - 1$.

Check node BPQM applies the gate $\text{CNOT}_{D_e D_{e'}}$ to $|\Xi(e)\rangle \otimes |\Xi(e')\rangle$. By (39), the resulting state is

$$\sum_{\vec{j},\vec{j}',l\in\{0,1\}} c_{e,\vec{j}}(\vec{d}) c_{e',\vec{j}'}(\vec{d}) (-1)^{l\cdot z_{e'}} \sqrt{p_{e,\vec{j}} p_{e',\vec{j}'}} \sqrt{\frac{1+(-1)^l \cos\theta_{e,\vec{j}} \cos\theta_{e',\vec{j}'}}{2}} \, |Q(z, \theta_{e,\vec{j}} \circledast_l \theta_{e',\vec{j}'})\rangle_{D_{\hat{e}}} \\ \otimes \left(|\vec{j}_i\rangle |\vec{j}'\rangle |l\rangle \right)_{A_{\hat{e}}} \otimes |0^{n_{\hat{e}}-k_{\hat{e}}}\rangle_{Z_{\hat{e}}} \tag{66}$$

which is again of the form of (63), by the same argument as the equality node above. Note that here we have $n_{\hat{e}} = n_e + n_{e'}$ and $k_{\hat{e}} = k_e + k_{e'}$. It remains to show (39); it is directly implied by the following identites, which can be derived from standard trigonometric identities:

$$|Q(x, \alpha \circledast_i \beta)\rangle = \cos\left(\frac{\alpha \circledast_i \beta}{2}\right) |0\rangle + (-1)^x \sin\left(\frac{\alpha \circledast_i \beta}{2}\right) |1\rangle \tag{67}$$

$$= \frac{1}{\sqrt{2}} \frac{1}{\sqrt{1+(-1)^i \cos\alpha \cos\beta}} \sum_{j=0}^{1} (-1)^{xj} \sqrt{1+(-1)^j \cos\alpha} \sqrt{1+(-1)^{i+j}\cos\beta} \, |j\rangle \,, \tag{68}$$

$$2\,\text{CNOT}\, |Q(y_1,\alpha)\rangle |Q(y_2,\beta)\rangle = \\ \sqrt{1+\cos\alpha}\sqrt{1+\cos\beta}\,|00\rangle + (-1)^{y_1+y_2}\sqrt{1-\cos\alpha}\sqrt{1-\cos\beta}\,|10\rangle \\ + (-1)^{y_2}\left(\sqrt{1+\cos\alpha}\sqrt{1-\cos\beta}\,|01\rangle + (-1)^{y_1+y_2}\sqrt{1-\cos\alpha}\sqrt{1+\cos\beta}\,|11\rangle\right) . \tag{69}$$

□

Now we turn to proving (38). Let E be the subset of MPG edges defined as follows: E contains the half-edge at the root. Furthermore, for every check node in the MPG, E contains the second edge leading into the node (i.e. the edge that is not denoted by a dot in the MPG).

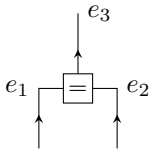
Lemma B.2. *$\{\vec{g}_e | e \in E\}$ spans the space $\mathbb{F}_2^k$.*

Since E contains precisely k elements, the $\{\vec{g}_e | e \in E\}$ thus form a basis of $\mathbb{F}_2^k$. Put differently, this lemma states that the random variables associated to the edges in E are independent and generate all other variables in the factor graph.

Proof. We use an inductive approach, where we visit all edges from the MPG, starting from the half-edge and working towards the leaves. During this traversal, we gradually build up a list $\tilde{E}$ of edges, which at the end of the traversal will end up being exactly E. At any point during the traversal, the following property will hold:

The subspace spanned by $\{\vec{g}_e | $ The edge e has already been visited$\}$ is exactly equal to the subspace spanned by $\{\vec{g}_e | e \in \tilde{E}\}$.

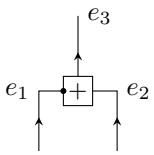
When starting the traversal at the half-edge, we choose $\tilde{E}$ to simply contain the half-edge itself, such that this property trivially holds. Every time we come across an equality node



(70)

one has by definition $\vec{g}_{e_3} = \vec{g}_{e_1} = \vec{g}_{e_2}$, so we do not need to extend $\tilde{E}$ in order for the desired

property to hold. However, when we traverse a check node



(71)

we now have to insert e_2 to the list $\tilde{E}$. Since $\vec{g}_{e_1} = \vec{g}_{e_2} + \vec{g}_{e_3}$, the desired property now holds again. By construction, at the end of the traversal the list $\tilde{E}$ will exactly be equal to E. □

Notice in (39) that the phases in $|\Xi(e)\rangle$ arise from check node operations are determined by the edge variable of the target edge as well as the branch variable. These simply accumulate as the algorithm proceeds, and therefore $c_{e,\vec{j}}(\vec{d})$ takes the form $c_{e,\vec{j}}(\vec{d}) = (-1)^{\vec{d}\cdot\vec{h}_{e,\vec{j}}}$ for some vector $\vec{h}_{e,\vec{j}} \in \mathbb{F}_2^k$. Define $H_e := \{\vec{h}_{e,\vec{j}} | \vec{j} \in \mathbb{Z}_2^{k_e - 1}\}$, with $H_e = \{\vec{0}\}$ for $k_e = 1$. Then we have

Lemma B.3. *For each edge e, H_e is the subspace of $\mathbb{F}_2^k$ spanned by $\{\vec{g}_{e'} | e' \in E$ and e' precedes $e\}$.*

Since the $\vec{g}_e$ are a basis, the span of $\{\vec{g}_{e'} | e' \in E$ and e' precedes $e\}$ has dimension $k_e - 1$, and therefore H_e must contain $2^{k_e - 1}$ distinct elements. For the half-edge at the root, call it e_r, $\vec{g}_{e_r}$ is the standard basis vector with a single 1 in the rth entry. Notice that H_{e_r} does not contain $\vec{g}_{e_r}$.

Proof. As usual, the proof is by induction. At leaf edges $H_e = \{\vec{0}\}$.

Suppose e and e' are the first and second input edges to an equality node with output edge $\hat{e}$. By (65) the phases multiply and therefore the associated $\vec{h}$ vectors add:

$$H_{\hat{e}} = \{\vec{h} + \vec{h}' | \vec{h} \in H_e, \vec{h}' \in H_{e'}\}. \tag{72}$$

This is the subspace spanned by $\{\vec{g}_{\bar{e}} | \bar{e} \in E$ and $\bar{e}$ precedes e or $e'\}$, which is exactly $\{\vec{g}_e | e \in E$ and e precedes $\hat{e}\}$.

Similarly, for e and e' incident on a check node, (66) implies

$$H_{\hat{e}} = \{\vec{h} + \vec{h}' + l \cdot \vec{g}_{e'} | \vec{h} \in H_e, \vec{h}' \in H_{e'}, l \in \mathbb{Z}_2\}. \tag{73}$$

Since e' is an element of E, $H_{\hat{e}}$ is the space spanned by $\{\vec{g}_{e'}\} \cup \{\vec{g}_{\bar{e}} | \bar{e} \in E$ precedes $e\} \cup \{\vec{g}_{\bar{e}} | \bar{e} \in E$ precedes $e'\}$. Again this is precisely $\{\vec{g}_e | e \in E$ and e precedes $\hat{e}\}$. □

For the root node e_r, just write $c_{\vec{j}}$ for $c_{e_r,\vec{j}}$, and similarly for $\vec{h}_{\vec{j}}$. Then

$$\sum_{\vec{x}_r} c_{\vec{j}}(x_1, \ldots, x_k) c_{\vec{j}'}(x_1, \ldots, x_k) = \sum_{\vec{x}_r} (-1)^{\vec{d}\cdot(\vec{h}_{\vec{j}} + \vec{h}_{\vec{j}'})}. \tag{74}$$

The only way to avoid the summation resulting in zero is for $\vec{h}_{\vec{j}} + \vec{h}_{\vec{j}'}$ to be zero everywhere except possibly in the rth position, i.e., $\vec{h}_{\vec{j}} + \vec{h}_{\vec{j}'} = l\vec{g_{e_r}}$ for $l \in \{0, 1\}$. Since the $\vec{h}_{\vec{j}}$ are all distinct, the $l = 0$ option is only possible when $\vec{j} = \vec{j}'$. The $l = 1$ option is impossible for any $\vec{j}$ and $\vec{j}'$ since H_{e_r} does not contain $\vec{g}_{e_r}$. Therefore (38) holds.

C Relation between BPQM and classical BP

While the BPQM algorithm and classical BP both allow for decoding classical information by passing around certain messages in a graph, at first glance this might seem to be the only similarity between the two algorithms. In this section we show that this is not the case. More precisely, we show that classical belief propagation decoding on the binary symmetric channel (BSC) can be expressed in a language very similar to BPQM, which clearly illustrates how BPQM can be considered a quantum analog of BP.

Consider the task of decoding a single codeword bit $X_r, r \in \{1, \ldots, n\}$ given the noisy channel outputs. Classical BP decoding can be described as a message-passing algorithm on the Tanner

graph with added nodes for the channel output, as depicted in Fig. 3c. Alternatively, BP can also be expressed as a message-passing algorithm on the Forney-style representation of that same graph [6]. One can expand this factor graph as described in Fig. 7 such that it only contains equality and check nodes of degree 3. This is possible as only the marginalization of the random variable X_r is of interest. When the resulting graph is a tree, we can regard BP as operating on the the same MPG as BPQM. The messages that BP passes across the edges of the MPG are real numbers l representing log-likelihood ratios. The initial messages generated by the channel nodes are the log-likelihood ratios $l = \log \frac{P_{Y|X}(y|0)}{P_{Y|X}(y|1)}$ where $P_{Y|X}$ describes the BSC and y is the observed channel output. Every equality node operation consists of summing the two incoming messages l_1, l_2,

$$l_1, l_2 \mapsto l_1 + l_2 \,. \tag{75}$$

and every check node operation consists of the following operation:

$$l_1, l_2 \mapsto 2 \tanh^{-1} \left(\tanh \frac{l_1}{2} \tanh \frac{l_2}{2} \right) . \tag{76}$$

If the final message l generated at the root is greater than zero, then the decoder output is 0. If l is smaller than zero, then the decoder output is 1.

To recognize the analogy between BP and BPQM, we separate the messages $l \in \mathbb{R}$ passed over the edge e in BP into two parts $l = (-1)^b c$, where $b \in \{0, 1\}$ and $c = |l|$. Intuitively, one can think of b being an estimate for the value of the random variable corresponding of the edge e, i.e., it represents the most likely value that this random variable has given the information from all preceding nodes in the MPG. On the other hand, c can be intuitively thought of as the reliability, i.e., how likely it is that the estimate b is correct. In fact, b turns out to be the classical analog of the data qubit passed over the edge e in BPQM and c plays the role analogous to the angle information in BPQM. So in message-passing BPQM, the data and angle parts of the messages exactly correspond to b and c in the classical case. In the language of b and c the equailty node operations of BP become

$$(b_1, c_1), (b_2, c_2) \mapsto \Big(\Theta\big((-1)^{b_1} c_1 + (-1)^{b_2} c_2\big), \big|(-1)^{b_1} c_1 + (-1)^{b_2} c_2\big| \Big) \tag{77}$$

where $\Theta(x)$ denotes the Heaviside function and and the check node operations become

$$(b_1, c_1), (b_2, c_2) \mapsto (b_1 \oplus b_2, 2 \tanh^{-1} \left(\tanh \frac{c_1}{2} \tanh \frac{c_2}{2} \right)) \tag{78}$$

where $\oplus$ denotes addition modulo 2. Note that the $\oplus$ operation is reminiscent to the CNOT operation used in BPQM.

The full similarity between BP and BPQM is unveiled by considering the effect of the BP node operations upon the state of the channel outputs. A contraction argument analogous to the one introduced in Section 3.2 then allows the resulting factor graph to be simplified to a form with a single channel node. More precisely, the channel from X_r to $Y_1, \ldots, Y_n$ is now a classical channel, as the channel outputs $Y_1, \ldots, Y_n$ are now classical systems. The factor graph representing the channel from X_1 to $Y_1, \ldots, Y_n$ for the 5-bit code is depicted in Fig. 19. Here BSC$[c]$ denotes the BSC with crossover probability corresponding to the reliability c, i.e., such that the log-likelihood ratio corresponding to the two possible outputs are $l = \pm c$.

By consulting the relations in (77) and (78), one can devise a set of two contraction identities depicted in Fig. 20. Then, by applying the corresponding node operations, one can iteratively simplify the factor graph as in BPQM. Note that the equality node produces two output bits: The first bit D, obtained as in (77), is the data bit. The second is the ancilla bit A which is simply defined as the parity $b_1 \oplus b_2$ of the two input bits b_1, b_2. Analogously to the equality node in BPQM which requires information about the qubit angles, the equality node operation in BP requires information about the reliabilities c_1, c_2 of the two input bits, and therefore it must be controlled by ancilla bits produced by previous node operations. But there is an important difference here: These ancilla bits are not produced by preceding check nodes, as in BPQM, but rather by preceding equality nodes. Of course, instead of conditioning equality nodes on all previous equality node ancillas, one can instead keep track of the reliability c on-the-fly to obtain a true message-passing algorithm. This is completely analogous to the on-the-fly bookkeeping of the angle in message-passing BPQM in Section 5, and in fact this is how BP is actually run in practice.

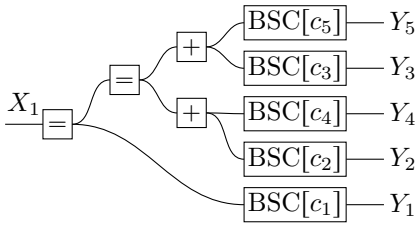


Figure 19: Factor graph describing the classical channel from X_1 to the channel outputs Y_1, Y_2, Y_3, Y_4, Y_5 for the 5-bit code.

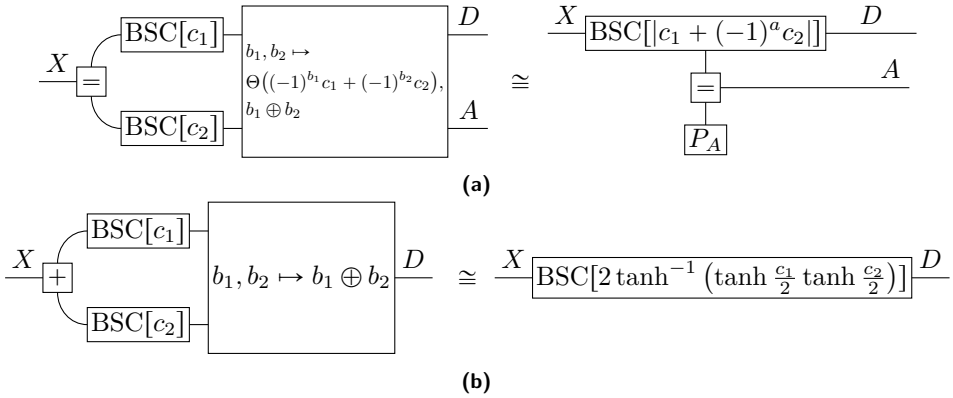


Figure 20: Graphical depiction of equality node (a) and check node (b) contraction identities. These identities are the classical counterpart to those depicted in Figs. 9 and 10. The second output bit A of the large box in figure (a) is necessary in order to make the computation reversible and thus provide information about the reliability to subsequent node operations (analogous to the check node ancilla Q_2 in Fig. 10). The probability distribution P_A is defined as $P_A(0) = 1/(\exp(c_1) + \exp(c_2) + \exp(c_1 + c_2))$ and $P_A(1) = 1 - P_A(0)$.

D Gate decomposition of equality node unitary

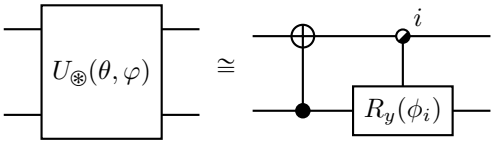
In this appendix, we derive the quantum circuit identity depicted in Fig. 14a. Recall, that the matrix representation of the variable node unitary in (25) is given by

$$U_{\circledast}(\theta,\varphi) = \begin{pmatrix} a_+ & 0 & 0 & a_- \\ -a_- & 0 & 0 & a_+ \\ 0 & b_- & b_+ & 0 \\ 0 & b_+ & -b_- & 0 \end{pmatrix}, \tag{79}$$

for $a_\pm$ and $b_\pm$ functions of θ and φ as in (26) and (27). Applying a CNOT gate controlled by the second qubit (which we denote as NOTC) produces the block diagonal form

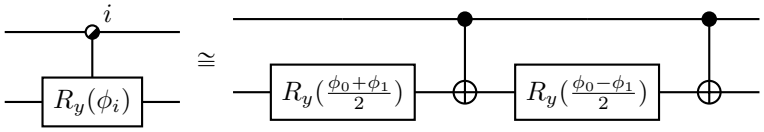
$$U_{\circledast}(\theta,\varphi) = \begin{pmatrix} a_+ & a_- & 0 & 0 \\ -a_- & a_+ & 0 & 0 \\ 0 & 0 & b_+ & b_- \\ 0 & 0 & -b_- & b_+ \end{pmatrix} \cdot \text{NOTC} = (|0\rangle\langle 0| \otimes A + |1\rangle\langle 1| \otimes B) \cdot \text{NOTC} \tag{80}$$

where $A := \begin{pmatrix} a_+ & a_- \\ -a_- & a_+ \end{pmatrix}$ and $B := \begin{pmatrix} b_+ & b_- \\ -b_- & b_+ \end{pmatrix}$. The blocks A and B are both R_y rotations. Denoting $R_y(\phi) = e^{-\frac{1}{2}i\phi\sigma_y}$, we have $A = R_y(\phi_0)$ for $\phi_0 := -2\arccos(a_+)$ and $B = R_y(\phi_1)$ for $\phi_1 := -2\arccos(b_+)$. Expressed as a circuit, this is



(81)

The desired identity then immediately follows from



(82)

E Proof of Lemma 5.1

Similarly to the proof of Lemma 4.4, we prove a more general statement about the intermediate states of the message-passing BPQM algorithm. This will directly imply the desired theorem.

Consider the BPQM algorithm executed on the MPG G w.r.t. the codeword bit X_r. As shown in Lemma B.1, the joint state of the qubit passed over some edge e together with all ancilla qubits produced by nodes preceding e can be written as

$$\sum_{\vec{j}\in\{0,1\}^{k_e-1}} \gamma_{e,\vec{j}} \, |Q(z_e,\theta_{e,\vec{j}})\rangle_{D_e} \otimes |0^{n_e-k_e}\rangle_{Z_e} \otimes |\vec{j}\rangle_{A_e}, \tag{83}$$

for some $\gamma_{e,\vec{j}} \in \mathbb{R}, z_e \in \{0,1\}, \theta_{e,\vec{j}} \in (0,\pi)$. Here n_e denotes the number of channel nodes preceding e and $k_e - 1$ denotes the number of check nodes preceding e. The system D_e describes the data qubit passed over the edge e, Z_e contains all ancilla qubits produced by equality nodes preceding e and A_e contains all ancilla qubits produced by check nodes preceding e.

In the course of the proof, we will inductively show the following statement for message-passing BPQM: The joint state of the message sent over the edge e together with all qubits stored in preceding check and variable nodes can be written in the form

$$\sum_{\vec{j}\in\{0,1\}^{k_e-1}} \tilde{\gamma}_{e,\vec{j}} \, |\tilde{\varphi}_{e,\vec{j}}\rangle_{D_eZ_e} \otimes |\vec{j}\rangle_{A_e} \otimes |\lightning(\tilde{c}_{e,\vec{j}})\rangle_{C_e} \otimes |\tilde{s}_{e,\vec{j}}\rangle_{S_e} \tag{84}$$

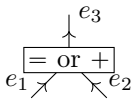
for some $\tilde{\gamma}_{e,\vec{j}} \in \mathbb{R}$, $(n_e - k_e + 1)$-qubit states $|\tilde{\varphi}_{e,\vec{j}}\rangle$, $\tilde{c}_{e,\vec{j}} \in \mathcal{A}_B$ and $(n_e - 1)2B$-qubit states $|\tilde{s}_{e,\vec{j}}\rangle$. The message passed over the edge e consists of the data part D_e as well as the angle part C_e. We denote by S_e the system of the remaining qubits produced by nodes preceeding e and that were not passed to the subsequent node.

Note that the statevectors in (83) and (84) are properly normalized, implying that $\sum_{\vec{j}} \gamma^2_{e,\vec{j}} = \sum_{\vec{j}} \tilde{\gamma}^2_{e,\vec{j}} = 1$.

Furthermore, this intermediate state for any edge e fulfills the following properties:

1. $\sum\limits_{\vec{j}} \gamma^2_{e,\vec{j}} |\cos\theta_{e,\vec{j}} - \tilde{c}_{e,\vec{j}}| \leqslant (2^{n_e+1} - 3)\delta$,
2. $\sum\limits_{\vec{j}} |\gamma^2_{e,\vec{j}} - \tilde{\gamma}^2_{e,\vec{j}}| \leqslant 2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}26^{n_e}$,
3. $\sum\limits_{\vec{j}} \gamma^2_{e,\vec{j}} \left\| |Q(z_e, \theta_{e,\vec{j}})\rangle \otimes |0^{n_e-k_e}\rangle - |\tilde{\varphi}_{e,\vec{j}}\rangle \right\| \leqslant 2^{n+1/2}\pi\sqrt{\delta}26^{n_e}$.

We prove our statement in an inductive manner. It is trivially true for the edges e coming out of a leaf node. So it suffices to show the following statement: If two edges e_1, e_2 which lead into the same check or equality node fulfill the desired properties



(85)

then the output edge e_3 also fulfills the desired properties. For this purpose, we deal separately with the two cases where the node is an equality or check node. In each case we will prove that the intermediate state at e_3 is of the correct form and that it fulfills the three desired properties.

For simplicity of notation, let us write the joint state of the data sent over the edges e_1, e_2 and the corresponding ancillas produced by preceding nodes as

$$\left(\sum_{\vec{j}_1 \in \{0,1\}^{k_1-1}} \gamma_{1,\vec{j}_1} |Q(z_1, \theta_{1,\vec{j}_1})\rangle_{D_1} |0^{n_1-k_1}\rangle_{Z_1} |\vec{j}_1\rangle_{A_1}\right) \otimes \left(\sum_{\vec{j}_2 \in \{0,1\}^{k_2-1}} \gamma_{2,\vec{j}_2} |Q(z_2, \theta_{2,\vec{j}_2})\rangle_{D_2} |0^{n_2-k_2}\rangle_{Z_2} |\vec{j}_2\rangle_{A_2}\right) \tag{86}$$

for BPQM and as

$$\left(\sum_{\vec{j}_1 \in \{0,1\}^{k_1-1}} \tilde{\gamma}_{1,\vec{j}_1} |\tilde{\varphi}_{1,\vec{j}_1}\rangle_{D_1 Z_1} \otimes |\vec{j}_1\rangle_{A_1} \otimes |\measuredangle(\tilde{c}_{1,\vec{j}_1})\rangle_{C_1} \otimes |\tilde{s}_{1,\vec{j}_1}\rangle_{S_1}\right) \otimes \left(\sum_{\vec{j}_2 \in \{0,1\}^{k_2-1}} \tilde{\gamma}_{2,\vec{j}_2} |\tilde{\varphi}_{2,\vec{j}_2}\rangle_{D_2 Z_2} \otimes |\vec{j}_2\rangle_{A_2} \otimes |\measuredangle(\tilde{c}_{2,\vec{j}_2})\rangle_{C_2} \otimes |\tilde{s}_{2,\vec{j}_2}\rangle_{S_2}\right) \tag{87}$$

for message-passing BPQM.

E.1 Check node

In the original BPQM description, the check node merely consists of a $\text{CNOT}_{D_1 D_2}$ gate on the qubits D_1 and D_2, therefore by using (39) the resulting state is given by

$$\sum_{\vec{j}_1, \vec{j}_2, l \in \{0,1\}} \gamma_{1,\vec{j}_1} \gamma_{2,\vec{j}_2} \kappa_{\vec{j}_1,\vec{j}_2,l} |Q(z_1 \oplus z_2, \theta_{1,\vec{j}_1} \circledast_l \theta_{2,\vec{j}_2})\rangle_{D_3} \otimes |0^{n_3-k_3}\rangle_{Z_3} \otimes \left(|\vec{j}_1\rangle |\vec{j}_2\rangle |l\rangle\right)_{A_3} \tag{88}$$

where $\kappa_{\vec{j}_1,\vec{j}_2,l} := (-1)^{l \cdot z_2} \frac{1}{\sqrt{2}} \sqrt{1 + (-1)^l \cos(\theta_{1,\vec{j}_1}) \cos(\theta_{2,\vec{j}_2})}$. Here we use $n_3 := n_1 + n_2$ and $k_3 := k_1 + k_2$ and introduced the new systems $D_3 = D_1$, $Z_3 = Z_1 Z_2$ and $A_3 = A_1 A_2 D_2$.

In a similar fashion, we now describe how the input state evolves in the message-passing picture. First we apply the $\text{CNOT}_{D_1D_2}$ gate. We write the effect on the state $|\varphi_{1,\vec{j}_1}\rangle_{D_1Z_1} \otimes |\varphi_{2,\vec{j}_2}\rangle_{D_2Z_2}$ as

$$\text{CNOT}_{D_1D_2} |\tilde{\varphi}_{1,\vec{j}_1}\rangle_{D_1Z_1} \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle_{D_2Z_2} = \tilde{\kappa}_{\vec{j}_1,\vec{j}_2,0} |\xi_{\vec{j}_1,\vec{j}_2,0}\rangle_{D_1Z_1Z_2} |0\rangle_{D_2} + \tilde{\kappa}_{\vec{j}_1,\vec{j}_2,1} |\xi_{\vec{j}_1,\vec{j}_2,1}\rangle_{D_1Z_1Z_2} |1\rangle_{D_2} , \tag{89}$$

for

$$\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l} := (-1)^{l\cdot z_2} \left\| P_l \cdot \text{CNOT}_{D_1D_2} \cdot |\tilde{\varphi}_{1,\vec{j}_1}\rangle_{D_1Z_1} \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle_{D_2Z_2} \right\| \in \mathbb{R} \tag{90}$$

and

$$|\xi_{\vec{j}_1,\vec{j}_2,l}\rangle := \frac{1}{\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}} P_l \cdot \text{CNOT}_{D_1D_2} \cdot |\tilde{\varphi}_{1,\vec{j}_1}\rangle_{D_1Z_1} \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle_{D_2Z_2} , \tag{91}$$

where P_l is the isometry $\mathbb{1}_{D_1Z_1Z_2} \otimes \langle l|_{D_2}$ giving the amplitude in the subspace where the target qubit of the CNOT gate is in the state $|l\rangle$ for $l \in \{0,1\}$. Considering the classical computation of the output angle, the resulting final state will thus be

$$\sum_{\vec{j}_1,\vec{j}_2,l} \tilde{\gamma}_{1,\vec{j}_1} \tilde{\gamma}_{2,\vec{j}_2} \tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l} |\xi_{\vec{j}_1,\vec{j}_2,l}\rangle_{D_3Z_3} \otimes (|\vec{j}_1\rangle |\vec{j}_2\rangle |l\rangle)_{A_3} \otimes |\measuredangle(R_B \left(\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1 + (-1)^l \tilde{c}_{1,\vec{j}_1} \tilde{c}_{2,\vec{j}_2}} \right))\rangle_{C_3}$$
$$\otimes \left(|\tilde{s}_{1,\vec{j}_1}\rangle |\tilde{s}_{2,\vec{j}_2}\rangle |\measuredangle(\tilde{c}_{1,\vec{j}_1})\rangle |\measuredangle(\tilde{c}_{2,\vec{j}_2})\rangle \right)_{S_3} \tag{92}$$

where S_3 contains S_1, S_2 and the two B-qubit registers consumed by the node operation (in accordance with Fig. 13).

Before we prove the three statements of the lemma, we show that the summed square difference between $\kappa_{\vec{j}_1,\vec{j}_2,l}$ and $\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}$ can be bounded:

$$\sum_l \left| \kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l} \right|$$
$$\leqslant \sum_l 2 \left| \left| \kappa_{\vec{j}_1,\vec{j}_2,l} \right| - \left| \tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l} \right| \right| \tag{93}$$
$$= 2 \sum_l \left| \left\| P_l \cdot \text{CNOT} \cdot |\tilde{\varphi}_{1,\vec{j}_1}\rangle \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle \right\| - \left\| P_l \cdot \text{CNOT} \cdot |Q(z_1, \theta_{1,\vec{j}_1})\rangle |0^{n_1-k_1}\rangle \otimes |Q(z_2, \theta_{2,\vec{j}_2})\rangle |0^{n_2-k_2}\rangle \right\| \right| \tag{94}$$
$$\leqslant 2 \sum_l \left\| P_l \cdot \text{CNOT} \cdot \left(|\tilde{\varphi}_{1,\vec{j}_1}\rangle \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle - |Q(z_1, \theta_{1,\vec{j}_1})\rangle \otimes |0^{n_1-k_1}\rangle \otimes |Q(z_2, \theta_{2,\vec{j}_2})\rangle \otimes |0^{n_2-k_2}\rangle \right) \right\| \tag{95}$$
$$\leqslant 4 \left\| |\tilde{\varphi}_{1,\vec{j}_1}\rangle \otimes |\tilde{\varphi}_{2,\vec{j}_2}\rangle - |Q(z_1, \theta_{1,\vec{j}_1})\rangle \otimes |0^{n_1-k_1}\rangle \otimes |Q(z_2, \theta_{2,\vec{j}_2})\rangle \otimes |0^{n_2-k_2}\rangle \right\| \tag{96}$$
$$\leqslant 4 \left\| |\tilde{\varphi}_{1,\vec{j}_1}\rangle - |Q(z_1, \theta_{1,\vec{j}_1})\rangle \otimes |0^{n_1-k_1}\rangle \right\| + 4 \left\| |\tilde{\varphi}_{2,\vec{j}_2}\rangle - |Q(z_2, \theta_{2,\vec{j}_2})\rangle \otimes |0^{n_2-k_2}\rangle \right\| , \tag{97}$$

where the first inequality follows from $|\kappa_{\vec{j}_1,\vec{j}_2,l}|, |\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}| \leqslant 1$ and the Lipschitz continuity of the function $x \mapsto x^2$ on the interval $[-1,1]$ as well as the fact that $\kappa_{\vec{j}_1,\vec{j}_2,l}$ and $\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}$ have the identical sign. The second inequality follows from the reverse triangle inequality, the third inequality follows because the operator norm of $P_l \cdot \text{CNOT}$ is 1, and the last inequality follows from the triangle inequality and the fact that

$$\begin{aligned} \| |f_1\rangle \otimes |g_1\rangle - |f_2\rangle \otimes |g_2\rangle \| &= \| |f_1\rangle \otimes |g_1\rangle - |f_2\rangle \otimes (|g_1\rangle + (|g_2\rangle - |g_1\rangle)) \| \\ &= \| (|f_1\rangle - |f_2\rangle) \otimes |g_1\rangle + |f_2\rangle \otimes (|g_2\rangle - |g_1\rangle) \| \\ &\leqslant \| |f_1\rangle - |f_2\rangle \| \| |g_1\rangle \| + \| |f_2\rangle \| \| |g_1\rangle - |g_2\rangle \| \\ &= \| |f_1\rangle - |f_2\rangle \| + \| |g_1\rangle - |g_2\rangle \| \end{aligned} \tag{98}$$

for any normalized vectors $|f_1\rangle, |f_2\rangle, |g_1\rangle |g_2\rangle$. We now prove the three statements of the lemma.

E.1.1 Property 1

$$\sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1} \gamma^2_{2,\vec{j}_2} \kappa^2_{\vec{j}_1,\vec{j}_2,l} \cdot \left| \cos(\theta_{1,\vec{j}_1} \circledast_l \theta_{2,\vec{j}_2}) - R_B \left(\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1 + (-1)^l \tilde{c}_{1,\vec{j}_1} \tilde{c}_{2,\vec{j}_2}} \right) \right| \tag{99}$$

$$\leqslant \delta + \sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l} \cdot \left|\cos(\theta_{1,\vec{j}_1} \circledast_l \theta_{2,\vec{j}_2}) - \frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}\right| \tag{100}$$

$$= \delta + \sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \cdot \left|\frac{\left(\cos\theta_{1,\vec{j}_1} + (-1)^l \cos\theta_{2,\vec{j}_2}\right)}{2} - \kappa^2_{\vec{j}_1,\vec{j}_2,l}\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}\right| \tag{101}$$

$$\leqslant \delta + \sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \cdot \left|\frac{\left(\cos\theta_{1,\vec{j}_1} + (-1)^l \cos\theta_{2,\vec{j}_2}\right)}{2} - \frac{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}{2}\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}\right| +$$
$$\sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \cdot \left|\frac{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}{2}\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}} - \kappa^2_{\vec{j}_1,\vec{j}_2,l}\frac{\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}}{1+(-1)^l \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}\right| \tag{102}$$

$$\leqslant \delta + \frac{1}{2}\sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\left(\cos\theta_{1,\vec{j}_1} + (-1)^l \cos\theta_{2,\vec{j}_2}\right) - \left(\tilde{c}_{1,\vec{j}_1} + (-1)^l \tilde{c}_{2,\vec{j}_2}\right)\right| +$$
$$\frac{1}{2}\sum_{\vec{j}_1,\vec{j}_2,l} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\cos\theta_{1,\vec{j}_1}\cos\theta_{2,\vec{j}_2} - \tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}\right| \tag{103}$$

Here we used that $\left|\frac{\tilde{c}_{1,\vec{j}_1}+(-1)^l\tilde{c}_{2,\vec{j}_2}}{1+(-1)^l\tilde{c}_{1,\vec{j}_1}\tilde{c}_{2,\vec{j}_2}}\right| \leqslant 1$.

$$\leqslant \delta + \sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}\right| + \sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}\right| +$$
$$\sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}\right| + \sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \left|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}\right| \tag{104}$$

$$\leqslant \delta + 2\left((2^{n_1+1}-3) + (2^{n_2+1}-3)\right)\delta = \delta\left(1 + 4(2^{n_1} + 2^{n-n_1}) - 12\right) \tag{105}$$

By using that $2^{n_1} + 2^{n_3-n_1}$ for $n_1 \in \{1, 2, \ldots, n_3 - 1\}$ is maximized by $n_1 = n_3 - 1$ we obtain that this term is bounded above by

$$\delta\left(1 + 2^{n_3+1} + 8 - 12\right) = \delta\left(2^{n_3+1} - 3\right). \tag{106}$$

Note that the above derivation of property 1 is precisely the reason why Lemma 5.1 is formulated in a probability-averaged manner instead of bounding the discretisation errors in every branch of the wavefunction separately.

E.1.2 Property 2

$$\sum_{\vec{j}_1,\vec{j}_2,l} \left|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\right|$$
$$= \sum_{\vec{j}_1,\vec{j}_2,l} \left|\left(\gamma^2_{1,\vec{j}_1} - \tilde{\gamma}^2_{1,\vec{j}_1}\right)\gamma^2_{2,\vec{j}_2}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l} + \left(\gamma^2_{2,\vec{j}_2} - \tilde{\gamma}^2_{2,\vec{j}_2}\right)\tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l} + \left(\kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\right)\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\right| \tag{107}$$

$$\leqslant \sum_{\vec{j}_1,\vec{j}_2,l} \left|\gamma^2_{1,\vec{j}_1} - \tilde{\gamma}^2_{1,\vec{j}_1}\right|\gamma^2_{2,\vec{j}_2}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l} + \left|\gamma^2_{2,\vec{j}_2} - \tilde{\gamma}^2_{2,\vec{j}_2}\right|\tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l} + \left|\kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\right|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \tag{108}$$

$$= \sum_{\vec{j}_1}\left|\gamma^2_{1,\vec{j}_1} - \tilde{\gamma}^2_{1,\vec{j}_1}\right| + \sum_{\vec{j}_2}\left|\gamma^2_{2,\vec{j}_2} - \tilde{\gamma}^2_{2,\vec{j}_2}\right| + \sum_{\vec{j}_1,\vec{j}_2,l}\left|\kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\right|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} \tag{109}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}26^{n_1} + 2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}26^{n_2} + 4\sum_{\vec{j}_1}\gamma^2_{1,\vec{j}_1}\left\||\tilde{\varphi}_{1,\vec{j}_1}\rangle - |Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|\vec{j}_1\rangle\right\|$$
$$+ 4\sum_{\vec{j}_2}\gamma^2_{2,\vec{j}_2}\left\||\tilde{\varphi}_{2,\vec{j}_2}\rangle - |Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|\vec{j}_2\rangle\right\|, \tag{110}$$

where we used the bound on the difference between $\kappa^2_{\vec{j}_1,\vec{j}_2,l}$ and $\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}$ shown previously. We thus obtain the upper bound

$$2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}\left(26^{n_1} + 26^{n_2} + 12\cdot 26^{n_1} + 12\cdot 26^{n_2}\right)$$
$$= 2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}\left(13\cdot 26^{n_1} + 13\cdot 26^{n_2}\right) \tag{111}$$
$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\frac{1}{3}\left(13\cdot 26^{n-1} + 13\cdot 26^{n-1}\right) \tag{112}$$
$$= 2^{n+1/2}\pi\sqrt{\delta}\tfrac{1}{3}26^{n}\,. \tag{113}$$

E.1.3 Property 3

$$\sum_{\vec{j}_1,\vec{j}_2,l}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l}\left\||Q(z_1\oplus z_2,\theta_{1,\vec{j}_1}\circledast_l\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_1-k_1}\rangle|0^{n_2k-2}\rangle - |\xi_{\vec{j}_1,\vec{j}_2,l}\rangle\right\|$$
$$= \sum_{\vec{j}_1,\vec{j}_2,l}\left\|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l}\,|Q(z_1\oplus z_2,\theta_{1,\vec{j}_1}\circledast_l\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_1-k_1}\rangle|0^{n_2-k_2}\rangle - \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l}\,|\xi_{\vec{j}_1,\vec{j}_2,l}\rangle\right\| \tag{114}$$
$$\leqslant \sum_{\vec{j}_1,\vec{j}_2,l}\left\|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l}\,|Q(z_1\oplus z_2,\theta_{1,\vec{j}_1}\circledast_l\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_1-k_1}\rangle|0^{n_2-k_2}\rangle - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\,|\xi_{\vec{j}_1,\vec{j}_2,l}\rangle\right\|$$
$$+ \sum_{\vec{j}_1,\vec{j}_2,l}\left|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l} - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\tilde{\kappa}^2_{\vec{j}_1,\vec{j}_2,l}\right|. \tag{115}$$

The second sum in (115) is upper bounded by $2^{n+1/2}\pi\sqrt{\delta}(\frac{13}{3}26^{n_1} + \frac{13}{3}26^{n_2})$ by point 2. For the first term we use

$$\kappa_{\vec{j}_1,\vec{j}_2,l}\,|Q(z_1\oplus z_2,\theta_{1,\vec{j}_1}\circledast_l\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_1-k_1}\rangle|0^{n_2-k_2}\rangle$$
$$= P_l\cdot\textsc{cnot}\cdot|Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|0^{n_1-k_1}\rangle\otimes|Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_2-k_2}\rangle \tag{116}$$

and

$$\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}\,|\xi_{\vec{j}_1,\vec{j}_2,l}\rangle = P_l\cdot\textsc{cnot}\cdot|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle \tag{117}$$

to obtain the upper bound

$$\sum_{\vec{j}_1,\vec{j}_2,l}\left\|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa_{\vec{j}_1,\vec{j}_2,l}\,|Q(z_1,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z_2,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\| \tag{118}$$

because the operator norm of $P_l\textsc{cnot}$ is 1. By the triangle inquality, this can be further upper bounded by

$$\sum_{\vec{j}_1,\vec{j}_2,l}|\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}|\left\|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\,|Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|0^{n_1-k_1}\rangle\otimes|Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_2-k_2}\rangle - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\|$$
$$+ \sum_{\vec{j}_1,\vec{j}_2,l}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\cdot|\kappa_{\vec{j}_1,\vec{j}_2,l} - \tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}| \tag{119}$$
$$\leqslant 2\sum_{\vec{j}_1,\vec{j}_2}\left\|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\,|Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|0^{n_1-k_1}\rangle\otimes|Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_2-k_2}\rangle - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\|$$

$$+2\sum_{\vec{j}_1,\vec{j}_2}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\cdot\left(\left\||\tilde{\varphi}_{1,\vec{j}_1}\rangle-|Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|\vec{j}_1\rangle\right\|+\left\||\tilde{\varphi}_{2,\vec{j}_2}\rangle-|Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|\vec{j}_2\rangle\right\|,\right) \tag{120}$$

where we used the previously shown bound on the difference

$$||\kappa_{\vec{j}_1,\vec{j}_2,l}|-|\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}||=|\kappa_{\vec{j}_1,\vec{j}_2,l}-\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}|\,. \tag{121}$$

By the induction step, the second term in (120) is less than $2^{n+1/2}\pi\sqrt{\delta}(2\cdot 26^{n_1}+2\cdot 26^{n_2})$. The first term can be bounded by

$$\leqslant 2\sum_{\vec{j}_1,\vec{j}_2}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left\||Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|0^{n_1-k_1}\rangle\otimes|Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_2-k_2}\rangle-|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\|$$
$$+2\sum_{\vec{j}_1,\vec{j}_2}|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}-\tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}| \tag{122}$$

$$\leqslant 2\sum_{\vec{j}_1}\gamma^2_{1,\vec{j}_1}\left\||Q(z_1,\theta_{1,\vec{j}_1})\rangle\otimes|0^{n_1-k_1}\rangle-|\tilde{\varphi}_{1,\vec{j}_1}\rangle\right\|+2\sum_{\vec{j}_2}\gamma^2_{2,\vec{j}_2}\left\||Q(z_2,\theta_{2,\vec{j}_2})\rangle\otimes|0^{n_2-k_2}\rangle-|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\|$$
$$+2\sum_{\vec{j}_1}|\gamma^2_{1,\vec{j}_1}-\tilde{\gamma}^2_{1,\vec{j}_1}|+2\sum_{\vec{j}_2}|\gamma^2_{2,\vec{j}_2}-\tilde{\gamma}^2_{2,\vec{j}_2}| \tag{123}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\left(2\cdot 26^{n_1}+2\cdot 26^{n_2}+\frac{2}{3}26^{n_1}+\frac{2}{3}26^{n_2}\right). \tag{124}$$

To summarize, we have found that

$$\sum_{\vec{j}_1,\vec{j}_2,l}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\kappa^2_{\vec{j}_1,\vec{j}_2,l}\left\||Q(z_1\oplus z_2,\theta_{1,\vec{j}_1}\circledast_l\theta_{2,\vec{j}_2})\rangle\otimes|\vec{j}_1\rangle|\vec{j}_2\rangle\otimes|l\rangle-|\xi_{\vec{j}_1,\vec{j}_2,l}\rangle\otimes|l\rangle\right\| \tag{125}$$
$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\left(\frac{13}{3}26^{n_1}+\frac{13}{3}26^{n_1}+2\cdot 26^{n_1}+2\cdot 26^{n_2}+2\cdot 26^{n_1}+2\cdot 26^{n_2}+\frac{2}{3}26^{n_1}+\frac{2}{3}26^{n_2}\right) \tag{126}$$
$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\left(9\cdot 26^{n_1}+9\cdot 26^{n_2}\right) \tag{127}$$
$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\left(13\cdot 26^{n_3-1}+13\cdot 26^{n_3-1}\right) \tag{128}$$
$$=2^{n+1/2}\pi\sqrt{\delta}\cdot 26^{n_3}\,. \tag{129}$$

E.2 Equality node

In the original BPQM description, the equality node consists of a uniformly-controlled $U_{\circledast}$ gate which transforms our state to

$$\sum_{\vec{j}_1\in\{0,1\}^{n_1-1},\vec{j}_2\in\{0,1\}^{n_2-1}}\gamma_{1,\vec{j}_1}\gamma_{2,\vec{j}_2}\,|Q(z,\theta_{1,\vec{j}_1}\circledast\theta_{2,\vec{j}_2})\rangle_{D_3}\otimes|0^{n_3-k_3}\rangle_{Z_3}\otimes\left(|\vec{j}_1\rangle|\vec{j}_2\rangle\right)_{A_3}, \tag{130}$$

where $n_3:=n_1+n_2$, $k_3:=k_1+k_2-1$ and $D_3=D_1$, $Z_3=Z_1Z_2$, $A_3=A_1A_2D_2$. Here we used the fact that $z_1=z_2=:z$, which is a direct consequence of claim 1 in the proof of Lemma 4.4. In the message-passing variant, the resulting state is

$$\sum_{\vec{j}_1\in\{0,1\}^{n_1-1},\vec{j}_2\in\{0,1\}^{n_2-1}}\tilde{\gamma}_{1,\vec{j}_1}\tilde{\gamma}_{2,\vec{j}_2}\left(V_{\vec{j}_1,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right)_{D_3Z_3}\otimes\left(|\vec{j}_1\rangle|\vec{j}_2\rangle\right)_{A_3}\otimes|Q(\tilde{c}_{1,\vec{j}_1}\cdot\tilde{c}_{2,\vec{j}_2})\rangle_{C_3}$$
$$\otimes\left(|\tilde{s}_{1,\vec{j}_1}\rangle|\tilde{s}_{2,\vec{j}_2}\rangle|\divideontimes(\tilde{c}_{1,\vec{j}_1})\rangle|\divideontimes(\tilde{c}_{2,\vec{j}_2})\rangle\right)_{S_3} \tag{131}$$

where $V_{\vec{j}_1,\vec{j}_2}$ describes the quantum circuit from the right-hand-side of Fig. 14a acting on the system D_1D_2 with the R_y rotation angles $\tilde{R}_B(\alpha(\tilde{\theta}_{1,\vec{j}_1},\tilde{\theta}_{2,\vec{j}_2}))$ and $\tilde{R}_B(\beta(\tilde{\theta}_{1,\vec{j}_1},\tilde{\theta}_{2,\vec{j}_2}))$. We now show that the three statements of the lemma hold in this case as well.

E.2.1 Property 1

In the first case we have

$$\sum_{\vec{j}_1,\vec{j}_2,k} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left|\cos(\theta_{1,\vec{j}_1} \circledast \theta_{2,\vec{j}_2}) - R_B(\tilde{c}_{1,\vec{j}_1}\cdot\tilde{c}_{2,\vec{j}_2})\right| \tag{132}$$

$$\leqslant \delta + \sum_{\vec{j}_1,\vec{j}_2,k} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left|\cos\theta_{1,\vec{j}_1}\cos\theta_{2,\vec{j}_2} - \tilde{c}_{1,\vec{j}_1}\cdot\tilde{c}_{2,\vec{j}_2}\right| \tag{133}$$

$$\leqslant \delta + \sum_{\vec{j}_1}\gamma^2_{1,\vec{j}_1}\left|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}\right| + \sum_{\vec{j}_2}\gamma^2_{2,\vec{j}_2}\left|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}\right| \tag{134}$$

$$\leqslant \delta + (2^{n_1+1}-3)\delta + (2^{n_2+1}-3)\delta \tag{135}$$

$$= \delta(2^{n_1+1} + 2^{n_2+1} - 5) \tag{136}$$

$$\leqslant \delta(2^{n_3} + 2^{n_3} - 3) \tag{137}$$

$$= (2^{n_3+1}-3)\delta\,. \tag{138}$$

E.2.2 Property 2

The second case is straightforward:

$$\sum_{\vec{j}_1,\vec{j}_2}|\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2} - \tilde{\gamma}^2_{1,\vec{j}_1}\tilde{\gamma}^2_{2,\vec{j}_2}| \leqslant \sum_{\vec{j}_1}|\gamma^2_{1,\vec{j}_1} - \tilde{\gamma}^2_{1,\vec{j}_1}| + \sum_{\vec{j}_2}|\gamma^2_{2,\vec{j}_2} - \tilde{\gamma}^2_{2,\vec{j}_2}| \tag{139}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\tfrac{1}{3}(26^{n_1} + 26^{n_2}) \tag{140}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\tfrac{1}{3}(26^{n_3-1} + 26^{n_3-1}) \tag{141}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\tfrac{1}{3}26^{n_3}\,. \tag{142}$$

E.2.3 Property 3

Finally, for the third statement we use the triangle inequality to bound the error by a sum of three terms, which we then address individually.

$$\sum_{\vec{j}_1,\vec{j}_2,k}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left\|U_{\circledast}(\theta_{1,\vec{j}_1},\theta_{2,\vec{j}_2})\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle - V_{\vec{j}_1,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\| \tag{143}$$

$$\begin{aligned}
\leqslant \sum_{\vec{j}_1,\vec{j}_2}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}&\left\|V_{\vec{j}_1,\vec{j}_2}\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle - V_{\vec{j}_1,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle\right\| \\
&+ \sum_{\vec{j}_1,\vec{j}_2}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\|U_{\circledast}(\theta_{1,\vec{j}_1},\theta_{2,\vec{j}_2})\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle \\
&\qquad -U_{\circledast}(\tilde{\theta}_{1,\vec{j}_1},\tilde{\theta}_{2,\vec{j}_2})\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_2-k_2}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle\| + \\
&+ \sum_{\vec{j}_1,\vec{j}_2}\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\|U_{\circledast}(\tilde{\theta}_{1,\vec{j}_1},\tilde{\theta}_{2,\vec{j}_2})\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle \\
&\qquad -V_{\vec{j}_1,\vec{j}_2}\,|Q(z,\theta_{1,\vec{j}_1})\rangle|0^{n_1-k_1}\rangle\otimes|Q(z,\theta_{2,\vec{j}_2})\rangle|0^{n_2-k_2}\rangle\|
\end{aligned} \tag{144}$$

where we introduced $\tilde{\theta}_{i,\vec{j}_i} := \arccos\tilde{c}_{\theta_{i,\vec{j}_i}}$. The first term of (144) is simply the error of the incoming state. By removing $V_{\vec{j}_1,\vec{j}_2}$ from the expression, we see that it can be bounded by $2^{n+1/2}\pi\sqrt{\delta}(26^{n_1} + 26^{n_2})$.

The second term of (144) is the error caused by the fact that we are executing the wrong equality node unitary, due to the errornous angle stored in the angle register. We make use of following property for any set of angles $\phi_1, \phi_2, \tilde{\phi}_1, \tilde{\phi}_2$ and $w \in \{0,1\}$:

$$\left\|U_{\circledast}(\phi_1,\phi_2)\,|Q(w,\phi_1)\rangle|Q(w,\phi_2)\rangle - U_{\circledast}(\tilde{\phi}_1,\tilde{\phi}_2)\,|Q(w,\phi_1)\rangle|Q(w,\phi_2)\rangle\right\| \tag{145}$$

$$\leqslant \left\| U_{\circledast}(\phi_1,\phi_2)\,|Q(w,\phi_1)\rangle\,|Q(w,\phi_2)\rangle - U_{\circledast}(\tilde{\phi}_1,\tilde{\phi}_2)\,|Q(w,\tilde{\phi}_1)\rangle\,|Q(w,\tilde{\phi}_2)\rangle \right\|$$
$$+ \left\| U_{\circledast}(\tilde{\phi}_1,\tilde{\phi}_2)\,|Q(w,\tilde{\phi}_1)\rangle\,|Q(w,\tilde{\phi}_2)\rangle - U_{\circledast}(\tilde{\phi}_1,\tilde{\phi}_2)\,|Q(w,\phi_1)\rangle\,|Q(w,\phi_2)\rangle \right\| \tag{146}$$
$$\leqslant \left\| |Q(w,\phi_1 \circledast \phi_2)\rangle - |Q(w,\tilde{\phi}_1 \circledast \tilde{\phi}_2)\rangle \right\| + \left\| |Q(w,\phi_1)\rangle\,|Q(w,\phi_2)\rangle - |Q(w,\tilde{\phi}_1)\rangle\,|Q(w,\tilde{\phi}_2)\rangle \right\| \tag{147}$$
$$\leqslant \left\| |Q(w,\phi_1 \circledast \phi_2)\rangle - |Q(w,\tilde{\phi}_1 \circledast \tilde{\phi}_2)\rangle \right\| + \left\| |Q(w,\phi_1)\rangle - |Q(w,\tilde{\phi}_1)\rangle \right\| + \left\| |Q(w,\phi_2)\rangle - |Q(w,\tilde{\phi}_2)\rangle \right\| \tag{148}$$
$$\leqslant \tfrac{1}{2}\left(|\phi_1 \circledast \phi_2 - \tilde{\phi}_1 \circledast \tilde{\phi}_2| + |\phi_1 - \tilde{\phi}_1| + |\phi_2 - \tilde{\phi}_2| \right), \tag{149}$$

where we used $\| |Q(w,\alpha)\rangle - |Q(w,\beta)\rangle \| = 2|\sin\frac{\alpha-\beta}{4}| \leqslant \frac{|\alpha-\beta|}{2}$. Therefore, the second term can be bounded by

$$\frac{1}{2}\sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left(|\theta_{1,\vec{j}_1} \circledast \theta_{2,\vec{j}_2} - \arccos(\tilde{c}_{1,\vec{j}_1}\cdot\tilde{c}_{2,\vec{j}_2})| + |\theta_{1,\vec{j}_1} - \arccos\tilde{c}_{1,\vec{j}_1}| + |\theta_{2,\vec{j}_2} - \arccos\tilde{c}_{2,\vec{j}_2}| \right) \tag{150}$$
$$\leqslant \frac{\pi}{2\sqrt{2}}\sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left(\sqrt{|\cos\theta_{1,\vec{j}_1}\cdot\cos\theta_{2,\vec{j}_2} - \tilde{c}_{1,\vec{j}_1}\cdot\tilde{c}_{2,\vec{j}_2}|} + \sqrt{|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}|} + \sqrt{|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}|} \right) \tag{151}$$
$$\leqslant \frac{\pi}{\sqrt{2}}\sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left(\sqrt{|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}|} + \sqrt{|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}|} \right) \tag{152}$$
$$\leqslant \frac{\pi}{\sqrt{2}}\sum_{\vec{j}_1,\vec{j}_2} \sqrt{\gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}}\left(\sqrt{|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}|} + \sqrt{|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}|} \right), \tag{153}$$

which follows from the Hölder-continuity of the arccos function $|\arccos x - \arccos y| \leqslant \frac{\pi}{\sqrt{2}}\sqrt{|x-y|}$ as well as the subadditivity of the square root. A direct consequence of the Cauchy-Schwarz inequality is the statement $\|\vec{z}\|_1 \leqslant \sqrt{M}\,\|\vec{z}\|_2$ for $\vec{z} \in \mathbb{R}^M$, which allows us to upper bound the second term by

$$2^{n/2}\frac{\pi}{\sqrt{2}}\left(\sqrt{\sum_{\vec{j}_1}\gamma^2_{1,\vec{j}_1}|\cos\theta_{1,\vec{j}_1} - \tilde{c}_{1,\vec{j}_1}|} + \sqrt{\sum_{\vec{j}_2}\gamma^2_{2,\vec{j}_2}|\cos\theta_{2,\vec{j}_2} - \tilde{c}_{2,\vec{j}_2}|} \right) \tag{154}$$
$$\leqslant 2^{n/2}\frac{\pi}{\sqrt{2}}\sqrt{\delta}\left(\sqrt{2^{n_1+1}-3} + \sqrt{2^{n_2+1}-3} \right). \tag{155}$$

Finally, the third term of (144) entails the fact, that the unitary $V_{\vec{j}_1,\vec{j}_2}$ is an imperfect implementation of the unitary $U_{\circledast}(\tilde{\theta}_{1,\vec{j}_1},\tilde{\theta}_{2,\vec{j}_2})$ due to the discretization error in the computation of the angles α and β in the circuit of Fig. 14. Since we know that the discretization error in the computation of α,β is bounded by $\pi\delta$, this implies that the R_y gates implement a rotation by an angle that differs by at most $\pi\delta$ from the desired value. By using

$$\|R_y(x+\Delta) - R_y(x)\|_{\mathrm{op}} \leqslant \frac{|\Delta|}{2} \tag{156}$$

for the operator norm $\|\cdot\|_{\mathrm{op}}$ and any real numbers x,Δ, as well as the fact that

$$\|U_1U_2\ldots U_s - \tilde{U}_1\tilde{U}_2\ldots\tilde{U}_s\|_{\mathrm{op}} \leqslant \|U_1-\tilde{U}_1\|_{\mathrm{op}} + \|U_2-\tilde{U}_2\|_{\mathrm{op}} + \cdots + \|U_s-\tilde{U}_s\|_{\mathrm{op}} \tag{157}$$

for any sequences of unitaries $U_i,\tilde{U}_i$ of length s, we can upper bound the third term by

$$\sum_{\vec{j}_1,\vec{j}_2} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}(\tfrac{1}{2}\pi\delta + \tfrac{1}{2}\pi\delta) = \pi\delta\,. \tag{158}$$

To summarize, we have shown that

$$\sum_{\vec{j}_1,\vec{j}_2,k} \gamma^2_{1,\vec{j}_1}\gamma^2_{2,\vec{j}_2}\left\| U_{\circledast}(\theta_{1,\vec{j}_1},\theta_{2,\vec{j}_2})\,|Q(z,\theta_{1,\vec{j}_1})\rangle\,|0^{n_1-k_1}\rangle \otimes |Q(z,\theta_{2,\vec{j}_2})\rangle\,|0^{n_2-k_2}\rangle - V_{\vec{j}_1,\vec{j}_2}\,|\tilde{\varphi}_{1,\vec{j}_1}\rangle|\tilde{\varphi}_{2,\vec{j}_2}\rangle \right\| \tag{159}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}(26^{n_1}+26^{n_2})+2^{(n-1)/2}\pi\sqrt{\delta}\left(\sqrt{2^{n_1+1}-3}+\sqrt{2^{n_2+1}-3}\right)+\pi\delta \tag{160}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}(26^{n_1}+26^{n_2})+2^{(n-1)/2}\pi\sqrt{\delta}\left(\sqrt{2^{n}}+\sqrt{2^{n}}\right)+\pi\sqrt{\delta} \tag{161}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}\left(26^{n_3-1}+26^{n_3-1}+1+1\right) \tag{162}$$

$$\leqslant 2^{n+1/2}\pi\sqrt{\delta}26^{n_3}\,. \tag{163}$$

F Proof of Theorem 5.2

Directly comparing BPQM with message-passing BPQM is impeded by the somewhat technical difficulty that the two corresponding unitaries operate on a different number of qubits, since the message-passing BPQM requires additional ancilla B-qubit registers. For reasons purely related to the analysis of message-passing BPQM we now introduce a third variant of BPQM, which we call *extended BPQM*. Extended BPQM keeps track of the angles of the involved qubits in B-qubit quantum registers as in message-passing BPQM, and therefore it operates on the same number of qubits as message-passing BPQM. The only difference to the message-passing variant is that the equality node unitaries $U_{\circledast}$ do not make use of the angle registers, but rather condition on the check node ancilla qubits as in BPQM. Since the angle registers never influence the data registers, the decoding is equivalent to BPQM and is therefore optimal. We denote the BPQM unitaries to decode the codeword bit X_r by V_r^{mp} for the message-passing algorithm and V_r^{ext} for the extended algorithm.

Lemma 5.1 makes a statement about a weighted average of vector norms. We translate this into a statement about the distance between the final states of message-passing BPQM and extended BPQM applied on some state $|\Psi_{\vec{x}}\rangle, \vec{x}\in\mathcal{C}$ by using the Cauchy-Schwarz inequality and the triangle inequality:

$$\left\|V_r^{\mathrm{ext}}|\Psi_{\vec{x}}\rangle|\vec{0}\rangle - V_r^{\mathrm{mp}}|\Psi_{\vec{x}}\rangle|\vec{0}\rangle\right\| = \left\|\sum_{\vec{j}}\gamma_{\vec{j}}|Q(z,\theta_{\vec{j}})\rangle|0^{n-k}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle - \sum_{\vec{j}}\tilde{\gamma}_{\vec{j}}|\tilde{\varphi}_{\vec{j}}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle\right\| \tag{164}$$

$$\leqslant \left\|\sum_{\vec{j}}\gamma_{\vec{j}}|Q(z,\theta_{\vec{j}})\rangle|0^{n-k}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle - \sum_{\vec{j}}\gamma_{\vec{j}}|\tilde{\varphi}_{\vec{j}}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle\right\|$$
$$+\left\|\sum_{\vec{j}}\gamma_{\vec{j}}|\tilde{\varphi}_{\vec{j}}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle - \sum_{\vec{j}}\tilde{\gamma}_{\vec{j}}|\tilde{\varphi}_{\vec{j}}\rangle|\vec{j}\rangle|\tilde{\theta}_{\vec{j}}\rangle|\tilde{s}_{\vec{j}}\rangle\right\| \tag{165}$$

$$\leqslant \sum_{\vec{j}}|\gamma_{\vec{j}}|\left\||Q(z,\theta_{\vec{j}})\rangle|0^{n-k}\rangle - |\tilde{\varphi}_{\vec{j}}\rangle\right\| + \sum_{\vec{j}}|\gamma_{\vec{j}}-\tilde{\gamma}_{\vec{j}}| \tag{166}$$

$$\leqslant \sqrt{2^n}\sqrt{\sum_{\vec{j}}\gamma_{\vec{j}}^2\left\||Q(z,\theta_{\vec{j}})\rangle|0^{n-k}\rangle - |\tilde{\varphi}_{\vec{j}}\rangle\right\|^2} + \sum_{\vec{j}}\sqrt{|\gamma_{\vec{j}}^2-\tilde{\gamma}_{\vec{j}}^2|} \tag{167}$$

$$\leqslant \sqrt{2^n}\sqrt{\sum_{\vec{j}}\gamma_{\vec{j}}^2\left\||Q(z,\theta_{\vec{j}})\rangle|0^{n-k}\rangle - |\tilde{\varphi}_{\vec{j}}\rangle\right\|} + \sqrt{2^n}\sqrt{\sum_{\vec{j}}|\gamma_{\vec{j}}^2-\tilde{\gamma}_{\vec{j}}^2|}\,, \tag{168}$$

where $\vec{0}$ captures all the ancilla qubits required to perform the message-passing/extended BPQM variant. The fact that $|\gamma_{\vec{j}}-\tilde{\gamma}_{\vec{j}}|\leqslant\sqrt{|\gamma_{\vec{j}}^2-\tilde{\gamma}_{\vec{j}}^2|}$ follows from the fact that $\gamma_{\vec{j}}$ and $\tilde{\gamma}_{\vec{j}}$ have the same sign, which in turn follows directly from the definition (90) of $\tilde{\kappa}_{\vec{j}_1,\vec{j}_2,l}$ in Lemma 5.1 that guarantees it to have the same sign as $\kappa_{\vec{j}_1,\vec{j}_2,l}$. Appealing to Items 2 and 3 in Lemma 5.1 and using $\delta\leqslant 2^{-B}$ gives

$$\left\|V_r^{\mathrm{ext}}|\Psi_{\vec{x}}\rangle|\vec{0}\rangle - V_r^{\mathrm{mp}}|\Psi_{\vec{x}}\rangle|\vec{0}\rangle\right\| \leqslant \frac{2^{5/4}\sqrt{\pi}}{\sqrt{3}}2^{n\cdot(1+\frac{1}{2}\frac{\log 26}{\log 2})-\frac{1}{4}B}$$

$$= \alpha 2^{\beta n - \gamma B}\,, \tag{169}$$

defining $\alpha := \frac{2^{5/4}\sqrt{\pi}}{\sqrt{3}}$, $\beta := 1 + \frac{1}{2}\frac{\log 26}{\log 2}$, $\gamma := 1/4$.

Using this result, we can now analyze how message-passing BPQM performs when decoding the complete codeword. More specifically, we consider that we decode in sequence the codeword bits $X_1, \ldots, X_k$. We want to determine how much lower the probability of decoding all codeword bits correctly is compared to the extended variant of BPQM. For that purpose, we consider the non-normalized post-measurement state conditioned on the k measurement outcomes all being correct. [18] More precisely, for $P_j = (H\,|x_j\rangle\langle x_j|\,H \otimes \mathbb{1})$ the projection operator corresponding to correctly measuring the jth codeword bit, we inductively define

$$|\phi_0^{\mathrm{mp}}\rangle := |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle\,,\ |\phi_0^{\mathrm{ext}}\rangle := |\phi_0^{\mathrm{mp}}\rangle \tag{170}$$

$$|\phi_j^{\mathrm{mp}}\rangle := (V_j^{\mathrm{mp}})^\dagger P_j (V_j^{\mathrm{mp}})\,|\phi_{j-1}^{\mathrm{mp}}\rangle \tag{171}$$

$$|\phi_j^{\mathrm{ext}}\rangle := (V_j^{\mathrm{ext}})^\dagger P_j (V_j^{\mathrm{ext}})\,|\phi_{j-1}^{\mathrm{ext}}\rangle \tag{172}$$

such that $|\phi_j^{\mathrm{mp}}\rangle$ and $|\phi_j^{\mathrm{ext}}\rangle$ denote the state of the message-passing algorithm, respectively extended algorithm, after the j-th codeword bit has been correctly decoded, where $j = 1, \ldots, k$. The probability of correctly decoding the complete codeword is therefore given by $\||\phi_k^{\mathrm{mp}}\rangle\|$ for the message-passing algorithm, respectively by $\||\phi_k^{\mathrm{ext}}\rangle\|$ for the extended variant of BPQM. Surely the latter is larger than the former, since the BPQM algorithm is itself optimal.

Let $\mathcal{H}$ denote the space spanned by the states $\{|\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle\,|\vec{x} \in \mathcal{C}\}$. As seen in Section 4, this is also the space spanned by the orthonormal vectors of the PGM $\{|f_{\vec{x}}\rangle \otimes |\vec{0}\rangle\,|\vec{x} \in \mathcal{C}\}$. The central part of the proof is an analysis how the discretization errors propagate throughout the decoding process. We capture the result in the following claim:

Claim F.1. *For all $j \in \{0, \ldots, k\}$, $|\phi_j^{\mathrm{ext}}\rangle \in \mathcal{H}$ and $\big\||\phi_j^{\mathrm{ext}}\rangle - |\phi_j^{\mathrm{mp}}\rangle\big\| \leqslant j \cdot 2^{k/2+1}\alpha 2^{\beta n - \gamma B}$.*

The reverse triangle inequality immediately gives $\big\||\phi_j^{\mathrm{ext}}\rangle\big\| - \big\||\phi_j^{\mathrm{mp}}\rangle\big\| \leqslant \big\||\phi_j^{\mathrm{ext}}\rangle - |\phi_j^{\mathrm{mp}}\rangle\big\|$, which by the claim implies

$$p^{(\mathrm{ext})} - p^{(\mathrm{mp})} \leqslant k 2^{k/2+1}\alpha 2^{\beta n - \gamma B} \leqslant n 2^{n/2+1}\alpha 2^{\beta n - \gamma B} = 2n\alpha 2^{(\beta+1/2)n - \gamma B}\,. \tag{173}$$

It now only remains to show the validity of the claim. We make an inductive argument in j. Obviously for $j = 0$ the statement holds, as $|\phi_0^{\mathrm{ext}}\rangle = |\phi_0^{\mathrm{mp}}\rangle = |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle$. We now show the induction step as follows. First,

$$\begin{aligned}
\big\||\phi_{j+1}^{\mathrm{ext}}\rangle - |\phi_{j+1}^{\mathrm{mp}}\rangle\big\| &= \big\|(V_{j+1}^{\mathrm{ext}})^\dagger P_{j+1} V_{j+1}^{\mathrm{ext}}\,|\phi_j^{\mathrm{ext}}\rangle - (V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{mp}}\,|\phi_j^{\mathrm{mp}}\rangle\big\| && (174)\\
&\leqslant \big\|(V_{j+1}^{\mathrm{ext}})^\dagger P_{j+1} V_{j+1}^{\mathrm{ext}}\,|\phi_j^{\mathrm{ext}}\rangle - (V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{ext}}\,|\phi_j^{\mathrm{ext}}\rangle\big\| \\
&\quad + \big\|(V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{ext}}\,|\phi_j^{\mathrm{ext}}\rangle - (V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{mp}}\,|\phi_j^{\mathrm{ext}}\rangle\big\| \\
&\quad + \big\|(V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{mp}}\,|\phi_j^{\mathrm{ext}}\rangle - (V_{j+1}^{\mathrm{mp}})^\dagger P_{j+1} V_{j+1}^{\mathrm{mp}}\,|\phi_j^{\mathrm{mp}}\rangle\big\| && (175)\\
&\leqslant \big\|(V_{j+1}^{\mathrm{ext}})^\dagger\,|\varphi_{j+1}\rangle - (V_{j+1}^{\mathrm{mp}})^\dagger\,|\varphi_{j+1}\rangle\big\| \\
&\quad + \big\|V_{j+1}^{\mathrm{ext}}\,|\phi_j^{\mathrm{ext}}\rangle - V_{j+1}^{\mathrm{mp}}\,|\phi_j^{\mathrm{ext}}\rangle\big\| + \big\||\phi_j^{\mathrm{ext}}\rangle - |\phi_j^{\mathrm{mp}}\rangle\big\|\,, && (176)
\end{aligned}$$

where the first inequality is the triangle inequality and the second follows from the fact that a projection cannot increase the norm. Additionally, we have defined the vectors

$$|\varphi_{j+1}\rangle := P_{j+1}(V_{j+1}^{\mathrm{ext}})\,|\phi_j^{\mathrm{ext}}\rangle\,. \tag{177}$$

The third term of (176) can be bounded by $j \cdot 2\alpha 2^{\beta n - \gamma B}(1 + 2^{k/2})$ by the induction hypothesis.

[18] The postulates of quantum mechanics dictate that a post-measurement state (conditioned on some outcome) is obtained by applying a projector on our state and then renormalizing it with the corresponding probability of the measurement outcome. By explicitly not performing this normalization, the terms involved in our analysis remain simpler.

By the induction hypothesis, $|\phi_j^{\text{ext}}\rangle$ must lie in $\mathcal{H}$ and therefore one can find coefficients $a_{\vec{x}}$ such that

$$|\phi_j^{\text{ext}}\rangle = \sum_{\vec{x}\in\mathcal{C}} a_{\vec{x}} |\Psi_{\vec{x}}\rangle \tag{178}$$

and $\sum_{\vec{x}} |a_{\vec{x}}|^2 \leqslant 1$. By the triangle inequality we have

$$\begin{aligned} &\left\| V_{j+1}^{\text{ext}} |\phi_j^{\text{ext}}\rangle - V_{j+1}^{\text{mp}} |\phi_j^{\text{ext}}\rangle \right\| \\ &\leqslant \sum_{\vec{x}\in\mathcal{C}} |a_{\vec{x}}| \left\| V_{j+1}^{\text{ext}} |\Psi_{\vec{x}}\rangle - V_{j+1}^{\text{mp}} |\Psi_{\vec{x}}\rangle \right\| . \end{aligned} \tag{179}$$

Bounding the norm in the summation by (169) and invoking the Cauchy-Schwarz inequality to bound $\sum_{\vec{x}\in\mathcal{C}} |a_{\vec{x}}|$ by $\sqrt{2^k}$ gives

$$\left\| V_{j+1}^{\text{ext}} |\phi_j^{\text{ext}}\rangle - V_{j+1}^{\text{mp}} |\phi_j^{\text{ext}}\rangle \right\| \leqslant 2^{k/2} \alpha 2^{\beta n - \gamma B} \tag{180}$$

For the first term of (176) we can make a similar argument: $|\varphi_{j+1}\rangle$ can be expanded as

$$|\varphi_{j+1}\rangle = \sum_{\vec{x}\in\mathcal{C}} b_{\vec{x}}(V_{j+1}^{\text{ext}}) |\Psi_{\vec{x}}\rangle \tag{181}$$

for some coefficients $b_{\vec{x}}$. We thus get

$$\begin{aligned} &\left\| (V_{j+1}^{\text{ext}})^\dagger |\varphi_{j+1}\rangle - (V_{j+1}^{\text{mp}})^\dagger |\varphi_{j+1}\rangle \right\| \\ &\leqslant \sum_{\vec{x}\in\mathcal{C}} |b_{\vec{x}}| \left\| (V_{j+1}^{\text{ext}})^\dagger (V_{j+1}^{\text{ext}}) |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle - (V_{j+1}^{\text{mp}})^\dagger (V_{j+1}^{\text{ext}}) |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle \right\| && (182) \\ &= \sum_{\vec{x}\in\mathcal{C}} |b_{\vec{x}}| \left\| (V_{j+1}^{\text{mp}}) |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle - (V_{j+1}^{\text{ext}}) |\Psi_{\vec{x}}\rangle \otimes |\vec{0}\rangle \right\| && (183) \\ &\leqslant 2^{k/2} \alpha 2^{\beta n - \gamma B} . && (184) \end{aligned}$$

Summarizing all three terms, we thus obtain

$$\begin{aligned} \left\| |\phi_{j+1}^{\text{ext}}\rangle - |\phi_{j+1}^{\text{mp}}\rangle \right\| &\leqslant j 2^{k/2+1} \alpha 2^{\beta n - \gamma B} + 2 \cdot 2^{k/2} \alpha 2^{\beta n - \gamma B} && (185) \\ &= (j+1) 2^{k/2+1} \alpha 2^{\beta n - \gamma B} . && (186) \end{aligned}$$

To complete the proof, we now argue that $|\phi_{j+1}^{\text{ext}}\rangle \in \mathcal{H}$. This follows directly from $|\phi_j^{\text{ext}}\rangle \in \mathcal{H}$ and the observation made in Section 4 that $(V_j^{\text{ext}})^\dagger P_j (V_j^{\text{ext}})$ realizes the rank-2^{k-1} projector $\Pi_{x_r = m_r}$ introduced in (47).

G Numerical simulation of discretization errors

Here we consider a $(17, 11)$ code, specifically chosen for its MPG structure, which contains multiple subsequent equality and check nodes. The generator matrix and parity-check matrix in standard form are

$$G = \begin{pmatrix} 1&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\ 0&1&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \end{pmatrix} , \tag{187}$$

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \tag{188}$$

Simulating the full quantum circuit of message-passing BPQM would be unfeasible, due to the high number of qubits involved, which is larger than $n(B+1)$. Thanks to the structure of the algorithm, it is possible to store the intermediate states of the circuit with a more refined approach. In fact, following the argument in the proof of Lemma 5.1, one can see that the state of the message passed over some edge e can be written as

$$\sum_{\vec{j}\in\{0,1\}^{k_e-1}} p_{\vec{j}}\rho_{\vec{j},D_e} \otimes |\measuredangle(c_{\vec{j}})\rangle\langle\measuredangle(c_{\vec{j}})|_{C_e} \tag{189}$$

where k_e is the number of check nodes preceding e, $\{p_{\vec{j}}|\vec{j}\}$ is a probability distribution, $\rho_{\vec{j}}$ is a single-qubit density matrix, $c_{\vec{j}} \in \mathcal{A}_B$. Here D_e and C_e denote the data part, respectively the angle part, of the message passed over e. Note that this description is not pure, but rather in terms of a density matrix. This is necessary, since we implicitly traced out the systems A_e, Z_e and S_e.[19]

We can store a state of the form in (189) as a list of 2^{k_e-1} tuples $(p_i, \rho_i, c_i)_{i=0,\dots,2^{k_e-1}-1}$ where the p_i are non-negative and sum up to one, ρ_i are real 2×2 matrices and $c_i \in \mathcal{A}_B$ are angle cosines. For simplicity, we actually store the c_i as floating-point numbers and always round them to the closest value in $\mathcal{A}_B$. Since we do not store the full state and only the reduced state on D_e and C_e, this representation makes it impossible to undo the action of BPQM, so this trick is only useful to determine the decoding performance of a single codeword bit.

An equality node operation takes two messages

$$(p_{1,i}, \rho_{1,i}, c_{1,i})_{i=0,\dots,2^{k_1-1}-1} \text{ and } (p_{2,j}, \rho_{2,j}, c_{2,j})_{j=0,\dots,2^{k_2-1}-1} .$$

Its output message is denoted by

$$(p_{3,ij}, \rho_{3,ij}, c_{3,ij})_{i=0,\dots,2^{k_1-1}-1,j=0,\dots,2^{k_2-1}-1} .$$

$\rho_{3,ij}$ is obtained by applying the unitary $U_{\circledast}$ characterized by the angles $\tilde{R}_B(\alpha(c_{1,i}, c_{2,j}))$ and $\tilde{R}_B(\beta(c_{1,i}, c_{2,j}))$ on the state $\rho_{1,i}\otimes\rho_{2,j}$ and then tracing out the ancilla output qubit. The angle $c_{3,ij}$ is simply given by $R_B(c_{1,i}\cdot c_{2,j})$ and $p_{3,ij}$ by $p_{1,i}p_{2,j}$.

Similarly, the check node operation takes two messages

$$(p_{1,i}, \rho_{1,i}, c_{1,i})_{i=0,\dots,2^{k_1-1}-1} \text{ and } (p_{2,j}, \rho_{2,j}, c_{2,j})_{j=0,\dots,2^{k_2-1}-1} .$$

Its output message is denoted by

$$(p_{3,ijl}, \rho_{3,ijl}, c_{3,ijl})_{i=0,\dots,2^{k_1-1}-1,j=0,\dots,2^{k_2-1}-1,l=0,1} .$$

and is defined by

$$\rho_{3,ijl} = \frac{(\mathbb{1}\otimes\langle l|)\cdot \text{CNOT}\cdot(\rho_{1,i}\otimes\rho_{2,j})\cdot\text{CNOT}\cdot(\mathbb{1}\otimes|l\rangle)}{\text{Tr}\left[(\mathbb{1}\otimes\langle l|)\cdot \text{CNOT}\cdot(\rho_{1,i}\otimes\rho_{2,j})\cdot\text{CNOT}\cdot(\mathbb{1}\otimes|l\rangle)\right]} \tag{190}$$

$$p_{3,ijl} = p_{1,i}p_{2,j}\cdot \text{Tr}\left[(\mathbb{1}\otimes\langle l|)\cdot \text{CNOT}\cdot(\rho_{1,i}\otimes\rho_{2,j})\cdot\text{CNOT}\cdot(\mathbb{1}\otimes|l\rangle)\right] , \tag{191}$$

and

$$c_{3,ijl} = R_B\left(\frac{c_{1,i}+(-1)^l c_{2,j}}{1+(-1)^l c_{1,i}c_{2,j}}\right) . \tag{192}$$

The statistics of measuring the data qubit of the final message $(p_i, \rho_i, c_i)_{i=0,\dots,2^{k-1}-1}$ in the $|\pm\rangle$ basis are given by the probabilities of measuring ρ_i in the $|\pm\rangle$ basis, summed and weighted with the probabilities p_i.

[19] $\rho_{\vec{j}}$ can be obtained by tracing out the system Z in the state $|\tilde{\varphi}_e\rangle$ seen in the proof of Lemma 5.1.

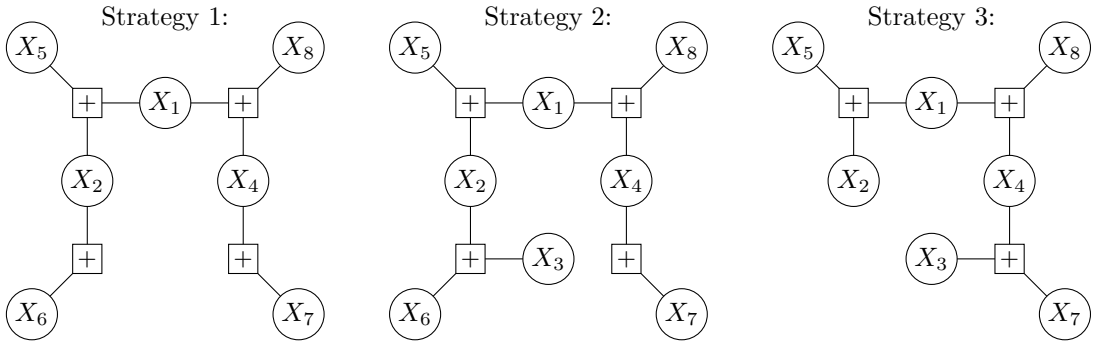


Figure 21: Three candidate strategies for cloning-free decoders.

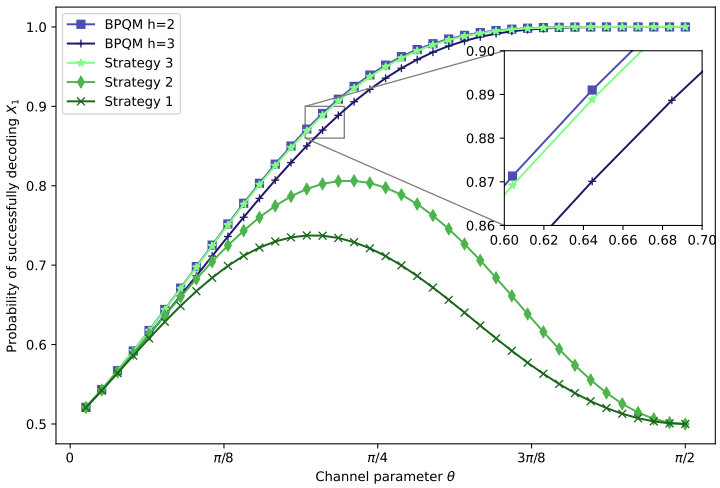


Figure 22: Decoding performance of the three candidate strategies depicted in Fig. 21 for cloning-free decoders of the 8-bit code.

H Alternative decoders for 8-bit code without cloning

In Section 6.1 we investigated the performance of BPQM decoding of X_1 when we unroll the computation tree for different number of steps h. The result was that the decoder performed best under the choice of $h = 2$, in which case the channel output of the bit X_3 is approximately cloned. However, this is not a conclusive proof that cloning certain channel outputs is necessary in order to obtain the best possible decoder. One might argue that there could exist some other choice of $\mathcal{C}'$ that does not require the use of approximate cloning and that achieves better result than the $h = 2$ case. The goal of this appendix is to argue that this is not the case, at least for the arguably natural candidates for such cloning-free decoders.

We consider certain tree subgraphs of the Tanner graph of the 8-bit code, which each define a code, which in turn has its optimal decoder realized by BPQM. We take these induced decoders as candidates for cloning-free decoders which could potentially outperform the $h = 2$ decoder. For example, if one were to choose the subtree consisting of the variable nodes X_1, X_2, X_4, X_5, X_8 and the in-between check nodes, then one would retrieve the identical decoder as in the $h = 1$ case.

In Fig. 21 we depict the subtrees for three candidate strategies which we deem plausible to possibly outperform the $h = 2$ case. Each of them contains more variable nodes than the $h = 1$ case—because the more variables the decoder takes into account, the better we expect the result to be. The challenge in choosing these strategies is that we somehow want to include X_6 and X_7 without having to clone X_3. The numerical results of the three strategies are depicted in Fig. 22.

All three strategies are outperformed by $h = 2$, though strategy 3 performs only slightly worse than $h = 2$. A more careful analysis of the strategies 1 and 2 reveals that their performance depends on the input codeword. They perform badly on codewords where $X_3 = 1$, and in fact have a zero percent success probability for these codewords in the limit $\theta \to \pi/2$. This makes sense—the Tanner graph on which their decoder is based on the implicit assumption that $X_7 = X_4$, which only holds if $X_3 = 0$.

References

[1] J. M. Renes, "Belief propagation decoding of quantum channels by passing quantum messages", New Journal of Physics **19**, 072001 (2017) DOI: 10.1088/1367-2630/aa7c78, arXiv:1607.04833 [quant-ph] (pages 1, 2, 4, 13, 16, 33, 34).

[2] N. Rengaswamy, K. P. Seshadreesan, S. Guha, and H. D. Pfister, "Belief propagation with quantum messages for quantum-enhanced classical communications", npj Quantum Information **7**, 97 (2021) DOI: 10.1038/s41534-021-00422-1, arXiv:2003.04356 [quant-ph] (pages 1–4, 13, 16, 32).

[3] S. Kudekar, T. Richardson, and R. Urbanke, "Spatially Coupled Ensembles Universally Achieve Capacity Under Belief Propagation", IEEE Transactions on Information Theory **59**, 7761–7813 (2013) DOI: 10.1109/TIT.2013.2280915, arXiv:1201.2999 [cs.IT] (pages 1, 32).

[4] H. A. Loeliger and P. O. Vontobel, "Factor Graphs for Quantum Probabilities", IEEE Transactions on Information Theory **63**, 5642–5665 (2017) DOI: 10.1109/TIT.2017.2716422, arXiv:1508.00689 [cs.IT] (pages 4, 11).

[5] M. X. Cao and P. O. Vontobel, "Double-edge factor graphs: Definition, properties, and examples", in 2017 IEEE Information Theory Workshop (ITW) (2017), pp. 136–140, DOI: 10.1109/ITW.2017.8277985, arXiv:1706.00752 [cs.IT] (pages 4, 7, 11).

[6] H.-A. Loeliger, "An introduction to factor graphs", IEEE Signal Processing Magazine **21**, 28–41 (2004) DOI: 10.1109/MSP.2004.1267047 (pages 4, 7, 8, 38).

[7] V. P. Belavkin, "Optimal multiple quantum statistical hypothesis testing", Stochastics **1**, 315 (1975) DOI: 10.1080/17442507508833114 (pages 4, 20).

[8] P. Hausladen and W. K. Wootters, "A 'Pretty Good' Measurement for Distinguishing Quantum States", Journal of Modern Optics **41**, 2385 (1994) DOI: 10.1080/09500349414552221 (pages 4, 20).

[9] N. Rengaswamy and H. D. Pfister, "A Semiclassical Proof of Duality Between the Classical BSC and the Quantum PSC", Mar. 16, 2021, DOI: 10.48550/arXiv.2103.09225, arXiv:2103.09225 [quant-ph] (page 4).

[10] M. Ban, K. Kurokawa, R. Momose, and O. Hirota, "Optimum measurements for discrimination among symmetric quantum states and parameter estimation", International Journal of Theoretical Physics **36**, 1269–1288 (1997) DOI: 10.1007/BF02435921 (pages 4, 20).

[11] M. Sasaki, K. Kato, M. Izutsu, and O. Hirota, "Quantum channels showing superadditivity in classical capacity", Physical Review A **58**, 146–158 (1998) DOI: 10.1103/PhysRevA.58.146, arXiv:quant-ph/9801012 (pages 4, 20).

[12] Y. Eldar and J. Forney G.D., "On quantum detection and the square-root measurement", IEEE Transactions on Information Theory **47**, 858–872 (2001) DOI: 10.1109/18.915636, arXiv:quant-ph/0005132 (pages 4, 20).

[13] T. Richardson and R. Urbanke, *Modern Coding Theory* (Cambridge University Press, 2008), DOI: 10.1017/CBO9780511791338 (pages 5, 8, 28, 34).

[14] D. Poulin, "Optimal and efficient decoding of concatenated quantum block codes", Physical Review A **74**, 052333 (2006) DOI: 10.1103/PhysRevA.74.052333, arXiv:quant-ph/0606126 (page 6).

[15] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes", Quantum Information and Computation **8**, 987–1000 (2008) DOI: 10.26421/QIC8.10-8, arXiv:0801.1241 [quant-ph] (page 6).

[16] Y.-J. Wang, B. C. Sanders, B.-M. Bai, and X.-M. Wang, "Enhanced Feedback Iterative Decoding of Sparse Quantum Codes", IEEE Transactions on Information Theory **58**, 1231–1241 (2012) DOI: 10.1109/TIT.2011.2169534, arXiv:0912.4546 (page 6).

[17] B. Criger and I. Ashraf, "Multi-path Summation for Decoding 2D Topological Codes", Quantum **2**, 102 (2018) DOI: 10.22331/q-2018-10-19-102, arXiv:1709.02154 (page 6).

[18] Y.-H. Liu and D. Poulin, "Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes", Physical Review Letters **122**, 200501 (2019) DOI: 10.1103/PhysRevLett.122.200501, arXiv:1811.07835 (page 6).

[19] A. Rigby, J. C. Olivier, and P. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes", Physical Review A **100**, 012330 (2019) DOI: 10.1103/PhysRevA.100.012330, arXiv:1903.07404 [quant-ph] (page 6).

[20] P. Panteleev and G. Kalachev, "Degenerate Quantum LDPC Codes With Good Finite Length Performance", Quantum **5**, 585 (2021) DOI: 10.22331/q-2021-11-22-585 (page 6).

[21] J. X. Li and P. O. Vontobel, "Pseudocodeword-based Decoding of Quantum Stabilizer Codes", in 2019 IEEE International Symposium on Information Theory (ISIT) (2019), pp. 2888–2892, DOI: 10.1109/ISIT.2019.8849833, arXiv:1903.01202 [cs.IT] (page 6).

[22] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape", Physical Review Research **2**, 043423 (2020) DOI: 10.1103/PhysRevResearch.2.043423, arXiv:2005.07016 [quant-ph] (page 6).

[23] J. X. Li, J. M. Renes, and P. O. Vontobel, "Pseudocodeword-based Decoding of Quantum Color Codes", in (2020), DOI: 10.48550/arXiv.2010.10845, arXiv:2010.10845 [quant-ph] (page 6).

[24] S. Kasi and K. Jamieson, "Towards Quantum Belief Propagation for LDPC Decoding in Wireless Networks", Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 1–14 (2020) DOI: 10.1145/3372224.3419207, arXiv:2007.11069 [cs.NI] (page 6).

[25] M. Leifer and D. Poulin, "Quantum Graphical Models and Belief Propagation", Annals of Physics **323**, 1899–1946 (2008) DOI: 10.1016/j.aop.2007.10.001, arXiv:0708.1337 [quant-ph] (page 7).

[26] H. A. Bethe, "Statistical Theory of Superlattices", Proceedings of the Royal Society A **150**, 552–575 (1935) DOI: 10.1098/rspa.1935.0122 (page 7).

[27] R. Peierls, "Statistical theory of superlattices with unequal concentrations of the components", Proceedings of the Royal Society A **154**, 207–222 (1936) DOI: 10.1098/rspa.1936.0047 (page 7).

[28] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation", in Proceedings of the 13th International Conference on Neural Information Processing Systems, NIPS'00 (2000), pp. 668–674 (page 7).

[29] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding Belief Propagation and Its Generalizations", in *Exploring Artificial Intelligence in the New Millennium*, edited by G. Lakemeyer and B. Nebel (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003), pp. 239–269 (page 7).

[30] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms", Information Theory, IEEE Transactions on **51**, 2282–2312 (2005) DOI: 10.1109/TIT.2005.850085 (page 7).

[31] M. B. Hastings, "Quantum belief propagation: An algorithm for thermal quantum systems", Physical Review B **76**, 201102 (2007) DOI: 10.1103/PhysRevB.76.201102, arXiv:0706.4094 [cond-mat] (page 7).

[32] D. Poulin and M. B. Hastings, "Markov Entropy Decomposition: A Variational Dual for Quantum Belief Propagation", Physical Review Letters **106**, 080403 (2011) DOI: 10.1103/PhysRevLett.106.080403, arXiv:1012.2050 [quant-ph] (page 7).

[33] M. X. Cao and P. O. Vontobel, "Quantum factor graphs: Closing-the-box operation and variational approaches", in 2016 International Symposium on Information Theory and Its Applications (ISITA) (2016), pp. 651–655 (page 7).

[34] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm", IEEE Transactions on Information Theory **47**, 498–519 (2001) DOI: 10.1109/18.910572 (page 7).

[35] G. D. Forney, "Codes on graphs: normal realizations", IEEE Transactions on Information Theory **47**, 520–548 (2001) DOI: 10.1109/18.910573 (page 7).

[36] C. W. Helstrom, *Quantum detection and estimation theory*, Vol. 123, Mathematics in Science and Engineering (Academic, London, 1976), DOI: 10.1016/S0076-5392(08)60247-7 (page 9).

[37] S. Guha, "Structured optical receivers to attain superadditive capacity and the Holevo limit", Physical Review Letters **106**, 240502 (2011) DOI: 10.1103/PhysRevLett.106.240502, arXiv:1101.1550 [quant-ph] (page 9).

[38] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?", IEEE Transactions on Information Theory **45**, 2173–2181 (1999) DOI: 10.1109/18.782170 (page 9).

[39] J. C. Bridgeman and C. T. Chubb, "Hand-waving and interpretive dance: an introductory course on tensor networks", Journal of Physics A: Mathematical and Theoretical **50**, 223001 (2017) DOI: 10.1088/1751-8121/aa6dc3, arXiv:1603.03039 [quant-ph] (page 9).

[40] V. Bergholm, J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, "Quantum circuits with uniformly controlled one-qubit gates", Physical Review A **71**, 052330 (2005) DOI: 10.1103/PhysRevA.71.052330, arXiv:quant-ph/0410066 (page 16).

[41] C. H. Bennett, "Logical Reversibility of Computation", IBM Journal of Research and Development **17**, 525–532 (1973) DOI: 10.1147/rd.176.0525 (page 24).

[42] R. P. Brent, "Multiple-precision zero-finding methods and the complexity of elementary function evaluation", in *Analytic Computational Complexity*, edited by J. F. Traub (Academic Press, 1976), pp. 151–176, DOI: 10.1016/B978-0-12-697560-4.50014-9, arXiv:1004.3412 [cs.NA] (page 27).

[43] Harvey and van der Hoeven, "Integer multiplication in time O(n log n)", Annals of Mathematics **193**, 563 (2021) DOI: 10.4007/annals.2021.193.2.4 (page 27).

[44] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, "Quantum algorithm and circuit design solving the Poisson equation", New Journal of Physics **15**, 013021 (2013) DOI: 10.1088/1367-2630/15/1/013021, arXiv:1207.2485 [quant-ph] (page 27).

[45] M. K. Bhaskar, S. Hadfield, A. Papageorgiou, and I. Petras, "Quantum algorithms and circuits for scientific computing", Quantum Information & Computation **16**, 197–236 (2016) DOI: 10.26421/QIC16.3-4-2, arXiv:1511.08253 [quant-ph] (page 27).

[46] T. Häner, M. Roetteler, and K. M. Svore, "Optimizing Quantum Circuits for Arithmetic", DOI: 10.48550/arXiv.1805.12445 (2018) DOI: 10.48550/arXiv.1805.12445, arXiv:1805.12445 [quant-ph] (page 27).

[47] S. Wang, Z. Wang, W. Li, L. Fan, G. Cui, Z. Wei, and Y. Gu, "Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method", Quantum Information Processing **19**, 347 (2020) DOI: 10.1007/s11128-020-02855-7, arXiv:2001.00807 [quant-ph] (pages 27, 28).

[48] J. Watrous, *The Theory of Quantum Information* (Cambridge University Press, 2018), DOI: 10.1017/9781316848142 (page 30).

[49] D. Bruß, D. P. DiVincenzo, A. Ekert, C. A. Fuchs, C. Macchiavello, and J. A. Smolin, "Optimal universal and state-dependent quantum cloning", Physical Review A **57**, 2368–2378 (1998) DOI: 10.1103/PhysRevA.57.2368, arXiv:quant-ph/9705038 (page 31).

[50] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels", IEEE Transactions on Information Theory **55**, 3051–3073 (2009) DOI: 10.1109/TIT.2009.2021379, arXiv:0807.3917 [cs.IT] (page 33).

[51] M. M. Wilde and S. Guha, "Polar codes for classical-quantum channels", IEEE Transactions on Information Theory **59**, 1175–1187 (2013) DOI: 10.1109/TIT.2012.2218792, arXiv:1109.2591 [quant-ph] (page 33).

[52] J. M. Renes and M. M. Wilde, "Polar Codes for Private and Quantum Communication Over Arbitrary Channels", IEEE Transactions on Information Theory **60**, 3090–3103 (2014) DOI: 10.1109/TIT.2014.2314463, arXiv:1212.2537 [quant-ph] (page 33).

[53] S. Guha and M. Wilde, "Polar coding to achieve the Holevo capacity of a pure-loss optical channel", in Proceedings of the 2012 IEEE International Symposium on Information Theory (ISIT) (2012), pp. 546–550, DOI: 10.1109/ISIT.2012.6284250, arXiv:1202.0533 [cs.IT] (page 33).