Modular quantum signal processing in many variables

Zane M. Rossi[1], Jack L. Ceroni[2], and Isaac L. Chuang[1]

[1]Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA
[2]Department of Mathematics, University of Toronto, Toronto, Ontario M5S1A1, Canada
June 10, 2025

Despite significant advances in quantum algorithms, quantum programs in practice are often expressed at the circuit level, forgoing helpful structural abstractions common to their classical counterparts. Consequently, as many quantum algorithms have been unified with the advent of quantum signal processing (QSP) and quantum singular value transformation (QSVT), an opportunity has appeared to cast these algorithms as modules that can be combined to constitute complex programs. Complicating this, however, is that while QSP/QSVT are often described by the polynomial transforms they apply to the singular values of large linear operators, and the algebraic manipulation of polynomials is simple, the QSP/QSVT protocols realizing analogous manipulations of their embedded polynomials are non-obvious. Here we provide a theory of modular multi-input-output QSP-based superoperators, the basic unit of which we call a *gadget*, and show they can be snapped together with LEGO-like ease at the level of the functions they apply. To demonstrate this ease, we also provide a Python package for assembling gadgets and compiling them to circuits. Viewed alternately, gadgets both enable the efficient block encoding of large families of useful multivariable functions, and substantiate a functional-programming approach to quantum algorithm design in recasting QSP and QSVT as monadic types.

1 Introduction

Quantum algorithms, persistently strange, remain difficult to design and interpret [1]; correspondingly, great effort has been spent to not only generate algorithms, but formalize the motifs of quantum advantage [2, 11, 51]. Both of these desires have been partially addressed by the advent of quantum signal processing (QSP) [43–45, 76] and its lifted version quantum singular value transformation (QSVT) [26]. These algorithms allow one to modify the singular values of large linear operators by precisely tunable polynomial functions, unifying and simplifying most known quantum algorithms [47], with good numerical properties [9, 20, 21, 29], and deep, fruitful connections to well-studied matrix factorization techniques [67].

In reducing diverse algorithmic problems to statements on the existence of classes of polynomial functions, QSP and QSVT encourage but do not substantiate interpreting quantum computations *functionally*. That is, it becomes tempting to treat these algorithms purely in terms of the functions they apply, directly working within the natural ring over polynomials (generated by addition and multiplication), or the monoid associated with polynomial composition. Such *function-first* operations have clear semantic interpretation, offer connection to a wealth of preëxisting mathematical literature, and connect neatly to existing classical formalisms for the design, verification, and optimization of programs [23, 62, 63].

Limited techniques for treating QSP and QSVT function-first have appeared, answering the following narrow question: given repeatable oracle access to a QSP protocol, and a description of another QSP protocol, can one instantiate the protocol achieving *the composition of the functions achieved by each protocol* in black-box way? Independent lines of work [48, 60] have answered this positively, with the former enumerating necessary and sufficient conditions under which such composition is possible, and the latter examining recursive composition converging to a useful class of functions. In both works, however, only a single oracular unitary process is considered, and the resulting computations are described only by compositions of polynomials in a single variable, limiting the variety of achievable behavior and concomitant algorithmic use. We also note that such limited methods for self-embedding simple quantum subroutines has a long history [28, 31, 32, 46, 76], with roots in composite pulse theory [39, 68] and

Zane M. Rossi: This author contributed equally. zmr@mit.edu/zmr@g.ecc.u-tokyo.ac.jp
Jack L. Ceroni: This author contributed equally. jack.ceroni@mail.utoronto.ca/jceroni@uchicago.edu

classical signal processing [34, 61], albeit without clear functional interpretation. Ultimately, a mature, function-first theory, closed over operations natural to multivariable polynomials (e.g., ring operations, the composition monoid) has not been realized.

In moving to the multiple oracle setting, we are seeking a more ambitious construction than previous work; that is, we want to be able to blithely snap together basic modules, with structure similar to that of QSP, at the level of the functions they apply. We will show this is surprisingly possible, with a bit of additional work, through a careful synthesis of two previous techniques: multivariable QSP (M-QSP) [59] and an intricate series of powerful results on fractional queries from Sheridan, Maslov, and Mosca [64] extended beautifully to QSVT by Gilyén, Su, Low, and Weibe [26]. In turn, these M-QSP-based examples will then be generalized to a more encompassing theory of *gadgets*. Ultimately, we will be able to succinctly describe and subsume previous work on modular QSP-based algorithms [48, 60], the most obvious bridge between our work and previous work on the level of the aforementioned multivariable quantum signal processing (M-QSP) [59]. M-QSP (Def. 2.1), despite having a well-defined series of conditions under which a transform is achievable, has proven unwieldy in practice, as multivariable analogues of the algebraic geometric results characterizing achievable polynomial transforms (chiefly the Fejér-Riesz Theorem [24, 56]) are unavoidably weaker in the multivariable setting. We will show that such barriers, however, do not rule out the existence of independently interesting subroutines strictly subsuming M-QSP, and which allow one to freely manipulate and combine polynomial functions. That is, to yield the same functional class as the one originally desired but disallowed in standard M-QSP, we show it is sufficient to generate algebraic structures over polynomials with rich closures; constructively achieving such closures by expanding the class of circuits considered is a central result of this paper.

The fundamental units of computation we construct are termed *gadgets*, which take as input (possibly multiple) unknown unitary processes, and produce as output (possibly multiple) unitary processes, where outputs depend entirely on multivariable polynomials applied to parameters of the inputs. We highlight an important subclass of gadgets, *atomic gadgets*, which are built directly using M-QSP, and subsequently combined with rich fractional query techniques [26, 64] to realize *snappable gadgets*, which, having been suitably *corrected*, can be freely combined. We provide a characterization of the *syntax* for combining gadgets into composite gadgets (described by a two-level grammar, which under basic restrictions becomes context-free), as well as a description of valid *semantic* manipulations for functions achieved by these gadgets (in the form of a monadic type over M-QSP protocols). In providing both a *syntactic* and *semantic* theory of gadgets, we provide a formal response to the temptation to view QSP- and QSVT-based modules purely functionally, and embed our constructions within established terminology from functional programming and category theory.

Unlike in the single-oracle setting, the conditions under which functional manipulations are possible within gadgets (for which the work of [48, 60] consider a special case) do not depend solely on simple circuit symmetries; consequently, the bulk of appendices of this work are spent specifying detailed functional analytic arguments on the existence and performance of special *correction protocols* by which the output of a given gadget can be *corrected* such that it can be connected to the input of another gadget while preserving the desired functional manipulation. We show that such gadget connections are generally ill-conditioned without correction, and we prove that correction generically cannot be done exactly, only to arbitrary precision. In fact, it is this softened condition of approximate achieveability that enables our variety of functional transforms. We emphasize that unlike with other methods for generating composite superoperators, like linear combination of unitaries (LCU), we maintain both stringent space requirements and rapid convergence (polylogarithmic in inverse error) to intended functional transforms. Constructing these correction subroutines constitutes the core difficulty of this work—however, once constructed, these protocols allow one to immediately characterize the space and query complexity of gadgets, placing them in comparison with other block-encoding manipulation techniques [6, 26].

The results of this work can be posed a few ways, in order of increasing abstraction. At a practical level we achieve highly space- and query-efficient block encodings of multivariable polynomials in commuting linear operators, showing that QSVT can retain key advantages over alternative methods like linear combination of unitaries (LCU) [6, 7] in the multivariable setting (this fact is detailed in Appx. B). At a higher level, this work solidifies and expands the approach of [60] in treating QSP and QSVT protocols in the language of functional programming through the composite objects of *gadgets*. I.e., we frame our results as instantiating a *monad* over QSP and QSVT *types*, and situate this monad in quantum programming language theory, with possible compositions described by simple formal grammars.

To better support our claims we provide a series of example constructions of achievable multivariable functions (see Fig. 3), as well as a Python package, `pyqsp`, for automatically assembling gadgets and compiling their circuits located at `[https://github.com/ichuang/pyqsp/tree/beta]`. Note that this

package (in the `beta` branch) is compatible with phases computed by the (recently improved) `main` branch, but these branches are currently un-merged given competing conventions used (for legacy reasons) in some sub-modules of `main`. The current contact for and maintainer of this repository is the first listed author.

Our presentation is divided in two components: the main body Secs.1-5, and appendices A-I. In the main body, we provide the construction for *gadgets* (Sec. 2) (of which there are sub-types, Appx. D.3), define and discuss the connection of these gadgets into composite gadgets (Sec. 3), and provide a series of linked concrete examples of useful functions achieved by assembling gadgets (Sec. 4), with detailed cost analysis (Appx. C). Secondly, we provide a series of appendices which cover proofs of theorems stated in the main body (Appx. A), detailed analysis of the aforementioned correction protocols (Appxs. E and F), explanation of the non-obvious method for computing gadget cost (Appx. C), discussion of how gadgets instantiate monadic functions and relate to functional programming (gadget *semantics*, Appxs. I.1 and I.2), and finally discussion of a formal language for gadgets constituted by an attribute grammar (gadget *syntax*, Appx. I.3).

2 Motivating and constructing gadgets

We have claimed that gadgets, properly constructed, can enable the efficient achievement of multivariable polynomial transformations and a modular approach to assembling quantum algorithms. This is shown explicitly in this section, by building from M-QSP and its combination with fractional queries to create *atomic gadgets*, a remarkably useful subclass of gadgets. We use and extend fractional query techniques [26, 64] to allow certain imperfections in atomic gadgets to be corrected, leading the fundamental unit of our construction, the *snappable gadget*, which we suitably generalize beyond explicit reference to QSP. For the moment, however, we recall that QSP circuits prescribe how to take simply parameterized unknown unitary inputs and produce as output unitaries with properties non-linearly dependent on, and carefully tuneable with respect to, the input unitaries [26]. Crucially, within QSP, input unitaries are highly constrained (in fact rotations about a fixed, known axis), while the output unitiaries are not; even worse, if one were to consider a QSP-like ansatz depending on multiple unknown unitaries as input, not only can the form of the output unitary depend *on the function applied*, but also on the *relation between each of the many unknown inputs*. At a high level it is this *mismatch of basis* between input and output unitaries in QSP (and inherited in singular vector subspaces in QSVT) which prevents simple composition of protocols at the level of the functions they achieve.

The core result of this work (Thm. 3.2) lies in a terse equivalence between general compositions of multivariable polynomials and general compositions of gadgets (themselves based on QSP-like components) achieving these polynomials. Toward gathering the components to state this equivalence, we first build off of a QSP-related circuit to generate a powerful subclass of gadgets, followed by identifying key issues in their compositions, and special protocols to resolve these issues.

To start off, we recall the general definition of an *M-QSP protocol* [59], of which a standard QSP protocol is a special case. For the moment we will not need to know any properties of this algorithm (covered in Appx. D for the curious) besides its simple circuit form, other than to note that the problems M-QSP protocols exhibit when we attempt to combine them as modules will showcase shortcomings eventually resolved by the definition of *gadgets* and *correction protocols* given later in this section. These problems also distinguish the general setting from the single variable setting, where the former is substantively more difficult to address.

Definition 2.1 (M-QSP protocol)**.** Let $\Phi \equiv \{\phi_0, \ldots, \phi_n\} \in \mathbb{R}^{n+1}$ and $s \in [t]^n$, $t \in \mathbb{N}$, where $[t] = \{0, \ldots, t-1\}$. Then the t-variable M-QSP protocol specified by (Φ, s) generates the unitary circuit

$$\Phi[U_0, \cdots, U_{t-1}] \equiv e^{i\phi_0\sigma_z} \prod_{k=1}^{n} U_{s_k} e^{i\phi_k\sigma_z}, \tag{1}$$

where $U_0, \cdots, U_{t-1}$ are unitaries of the form $U_k \equiv e^{i\theta_k\sigma_x}, k \in [t]$ for unrelated θ_k (see Def. 2.2). Here σ_z, σ_x are the standard Pauli matrices. We say an M-QSP protocol is *antisymmetric* if both $(N \circ R)(\Phi) = \Phi$ and $R(s) = s$, where R, N reverse and negate lists of scalars respectively, with equality taken elementwise (when Φ contains zeros and the oracles can commute past each other, this definition can be replaced one stating $\Phi[U_0, \cdots, U_{t-1}] = \Phi[U_0, \cdots, U_{t-1}]^\dagger$.). Often s will be suppressed when denoting (Φ, s) as a superoperator (see Appx. D). An M-QSP protocol *achieves* a function f, where $f \equiv \langle 0|\Phi[U_0, \cdots, U_{t-1}]|0\rangle$ is a function over scalars (namely those $x_k \equiv \cos(\theta_k)$) parameterizing the oracles. □

To understand the problem eventually solved by gadgets, we recall one of the insights of previous work on embedding QSP-based circuits within themselves [60]. In this work, it was shown how to iteratively compose *antisymmetric* single-variable QSP protocols to compose their achieved functions. Such compositions can be realized by successively using the QSP unitary produced by some protocol Φ_k as the signal operator of another QSP protocol Φ_{k+1}, continuing perhaps for some collection protocols $\{\Phi_0, \ldots, \Phi_{n-1}\}$. The admissibility of this recursion follows from the fact that for any antisymmetric protocol Φ and oracle unitary $e^{i\theta\sigma_x}$, the polynomial P achieved by a single-variable QSP protocol Φ is invariant under *twisting*. More specifically, for any unitary U and any σ_z-rotation $e^{i\varphi\sigma_z}$, $\Phi[e^{i\varphi\sigma_z}Ue^{-i\varphi\sigma_z}] = e^{i\varphi\sigma_z}\Phi[U]e^{-i\varphi\sigma_z}$. Thus for a sequence of antisymmetric QSP protocols $\{\Phi_0, \ldots, \Phi_{n-1}\}$ the $(k+1)$-th protocol automatically treats the function achieved by the k-th protocol as a variable for the function achieved by Φ_{k+1}. The output of each protocol remains in some sense *embeddable*, and the composability of functions follows. Below, we clarify this notion of embeddability (introducing a few named types).

Definition 2.2 (Variations on embeddable unitaries)**.** An SU(2) unitary U is *embeddable* if it has the form $e^{i\theta\sigma_x}$ for some $\theta \in [0, \pi]$. A unitary is ε-embeddable if it is at most ε-far (in operator norm) from an embeddable unitary. An SU(2) unitary U is *twisted embeddable* (*half-twisted embeddable*) if it has the form $e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}$ for some $\theta \in [0, \pi]$ and $\varphi \in [-\pi, \pi]$ ($\varphi \in [-\pi/2, \pi/2]$). A unitary U is ε-twisted embeddable (ε-half-twisted embeddable) if it is at most ε-far from a twisted embeddable (half-twisted embeddable) unitary over a specified domain. Given a twisted or half-twisted embeddable unitary of the form $e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}$, we refer to $e^{i\theta\sigma_x}$ as its *embeddable component*. □

In what follows we show that maintaining such embeddability is vital for a variety of desired computational tasks and discuss some subtleties of the definitions contained in Def. 2.2. To begin, we highlight a pedagogical example: consider two antisymmetric, single-variable QSP protocols Φ_0 and Φ_1 and signal operators $e^{i\theta_0\sigma_x}$, $e^{i\theta_1\sigma_x}$. By Thm. D.7, $\Phi_0[e^{i\theta_0\sigma_x}]$ and $\Phi_1[e^{i\theta_1\sigma_x}]$ are twisted embeddable, with

$$\Phi_0[e^{i\theta_0\sigma_x}] = e^{i\varphi_0\sigma_z/2}e^{i\theta_0'\sigma_x}e^{-i\varphi_0\sigma_z/2} \quad \text{and} \quad \Phi_1[e^{i\theta_1\sigma_x}] = e^{i\varphi_1\sigma_z/2}e^{i\theta_1'\sigma_x}e^{-i\varphi_1\sigma_z/2}. \tag{2}$$

Suppose Φ_0 and Φ_1 achieve functions $f_0(\cos(\theta_0))$ and $f_1(\cos(\theta_1))$ as the top-left elements of their unitaries when applied to the signals θ_0, θ_1 (i.e., by the above equation, we meant that $\theta_0' = \arccos(f_0(\cos(\theta_0)))$ and $\theta_1' = \arccos(f_1(\cos(\theta_1)))$ for brevity). Then given a two-variable antisymmetric M-QSP protocol Φ achieving the function $g(x_0, x_1)$, in the general case,

$$\langle 0|\Phi\left[\Phi_0[e^{i\theta_0\sigma_x}], \Phi_1[e^{i\theta_1\sigma_x}]\right]|0\rangle \neq g\left(f_0(\cos(\theta_0)), f_1(\cos(\theta_1))\right). \tag{3}$$

This is entirely due to the conjugation of the embeddable components of $\Phi_0[e^{i\theta_0\sigma_x}]$ and $\Phi_1[e^{i\theta_1\sigma_x}]$ by *different* σ_z-rotations. Because $\varphi_0 \neq \varphi_1$, it is no longer possible to factor the σ_z-rotations out of the overall M-QSP protocol, in the same way as the single variable case. In single variable protocols, accumulating σ_z-conjugations by successively composing antisymmetric QSP protocols had no effect on the function achieved by the QSP polynomial. Here, one finds

$$\Phi[e^{i\varphi\sigma_z}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z}] = e^{i\varphi\sigma_z}\Phi[e^{i\theta\sigma_x}]e^{-i\varphi\sigma_z} \implies \langle 0|\Phi[e^{i\varphi\sigma_z}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z}]|0\rangle = \langle 0|\Phi[e^{i\theta\sigma_x}]|0\rangle. \tag{4}$$

Therefore, in the multivariable case, we require a technique for suppressing the disagreeing σ_z-conjugations of Eq. (2), leaving the relevant embeddable components such that there is no interference with the outer M-QSP protocol. Such a correction would turn Eq. (3) into an approximate equality. Consequently we identify a concrete question: is it possible to map a twisted embeddable unitary U with unknown φ and θ to its embeddable component? In other words, can we *align* the axes of rotation for two (or more) unitary oracles in an efficient, black box way? We provide an affirmative answer to this question, up to slightly weakened conditions, and call this process of alignment *correction*.

Lemma 2.1 (Efficient correction of ε-twisted embeddable unitaries)**.** Let U be a ν-twisted embeddable unitary, so U is ν-close to $U' = e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}$. Suppose $\cos(\theta) \in [-1+\delta, 1-\delta]$. Let $\varepsilon > 0$. Then, given controlled access to U, there exists a quantum circuit using $\zeta = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/2}))$ black-box calls to U and a single ancilla qubit which yields a $(\zeta\nu + \varepsilon)$-embeddable unitary V, which is at most $(\zeta\nu + \varepsilon)$-far from $e^{i\theta\sigma_x}$ (the embeddable component of U') with success probability at least $(1 - (\zeta\nu + \varepsilon))^2$. Alternatively, given access to two ancilla qubits and uncontrolled oracle access to U, it is possible to implement a circuit which achieves the same $(\zeta\nu + \varepsilon)$-approximation with the same asymptotic query complexity and success probability, on the domain $\cos(\theta_k) \in [\delta, 1-\delta]$. Finally, if U is ν-half-twisted embeddable, and satisfies the same conditions, with the additional constraint that $\cos(\varphi) \in [\gamma, \sqrt{1-\gamma^2}]$ (or $\varphi \in [-\pi/2 + 2\gamma, -2\gamma] \cup [2\gamma, \pi/2 - 2\gamma]$), there exists a quantum circuit using

$\zeta' = \mathcal{O}(\delta^{-1}\gamma^{-2}\log^2(\gamma^{-4}\varepsilon^{-2}\delta^{-1/2}))$ black-box calls to U and zero ancilla qubits which yields an $(\zeta'\nu+\varepsilon)$-embeddable unitary, at most $(\zeta'\nu+\varepsilon)$-far from the embeddable component $e^{i\theta\sigma_x}$, with unit success. In all cases, the circuits effectuating the corrections can be efficiently and constructively specified. □

Lem. 2.1 (proven in Appx. A) shows that given an approximately embeddable unitary, (1) the error accumulated during its correction is proportional to the error of the input, and (2) the constant of proportionality is linear in the length of the correction procedure. This scaling as not as poor as it first appears, mainly due to the ease with which ν can be made small and the limited number of correction procedures that will be used by any physically reasonable protocols. We call such protocols *acceptable* if their length scales at worst inverse polynomially in the desired output error, and argue for the reasonableness of this class of protocols in the context of the efficiency of Lem. 2.1 in the expanded Rem. A.1. The resource costs for the correction procedures under different access model assumptions is elaborated upon further in Appxs. B and H (we also provide a high level summary in Table 1, though proofs of all cited complexities are mostly relegated to the appendices mentioned above).

Method/Access	Complexity	Anc.	$\cos(\theta)$-dom.	$\cos(\varphi)$-dom.	Succ.
Anc./controlled	$\widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$	1	$[-1+\delta, 1-\delta]$	$[-1, 1]$	$1-\varepsilon$
No anc./un-controlled	$\widetilde{\mathcal{O}}(\delta^{-1}\gamma^{-2}\log^2(\varepsilon^{-1}))$	0	$[-1+\delta, 1-\delta]$	$[\gamma, \sqrt{1-\gamma^2}]$	1
Anc./un-controlled	$\widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$	2	$[\delta, 1-\delta]$	$[-1, 1]$	$1-\varepsilon$

Table 1: Comparison of the resource costs for three different correction procedures which apply under different assumptions about the oracle access model, using θ, ϕ conventions as in Lem. 2.1. These complexities are mainly analyzed in Appx. B; the distinguishing factor is whether subroutines are provided as coherently controlled quantum processes (presupposing access to clean ancillary qubits) or black-box, after which one can either controllize them using additional space under θ-restrictions, or perform correction in place (under ϕ-restrictions according to a factor γ, introduced and discussed in Appx. F.3).

The correction procedure described enables us to take M-QSP protocols, apply corrective protocols to their output, and pass these outputs into other M-QSP protocols, such that the function achieved is the multivariate composition of the functions achieved by the individual protocols. The closure over unitary superoperators achieved by arbitrarily composing sequences of M-QSP unitaries and corrections is formalized by the notion of a *gadget*. Below, we give two definitions for gadgets; the first, the *atomic gadget* (Def. 2.3), is built directly from M-QSP protocols, while the second, the *gadget* (Def. 2.4) captures the most general relevant definition for our purposes. Under special conditions (discussed in detail in Sec. D.3), atomic gadgets form a strict subset of gadgets, and we will work with such atomic gadgets exclusively. In the following section, we show that such gadgets can be successively composed to yield composite gadgets describing complex, expressive quantum computations and achieving general and interpretable functional transforms. Moreover, gadget objects have a highly-interpretable form; we describe their permitted combinations described by attribute grammars (a *syntax* for gadgets), and describe their functional action by monadic functions over QSP/QSVT types (a *semantics* for gadgets).

Definition 2.3 (Atomic (a, b) gadget)**.** Let (Ξ, S) be a tuple with $\Xi \equiv \{\Phi_0, \ldots, \Phi_{b-1}\}$ where $\Phi_k \in \mathbb{R}^{r_k+1}, r_k \in \mathbb{N}$ and $S \equiv \{s_0, \ldots, s_{b-1}\}$ where $s_k \in [a]^{r_k}$ and $[a] = \{0, \ldots, a-1\}$. Then the atomic (a, b) gadget labeled by (Ξ, S) refers to the parameterization of b M-QSP protocols using a single-qubit oracles U_k (by definition embeddable), where each of the b output unitaries $\Phi_k[U_0, \ldots, U_{a-1}], k \in [b]$ has parameterization (Φ_k, s_k) following Def. 2.1. A gadget is *antisymmetric* if its constituting (Ξ, S) generate only antisymmetric M-QSP protocols [60]. Note that not all[1] *atomic gadgets* are *gadgets* (Def. 2.4) (see Sec. D.3), though atomic gadgets considered in this work (e.g., antisymmetric gadgets) *will* be. □

Definition 2.4 ((a, b) gadget)**.** An (a, b) gadget $\mathfrak{G}$ is a superoperator which takes as input a single-qubit embeddable oracles U_k for $k \in [a]$, and outputs b twisted-embeddable unitaries U'_j for $j \in [b]$, denoted by $U'_j = \mathfrak{G}[U_0, \ldots, U_{a-1}]_j$. Note that *antisymmetric* atomic (a, b) gadgets (Def. 2.3) are a strict subset of (a, b) gadgets (Thm. D.7). However, there also exist other circuit-based methods for constructing gadgets from M-QSP (Def. D.2). A gadget $\mathfrak{G}$ is said to achieve $F \equiv \{f_0, \ldots, f_{b-1}\}$ where $f_j \equiv \langle 0|U'_j|0\rangle$ is the top-left matrix element of the j-th output unitary given by applying $\mathfrak{G}$ to a set of oracles. In the case that the input oracles are parameterized by variables $x_0, \ldots, x_{a-1}$, each output unitary U'_j and each f_j can be treated as functions $U'_j(x_0, \ldots, x_{a-1})$ and $f_j(x_0, \ldots, x_{a-1})$. □

[1] This may seem like a poor choice of convention, but we leave the antisymmetry requirement out of our definition here because general atomic gadgets can still be useful when restricted to *discrete* signals. In almost all cases however one should assume antisymmetry.

Remark 2.1 (Gadget terminology)**.** Going forward, if an (a, b) gadget $\mathfrak{G}$ is said to achieve output unitaries $U'_j(x_0, \ldots, x_{a-1}), j \in [b]$ and functions $f_j(x_0, \ldots, x_{b-1}), j \in [b]$, it is meant that the gadget achieves these functions for input unitaries $U_k = e^{i\theta_k \sigma_x}, k \in [a]$, where each $x_k = \cos(\theta_k), k \in [a]$ is possibly restricted to a fixed domain specified along with the gadget. □

In many cases, a gadget will achieve its output unitaries U'_j via some sequence of products and interspersed quantum (and possibly even classical) operations. The only criteria on the input oracles is that the gadget sees them as black-boxes, which can only be utilized through quantum circuit queries. When we refer to the *cost* associated with a gadget $\mathfrak{G}$, we are referencing the number of black-box queries that must be made to the input oracles to achieve some particular output unitary, possibly at a specified precision over a specified parameter domain. Detailed discussion of cost and its associated objects for gadgets are restricted to Appx. C.

We now present the key theorem which describes how the correction protocol of Lem. 2.1 enables *snappable gadgets* to be constructed from general gadgets. Snappable gadgets are engineered such that their outputs are embeddable – not merely twisted-embeddable – and may be passed as input to another gadget in a black-box way and while respecting achieved functions. Note that going forward we let $\widetilde{\mathcal{O}}$ indicate asymptotic complexity up to leading order/logarithmic factors in each independent variable.

Theorem 2.1 (Efficient correction of unitaries produced by (a, b) gadgets)**.** Let $\varepsilon, \delta > 0$ and $\mathfrak{G}$ an (a, b) gadget whose output unitaries are guaranteed to be ν-twisted-embeddable: they are ν-close to unitaries $e^{i\varphi\sigma_z/2} e^{i\theta_k \sigma_x} e^{-i\varphi\sigma_z/2}$ whose embeddable components encode $\cos\theta_k, k \in [b]$ within $[-1+\delta, 1-\delta]$. Assuming an access model in which one can query the *controlled* single-qubit unitaries achieved by the output legs of $\mathfrak{G}$, then given $k \in [b]$, there exists a computable $\nu' = \widetilde{\mathcal{O}}(\delta\varepsilon)$ and a quantum circuit using $\zeta = \widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$ black-box calls to the controlled unitaries produced by executing $\mathfrak{G}$ on the input oracles, as well as one ancilla qubit, such that if $\nu \leq \nu'$ then the circuit achieves an ε-approximation of $U'_k = e^{i\theta_k \sigma_x}$ (the embeddable component of the k-th output unitary of $\mathfrak{G}$), with success probability at least $(1-\varepsilon)^2$, for all k. Alternatively, given access to two ancilla qubits and access only to uncontrolled U queries, the same outcome can be achieved with the same asymptotic complexity and success probability over the domain $\cos(\theta_k) \in [\delta, 1-\delta]$. Finally, in the case that the output legs are all ν-half twisted embeddable with $\cos(\varphi) \in [\gamma, \sqrt{1-\gamma^2}]$, the same can be done with no extra space and unit success probability using $\zeta = \widetilde{\mathcal{O}}(\delta^{-1}\gamma^{-2}\log^2(\varepsilon^{-1}))$ black-box calls to $\mathfrak{G}$ and the guarantee that $\nu \leq \nu''$, for a computable $\nu'' = \widetilde{\mathcal{O}}(\delta\gamma^2\varepsilon)$. Each protocol described above can be constructed efficiently and obliviously to the output unitaries of the gadget. □

Essentially this theorem (proven in Appx. A) applies Lem. 2.1 to each output leg of a gadget, obtaining our desired endpoint for this section: *snappable gadgets*. While the following definition is somewhat innocuous, in the following we show that such gadgets, when assembled, induce natural and simple-to-track algebraic manipulations of their achieved transforms.

Definition 2.5 (Snappable gadget)**.** A gadget is said to be ε-δ-*snappable* if each of its output legs, over all unitary inputs with the form of σ_x-rotations by an angle θ such that $\delta < |\cos(\theta)| < 1-\delta$, produces a unitary which is ε-close to a σ_x-rotation in norm. □

3 Composing QSP gadgets

The gadgets of Defs. 2.3 and 2.4 and the correction procedure of Thm. 2.1 bring us to *snappable gadgets*. Such gadgets can finally be freely assembled into composite gadgets to build complex, useful functional transforms; in this section we formally define this assembly, and enumerate its properties. The desired endpoint of this section, the statement of Thm. 3.2, shows a neat equivalence between a series of (partially) composed polynomials achieved by individual gadgets, and a structured network of those gadgets. This section begins by focusing on the linkages of these networks, captured in Thm. 3.1, such that Thm. 3.2 can be used to snap together a series of example gadgets with LEGO-like ease in Sec. 4.

As mentioned, we show in this section that gadgets permit a simple diagrammatic representation of their semantics, depicted in Fig. 1; in this way, complex functions can be built hierarchically out of simpler ones and visually reasoned about in an intuitive way. More specifically, we can think of so-called (a, b) gadgets as boxes with a *input legs*, representing oracular unitary inputs, and b *output legs*, representing output unitaries (These boxes are shown in (c) of Fig. 1 for $a = b = 2$). To flesh out this diagrammatic language, we present a theorem (Thm. 3.1) enumerating the valid ways to link gadgets together, and translate the effect of such compositions into statements on algebraic manipulations over

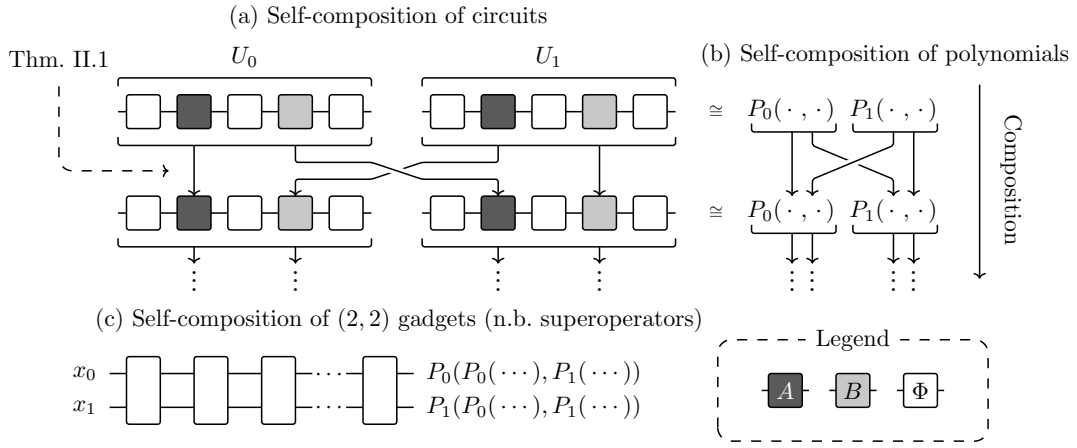


Figure 1: A gadget *Rosetta Stone* giving three depictions of the same self-composition of a $(2,2)$ gadget (Def. 2.3). Arrows indicate direct substitution, of (a) quantum circuits (time going left to right), (b) polynomials, and (c) gadget legs, where A, B are slots (places where an oracle unitary can be substituted) and Φ are programmable, generally distinct σ_z rotations. The (Laurent) polynomial transforms achieved by the two protocols, P_0, P_1, are real-valued with real arguments, and the correction protocol of Thm. 2.1 is implicitly interspersed between substitutions (represented explicitly by dots in Fig. 2). Note that the circuit (a) and functional (b) depictions are different from the *gadget* (c) depiction given, e.g., in Figs. 2 and 4 and elsewhere; this difference is detailed in Rem. I.2.

each gadget's achieved polynomial functions. Diagrammatically, linking gadgets is depicted by joining input and output legs. In this way, as exemplified in the *Rosetta Stone* diagram of Fig. 1, one can freely reason about assemblages of gadgets in terms of any among (a) the circuits realizing them, (b) the functional manipulations they achieve, or (c) as linkages between boxes with semantically simple input and output legs. While not discussed in this section explicitly, the syntax of gadget assemblages follows a formal grammar (Appx. I.3), while the semantics of achievable functional manipulations is described by the instantiation of a monadic type (Appx. I.2). We finally note briefly that depicting quantum superoperators as such acyclic graphs is not entirely new, as previous works considering quantum combs (circuits with open slots) have expressed them in terms of simple *causal networks* [12, 13, 38]; the difference between those works' general prescription and ours is that our networks depict semantic properties of interactions between achieved functional transforms, and we require careful constraints on allowed unitary inputs. For this reason, as well ease of description for higher-order operations over gadgets defined later, relying on causal networks rather than quantum combs to depict our superoperators leads to an expedient, clarified visual language.

We begin our description of gadget compositions with a definition.

Definition 3.1 (Interlink for gadgets)**.** Let $\mathfrak{G}$ and $\mathfrak{G}'$ be (a,b) and (c,d) gadgets respectively. An *interlink* between these gadgets specifies a way to validly connect them. Take $[b],[c]$ to be the length-b and length-c (zero-indexed) ordered lists of labels of the outputs of $\mathfrak{G}$ and the inputs of $\mathfrak{G}'$ respectively. An *interlink* is a three element list $\mathfrak{I} \equiv (B,C,W)$ with the following prescription: B is a sublist of $[b]$ of size $e \in \{0,1,\cdots,\min(b,c)\}$, C is a sublist of $[c]$ also of size e, and W is a member of S_e the permutation group over e elements. □

We can use this definition to precisely state the following theorem on the general combinations of gadgets. This theorem shows that gadgets can be composed in a nearly unconstrained way, snapped together like LEGOs, with well-understood associated cost and functional action.

Theorem 3.1 (Composing a gadget with an atomic gadget)**.** Let $\varepsilon, \delta > 0$, $\mathfrak{G}$ be an (a,b) gadget and (Ξ, S) an antisymmetric atomic (c,d) gadget, where $\Xi \equiv \{\Phi_0,\ldots,\Phi_{d-1}\}$ and $S \equiv \{s_0,\ldots,s_{d-1})$ for (Ξ,S). Suppose the gadget $\mathfrak{G}$ achieves

$$F(x) \equiv \{f_0(x_0,\cdots,x_{a-1}), f_1(x_0,\cdots,x_{a-1}),\ldots,f_{b-1}(x_0,\cdots,x_{a-1})\} \in (\mathbb{R}^a \to \mathbb{R}^b) \tag{5}$$

over $x \in [-1,1]^a$, and the atomic gadget (Ξ,S) achieves

$$G(y) \equiv \{g_0(y_0,\cdots,y_{c-1}), g_1(y_0,\cdots,y_{c-1}),\ldots,g_{d-1}(y_0,\cdots,y_{c-1})\} \in (\mathbb{R}^c \to \mathbb{R}^d) \tag{6}$$

over $y \in [-1,1]^c$. Let $\mathfrak{I} = (B, C, W)$ be an interlink between these gadgets. Then, there exists a gadget $\mathfrak{G}'$ which ε-approximately achieves

$$H(x, y') \equiv \bigcup_{k\in[d]} g_k \left(\bigcup_{j\in B} f_{W(j)}(x_0, \ldots, x_a) \cup \bigcup_{k\notin C} y_k \right) \cup \bigcup_{k\notin B} f_k(x_0, \ldots, x_a) \tag{7}$$

over $(F(x), y') \in \mathcal{D}$, where y' is the subset of y_k such that $k \notin C$ and $\mathcal{D}$ is a domain determined by the correction procedure utilized. The set union symbol is abused to mean concatenation of lists with respect to the pre-established order of the labels of the relevant input and output *lists* (not sets), and where $W(j)$ is the result of the application of the specified permutation applied to the *index* of the subset member j in the ordered sublist C of $[c]$. Moreover, $\mathfrak{G}'$ can be constructed efficiently, and uses only a description of (Ξ, S) and a total of $\widetilde{\mathcal{O}}(d|\Xi|_\infty \zeta)$ black-box calls to the *unitaries produced by running* $\mathfrak{G}$ to realize the function H.

Note ζ, the cost of correcting the gadget $\mathfrak{G}$, is precisely the cost to make $\mathfrak{G}$ *snappable*, and this snappability enables the simple compositional form of Eq. 7. ζ is one among two choices presented in Thm 2.1, with the variables ε and δ carrying over. Generally, $\zeta = \widetilde{O}(\text{polylog}(\varepsilon^{-1})\text{poly}(\delta^{-1}))$, with possible polynomial scaling in a parameter γ, given in Thm. 2.1, as well. This choice also dictates whether $\mathcal{O}(1)$ or zero ancilla qubits are required to perform this composition, and whether the success probability of achieving each individual function is at least $(1-\varepsilon)^2$, or 1. $|\Xi|_\infty$ is the maximum length of lists within Ξ. For a proof of this result, see Appx. A. □

Remark 3.1 (Arbitrary gadget interlinks)**.** While it is possible to consider interlinks between *arbitrary* gadgets, rather than assuming *a priori* that the second gadget is atomic, it isn't possible to make a general claim about the required complexity to implement such a composition. For a generic gadget, treated as a black box accepting and outputting unitaries, one cannot know the required query complexity of the input legs in order to achieve a given output, with some precision, without knowledge of exactly how the input legs are queried within the gadget: behaviour which can, in principle, vary dramatically between different gadgets. We do know this internal structure for the special case of atomic gadgets (they are collections of M-QSP circuits). Nevertheless, characterization of interlinks between gadgets and atomic gadgets allows for full characterizations of arbitrary "gadget networks" of many interlinks along with associated corrections (as an atomic gadget which has been corrected may not longer strictly be an atomic gadget, due to the possible introduction of ancilla qubits). As a result, Thm. 3.1 is sufficiently powerful for all practical purposes within the scope of this paper. □

Via Thm. 3.1, an interlink $\mathfrak{I}$ can be thought of as an operator over gadgets, mapping a pair of gadgets to a single gadget with a possibly new type

$$(a, b), (c, d) \to (a + c - e, b + d - e), \tag{8}$$

for any $e \in \{0, 1, \cdots, \min(b, c)\}$.

Remark 3.2. Given an interlink, $\mathfrak{I}$, a gadget $\mathfrak{G}$, and an antisymmetric atomic gadget (Ξ, S), we refer to the gadget $\mathfrak{G}'$ resulting from the above operation as $\mathfrak{I}[\mathfrak{G}, (\Xi, S)]$. Moreover, we can think of an interlink as inducing an operation of pairs of functions themselves. Going forward, given an interlink $\mathfrak{I}$, as well as functions F and G of the form of Eq. (5) and Eq. (6), we let $G \circ_{\mathfrak{I}} F$ denote the function H of Eq. (7). □

Graphically an interlink is simple to describe, as is shown in Fig. 2, which also depicts the general action described in Thm. 3.1. Beyond composition, one can define a variety of operators over gadgets; we note that, as these gadgets are themselves superoperators, we are free to duplicate and elide outputs at the possible cost of additional queries. As discussed in Appx. A, one can *augment* an (a, b) gadget to an $(a, b + c)$ gadget, as well as *elide* an (a, b) gadget to an $(a, b - c)$ gadget by ignoring outputs. Moreover, the input of an (a, b) gadget can be *pinned* to yield an $(a - c, b)$ gadget, or the output legs of an (a, b) gadget can be *permuted* to yield an (a, b) gadget. Finally, note that the correction protocol itself can be thought of as an operator over gadgets, leaving the function achieved by the gadget approximately unchanged, but transforming its unitary output by a σ_z conjugation. The form and cost of these basic operations is covered in Appx. A and, once generally computed, these rules allow the algorithmist to reason about complex, multi-input/output functional operations agnostic to their realizing circuits, in a *functional style*.

Before continuing, we note that defining the closure over arbitrarily interlinking finite-size gadgets is not a simple task. For instance, decomposing a given polynomial in a single-variable into an optimal

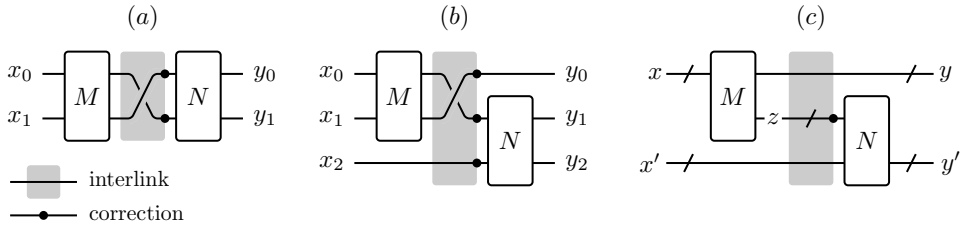


Figure 2: A depiction of gadgets coupled by interlinks (Def. 3.1) both in example (a-b), and generally (c). Given two gadgets, an interlink (B, C, W) specifies a subset of output legs B, input legs C, and a permutation W, joining gadgets according to Thm. 3.1. Two possible interlinks are shown explicitly in (a-b) for $(2, 2)$-gadgets. In (c), possibly many legs are suppressed in dashed legs, and the sets B, C in Thm. 3.1 are precisely those composing the composite leg z; in this way (c) captures the most general coupling of two gadgets, per Thm. 3.1. In all cases black dots indicate a correction protocol (2.1) necessary to properly couple legs.

tower of composed, lower degree polynomials (a strictly simpler problem) was once assumed to be a computationally hard problem [5, 18, 37], and the conditions under which such decomposition is efficient in the multivariable case have only recently been generally understood [22] and analyzed in the approximate setting [16, 17, 25], with the generic problem known to be NP-hard [19]. Nevertheless such decompositions, if they do exist, are basically unique (a consequence of Ritt's theorem [58]), and packages exist within most computer algebra systems for computing them. Consequently, while we show that we can achieve the full algebraic and monoidal operations natural to multivariable polynomials, our constructions only furnish competitive upper bounds on the required space, gate, and query complexity to achieve a given transform, with lower bounds depending on deep results in algorithmic complexity theory [8, 36, 65] and polynomial decomposition theorems not exposited here. Despite this fact, we are still able to provide a compact *equivalence theorem* for composite gadgets achieving polynomials which can be decomposed into a tower of lower-degree polynomials for which atomic gadgets are known.

Theorem 3.2 (Efficient equivalence of polynomial compositions and gadget assemblages)**.** Let the expression $\mathcal{L}(x_1, \ldots, x_n)$ denote the set of all polynomials achievable by atomic gadgets in the variables $x_1, \ldots, x_n$. Suppose $P(x_1, \ldots, x_n)$ is a polynomial of degree D and can be split into a tower of $m = \mathcal{O}(\log(D))$ *interlinked polynomials* (Rem. 3.2),

$$P = P^{(m-1)} \circ_{\mathfrak{I}_{m-2}} \left(P^{(m-2)} \circ_{\mathfrak{I}_{m-3}} \circ \left(\cdots \left(P^{(1)} \circ_{\mathfrak{I}_0} P^{(0)} \right) \cdots \right) \right) \tag{9}$$

such that $P_i^{(j)} \in \mathcal{L}(x_1, \ldots, x_n)$ for all i and j. Assume that the composition satisfies the domain condition (Rem. C.1) with P_k supported on $\mathcal{D}_k$ such that each $\mathcal{D}_k$ is separated from singular points ± 1 (and possibly 0) by some length $\delta \in \mathcal{O}(1)$ for all k. Then, there exists an assemblage of m atomic, snappable gadgets which ε-approximates the polynomial P on the domain yielded from the $\mathcal{D}_k$, with cost $\widetilde{O}(\mathrm{poly}(D)\,\mathrm{polylog}(\varepsilon))$. □

For a proof of this theorem, see Appx. A. We remark that the discussion of query and gate complexity in Thm. 3.1 and thus Thm. 3.2, while simple, does not immediately extend to composite gadgets with complex internal structure. For this purpose we introduce a variety of independently interesting mathematical objects which allow for expedient calculation of the cost of implementing a gadget which achieves a desired function up to a desired precision over a desired range of inputs. The basic object for such costs is a *cost matrix* (Def. C.1 in Appx. C), which permits one to determine the query complexity of a gadget with respect to a given input leg for a given output leg, up to a specified precision over a specified range of inputs. With a few simple abstractions, computing composite gadget costs is thus reduced to a lightly augmented form of matrix multiplication over sub-gadget cost matrices. Details of this calculation are presented in Appx. C.

4 Examples

The results of Sec. 3 together project a new path for building quantum subroutines: namely, the equivalence of Thm. 3.2 ensures that we can think purely in terms of the polynomials achieved by a series of gadgets when snapping them together visually as modular pieces, while this LEGO-like construction is governed behind the scenes in the circuit picture by the simple rules of Thm. 3.1. To highlight these principles together in action, we thus consider a series of linked examples making use of minimally complex

Algebraic structure	Manipulation	Algebraic form	Reference
QSP/M-QSP	Polynomial application	$x_0 \mapsto P(x_0)$,	Thm. D.1
	Multivariable variant†	$x_0, \ldots, x_n \mapsto P(x_0, \ldots, x_n)$,	Thm. D.2
Polynomial ring‡	Addition	$x_0, x_1 \mapsto (x_0 + x_1)/2$,	Thm. IV.3
	Multiplication	$x_0, x_1 \mapsto x_0 x_1$,	Thm. IV.2
	Scalar multiplication	$x_0 \mapsto \min(\alpha x_0, 1), \alpha \in \mathbb{R}$,	Ex. IV.4/5
Composition monoid	Polynomial composition	$f(x_0), g(x_0) \mapsto (f \circ g)(x_0)$,	Thm. III.1
General gadgets	Gadget linking	$\mathfrak{G}, \mathfrak{G}' \mapsto \mathfrak{G}\mathfrak{J}\mathfrak{G}'$,	Thm. III.1

Provided examples		
	Negation/inversion (Ex. IV.1)	Bandpass function (Ex. IV.11)
	Angle sum/difference (Ex. IV.2)	Majority vote (Ex. IV.12)
	Affine shift (Ex. IV.7)	Functional interpolation (Ex. IV.13)
	Step function (Ex. IV.8)	Chebyshev inverse (Thm. IV.1)
	General mean (Ex. IV.10)	

Figure 3: A summary of functional transforms achieved in this work, grouped (above the line) by their constitution of common mathematical structures over polynomials. Addition, polynomial multiplication, and scalar multiplication form a polynomial ring, where (‡) indicates we consider (possibly clipped to have a maximum norm of one) polynomials over $x \in [-1, 1]$ of definite parity taking real-values; these properties are preserved under the specified operations. We also achieve a monoid generated by single-variable polynomial composition, as well as its generalized form capturing tuples of multivariable polynomials, described by the grammar and language of gadgets (Appx. I). We stress that in the general case we cannot achieve these operations exactly, only to arbitrary precision and with an associated cost. These operations, together with the ability to generate arbitrary single-variable bounded, definite parity, real polynomials using QSP, and a special subset (†) of such polynomials with M-QSP [59], permit all highlighted examples of Sec. 4 (enumerated below the line).

pieces. We provide an overview of these examples in Fig. 3, grouping them in terms of the natural algebraic structures they constitute; in turn, these examples are exposited in Sec. 4.1, which covers common basic arithmetic, and Sec. 4.2, which combines these pieces to achieve a variety of familiar and utilitarian functional classes.

4.1 Basic arithmetic with gadgets

One of the most immediately useful applications of the protocols derived in the previous section is the ability to perform basic arithmetic and interpolation, coherently, between two polynomial functions, each achieved by an independently specified QSP protocol.

Example 4.1 (Inversion and negation)**.** We start with two simple $(1, 1)$ gadgets with utility in the creation of composite functions. Each take as input some $x_0 \in [-1, 1]$ as usual. The first, termed *inversion* pre-applies an iX (special unitary) to the oracle, producing $\sqrt{1 - x_0^2}$ which 'logically inverts' the input on $\{0, 1\}$. The second, termed *negation* conjugates the oracle by the (special unitary) iY, and achieves $-x_0$. Note that these are not *atomic gadgets* per Def. 2.3, as their form is outside that of M-QSP protocols, but that they produce embeddable output (Def. 2.2). □

Example 4.2 (Angle sum and difference)**.** To illustrate a useful but non-obviously antisymmetric gadget, we can consider the $(2, 1)$ gadget resulting from multiplying two oracles in different variables with QSP phases all zero, e.g., $U_0 U_1$, which achieves $x_0 x_1 - \sqrt{1 - x_0^2}\sqrt{1 - x_1^2}$. This corresponds to adding the angles associated with oracle unitaries. This can be changed to an angle difference by conjugating one oracle, e.g., $U_1 \mapsto e^{i\pi/4\sigma_z} U_1 e^{-i\pi/4\sigma_z}$. □

Example 4.3 (Multiplication)**.** Let $\mathfrak{G}_{\text{mult}} \equiv (\Xi, S)$ be the $(2, 1)$ atomic gadget with

$$\Xi \equiv \{\Phi\} = \{\{-\pi/4, \pi/4, -\pi/4, \pi/4\}\} \quad \text{and} \quad S = \{\{0, 1, 0\}\}. \tag{10}$$

This gadget achieves $f(x_0, x_1) = T_2(x_0) x_1$, which can be linked to gadget outputs to multiply functions (up to pre-application of $T_2(x) = 2x^2 - 1$, the second Chebyshev polynomial of the first kind, to the first argument). □

Example 4.4 (Sub-normalization)**.** Let (Ξ, S) be the $(2,1)$ atomic gadget of Ex. 4.3. Let $\mathfrak{G}_{\text{subnorm}}(a)$ be the $(1,1)$ gadget obtained from pinning (Def. A.1) the index-0 input leg to the unitary $U_0 = e^{i\arccos(x_0)\sigma_x}$ where $x_0 = \sqrt{(1+a)/2}$, for $a \in [-1,1]$. Then $\mathfrak{G}_{\text{subnorm}}(a)$ obtains the function $f(x) = T_2(\sqrt{(1+a)/2})x = ax$, for all $x \in [-1,1]$. □

Example 4.5 (Scaling)**.** Take the $(1,1)$ atypical gadget constructed in Ex. D.2, which achieves an arbitrary bounded real polynomial P of definite parity. Consider the polynomial approximation of P from Lemma 30 of [26], which ε-approximates the linear function ax for $a > 1$ on the interval $x \in [(-1+\delta)/a, (1-\delta)/a]$. Then the QSP phases Φ of the protocol achieving P can be inserted into the prescription of Ex. D.2 to yield a $(1,1)$ gadget which multiplies its input by a scalar greater than 1 with a clipped output: $\mathfrak{G}_{\text{scale}}(a)$. Moreover, the cost of this gadget is $\mathcal{O}([a/\delta]\log[a/\varepsilon])$ queries to x_0, by [26], and requires no additional space. □

Example 4.6 (Addition)**.** Let $\mathfrak{G}_{\text{add}} \equiv (\Xi, S)$ be the $(2,1)$ atomic gadget with

$$\Xi \equiv \{\Phi\} = \{0, \pi/4, 0, -\pi/4, 0\} \quad \text{and} \quad S = \{0, 1, 1, 0\} \tag{11}$$

This protocol will achieve the function $f(x_0, x_1) = T_2(x_0)T_2(x_1)$ for $x_0, x_1 \in [-1,1]$. Let U_0 and U_1 be embeddable, with $U_0 = e^{i\arccos(x_0)\sigma_x}$ and $U_1 = e^{i\arccos(x_1)\sigma_x}$. The products

$$V_0 = U_0 U_1 = e^{i(\arccos(x_0)+\arccos(x_1))\sigma_x}, \tag{12}$$

$$V_1 = U_0 e^{i\pi\sigma_z/2} U_1 e^{-i\sigma_x\pi/2} = U_0 U_1^\dagger = e^{i(\arccos(x_0)-\arccos(x_1))\sigma_x}, \tag{13}$$

define simple $(2,1)$ gadgets. We can achieve a $(2,1)$ gadget by first duplicating the input legs corresponding to U_0 and U_1, passing a pair (U_0, U_1) into each of the V_0 and V_1 gadgets, then passing these two output legs into (Ξ, S). However, note that V_0 and V_1, while being σ_x-rotations, can generally have their rotation angle in $[-\pi, \pi]$, rather than $[0, \pi]$. Thus, either V_0 is embeddable *or* $\sigma_z V_0 \sigma_z$ is embeddable, with the same holding true for V_1. However, as can be easily checked, this σ_z-ambiguity will not matter: these extra σ_z-rotations (when absorbed into the phase sequence Φ as extra $\pi/2$-rotations) will have no effect on the output polynomial: this gadget will always achieve

$$f(x_0, x_1) = T_2\left(\cos(\arccos(x_0) + \arccos(x_1))\right) T_2\left(\cos(\arccos(x_0) - \arccos(x_1))\right) \tag{14}$$

$$= \cos\left(2(\arccos(x_0) + \arccos(x_1))\right) \cos\left(2(\arccos(x_0) - \arccos(x_1))\right) \tag{15}$$

$$= \frac{\cos(4\arccos(x_0)) + \cos(4\arccos(x_1))}{2} = \frac{T_4(x_0) + T_4(x_1)}{2}. \tag{16}$$

See Fig. 4 for a graphical depiction of the interlinks constituting this gadget. □

Example 4.7 (Constant shift)**.** It is possible to pin the addition gadget to perform a "constant shift" (up to the fourth Chebyshev polynomial): $x \mapsto \frac{1}{2}T_4(x) + b$ for $b \in [-1/2, 1/2]$. This gadget, denoted $\mathfrak{G}_{\text{shift}}(b)$, is achieved by taking the gadget of Ex. 4.6 and pinning its second input (Def. A.1) to $e^{i\arccos(x_1)\sigma_x}$, with $T_4(x_1) = 2b$. Note it is possible to combine the sub-normalization and constant shift protocols into an *affine shift* of the form $x \mapsto \frac{1}{2}T_4(ax) + b$. □

The above examples, implementing basic arithmetic, have the constraint that they apply a Chebyshev polynomial to at least one of the inputs. As a final point, note that all of the provided examples can be rid of the inclusion of Chebyshev polynomials (approximately), via composition with the inverse Chebyshev protocol of Thm. 4.1, stated and proved below.

Theorem 4.1 (Inverses of Chebyshev polynomials)**.** Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists $(1,1)$ gadgets of depth and cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieve functions $T_{2^n}^{-1}(x)$ (right-inverses) for $x \in [-1+\delta, 1-\delta]$, using a single qubit (no ancillae). □

For a proof of this result, see Appx. H. Using this protocol (in particular, by interlinking the inverse Chebyshev gadget with the arithmetic gadgets), it is possible to realize the following transformations of the functions achieved above, approximately, on restricted domains:

$$f(x_0, x_1) = T_2(x_0)x_1 \implies f'(x_0, x_1) = x_0 x_1, \tag{17}$$

$$f(x_0, x_1) = \frac{T_4(x_0) + T_4(x_1)}{2} \implies f'(x_0, x_1) = \frac{x_0 + x_1}{2}. \tag{18}$$

This, of course, will come at the expense of added circuit depth. To this end, we have the following theorems.

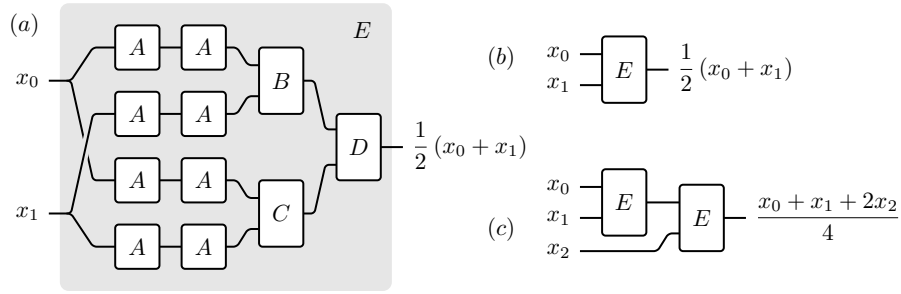


Figure 4: Component breakdown of the sum gadget (Ex. 4.6). The sum gadget (a) is achieved by the composition of multiple $(1,1)$ and $(2,1)$ gadgets as shown, specifically A (the square root gadget from Thm. 4.1 with T_2), B (angle sum gadget, Ex. 4.2), C (angle subtraction gadget, Ex. 4.2), and D (product gadget, Ex. 4.3). The achieved composite operation (b) can be treated as a $(2,1)$ gadget labeled by E, whose partial composition with itself (c) has the expected semantic behavior.

Theorem 4.2 (Arbitrary approximate multiplication)**.** Given $0 < \delta \leq 1$ an $0 < \varepsilon \leq 1/2$, there exists a $(2,1)$ gadget of depth/cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieves the function $f(x_0, x_1) = x_0 x_1$ for all $x_0 \in [-1+\delta, 1-\delta]$ and $x_1 \in [-1, 1]$ using no ancilla qubits. □

Theorem 4.3 (Arbitrary approximate addition)**.** Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists a $(2,1)$ gadget of depth/cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieves the function $f(x_0, x_1) = (x_0 + x_1)/2$ for all $x_0, x_1 \in [-1+\delta, 1-\delta]$ using no ancilla qubits. □

We refer to Appx. H for proofs of these results, and more details about the cost of performing these transformations. With these basic operations instantiated, it is now easy to see how Thm. 3.2 immediately implies that all multivariable polynomials up to bound and parity constraints are approximately achieveable by *some* composite gadget mirroring the structure of the algebraic manipulations achieved. While the question of the *most efficient* gadget realization is more difficult, we can freely use the gadgets above sparingly in the next section to instantiate a series of familiar and useful functions.

4.2 Familiar multivariable functions from gadgets

Having highlighted basic examples of arithmetic operations with respect to two variables, we leverage these operations to construct more complicated functional transformations. We begin with a short remark.

Remark 4.1 (Algebra of gadget operations)**.** Given the operations of addition and multiplication presented in Eq. (17), Eq. (18), as well as the scaling operations of Ex. 4.4 and Ex. 4.5, and the general ability to compose gadgets arbitrarily, it is possible to achieve approximations of arbitrary multivariate polynomials on mildly restricted domains. This is a somewhat surprising result in and of itself: sufficiently high-depth M-QSP protocols (up to the possible use of ancilla qubits during correction) can approximately achieve all multivariate polynomials, modulo certain constraints imposed in the examples of Sec. 4.1. In general, these naïve, "ground-up" constructions of polynomials will have poor scaling, due to the need to perform the highly non-linear inverse Chebyshev and corrective protocols. Nevertheless, there are also other examples of short protocols which achieve useful transformations *without* the added depth of performing square roots or many nested correction protocols. This fact is one of the fundamental takeaways of the following results we wish to emphasize: while it is *possible* to build all transformations from the repeated composition of only a few low-degree gadgets, in order to achieve experimentally reasonable protocols, it is much more advantageous to use such basic gadgets sparingly, relying on functions efficiently achieved by standard M-QSP protocols where possible. □

To begin our discussion of more advanced examples, we present a brief discussion of techniques for constructively achieving step and bandpass functions through gadget composition.

Example 4.8 (Basic step function of [48])**.** The protocol introduced in [48], refered to as *r-QSP* is an example of an n-fold composition of $(1,1)$ atomic gadgets. In particular, the authors (implicitly) construct a family of $(1,1)$ atomic gadgets $\mathfrak{G}^{(\ell)}_{\text{step}} \equiv (\Xi^{(\ell)}, S^{(\ell)}) = (\Phi^{(\ell)}_{\text{step}}, \{0, \ldots, 0\})$ with $|\Phi^{(\ell)}_{\text{step}}| = 2\ell + 1$ such that the n_ℓ-fold full composition (Thm. 3.1) $\mathfrak{G}^{(\ell)}_{\text{step}} \circ \cdots \circ \mathfrak{G}^{(\ell)}_{\text{step}}$ achieves an ε-approximation of the function

$f(x) = \text{sign}(x)$ on the interval $[-1, -\delta] \cup [\delta, 1]$ when the length of the resulting gadget, $\zeta = (2\ell+1)^{n_\ell}$, satisfies

$$\zeta = \mathcal{O}\left(\delta^{-2(\nu_\ell+1)} \log^{1+\nu_\ell}\left(\varepsilon^{-1}\right)\right), \quad \nu_\ell = \frac{\log(2\ell+1)}{\log(\ell+1)} - 1. \tag{19}$$

□

Making use of the basic step function construction of [48], as well as the arithmetic operations of the previous section, it is possible to construct a simple, infinite family of bandpass filters.

Example 4.9 (Simple bandpass family)**.** Given $a \in [\delta, 1/\sqrt{2}]$, there exists a family of $(1,1)$ gadgets $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$, such that $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$ ε-approximately achieves a function f satisfying $|f| \leq 1$ as well as

$$f(x) \in \begin{cases} [1-\varepsilon, 1] & \text{if } x \in [-a+\delta, a-\delta] \\ [-1, -1+\varepsilon] & \text{if } x \in \left[-\frac{1}{\sqrt{2}}, -a-\delta\right] \cup \left[a+\delta, \frac{1}{\sqrt{2}}\right], \end{cases} \tag{20}$$

where the total depth of the gadget, ζ', satisfies

$$\zeta' = \mathcal{O}\left((2a\delta)^{-4(\nu_\ell+1)} \log^{1+\nu_\ell}\left(\varepsilon^{-1}\right)\right). \tag{21}$$

Moreover, given the phases of $\Phi^{(\ell)}_{\text{step}}$, a sequence of Ex. 4.8, the gadget $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$ can be described analytically, in terms of these phases. □

For a proof of this result, see Appx. H. This example provide good illustration of the fundamental tension between the generality of the classes of functions that can be achieved via composition of gadgets, and the asymptotic scaling of resource requirements. Here, we have shown that it is possible to achieve a *restricted* class of bandpass filters via the gadget formalism. It is, generally speaking, possible to drop the restrictions of this example (in particular, the constraint that the function approximately achieves only the values ± 1), at the expense of more queries/corrective protocol applications (see Ex. 4.11).

There are many other examples of low-depth protocols which can be achieved via gadget composition, and the simple arithmetic operations discussed previously. We highlight some of these examples below.

Example 4.10 (2^n mean)**.** Let $\mathfrak{G}$ be the $(2,1)$ gadget achieving $(x_0 + x_1)/2$. Then one can build a composite $(2^n, 1)$ gadget $\mathfrak{G}'$, as well as the correction protocol, achieving $2^{-n}(x_0 + x_1 + \cdots + x_{2^n-1})$ using $(2^n - 1)$ instances of $\mathfrak{G}$ and successively pooling pairs of variables. E.g., $(x_0 + x_1)/2$ and $(x_2 + x_3)/2$ are taken to their average $(x_0 + x_1 + x_2 + x_3)/4$. □

The cost of implementing gadgets will depend on the range of input parameters for which it must output an ε-approximation of the desired function. Moreover, the cost will depend on whether the ancilla or ancilla-free gadget correction protocol is used, which will vary on a case-by-case basis. As an illustrative example, we provide a detailed discussion of the complexity of implementing the 2^n-gadget, and discuss how its cost differs from the cost LCU, in Appx. H. In addition to this example, we make note of some other gadgets, which are, in principle, possible to achieve. The complexity analysis of each, and how it compares to other techniques such as LCU, carries forward in the same manner of casework as is presented in the example of Appx. H.

Example 4.11 (Arbitrary one-dimensional bandpass functions)**.** Consider $\mathfrak{G}, \mathfrak{G}'$ two $(1,1)$ gadgets approximately achieving sign functions $\Theta(x_0 - a_0), \Theta(x_0 - a_1)$ (Ex. 4.8), where $\Theta(x_0)$ takes the value -1 for $x < 0$ and 1 for $x > 0$ on the interval $[-1, 1]$. Without loss of generality assume $a_0 < a_1$; then the averaging gadget applied to $\mathfrak{G}$ and $\mathfrak{G}''$ (the modified gadget achieving $-\Theta(a_1)$, enabled by Ex. 4.1) achieves an approximation to the following ideal bandpass function: $\Theta'(x_0) \equiv [\Theta(x_0 - a_0) - \Theta(x_0 - a_1)]/2$.which for $x_0 < a_0$ achieves 0, for $a_0 < x < a_1$ achieves 1, and for $x > a_1$ again achieves zero. □

Example 4.12 (Majority vote)**.** Majority vote among 2^n elements follows directly from the previous examples; let $\mathfrak{G}$ the $(2^n, 1)$ gadget of Ex. 4.10 and $\mathfrak{G}'$ the $(1,1)$ gadget thresholding at $x_0 = 1/2$ (Ex. 4.11). Then the simple composition $\mathfrak{G}' \circ \mathfrak{G}$ is a $(2^n, 1)$ gadget satisfying the desired properties. □

Example 4.13 (Subnormalized functional interpolation)**.** There exists a $(3,1)$ gadget that approximately achieves the function $f(x_0, x_1, x_2) = [x_0 x_2 + x_1\sqrt{1 - x_2^2}]/2$. In other words, depending on x_2, the gadget smoothly interpolates between the sub-normalized variables x_0 and x_1. If x_0, x_1 result from previous gadgets, this realizes a (sub-normalized) interpolation between two functions. Construction follows from previous gadgets. Namely the products $x_0 x_2$ and $x_1\sqrt{1 - x_2^2}$ are realized by Ex. 4.3, with $\sqrt{1 - x^2}$ achieved from x_2 by Ex. 4.1. Finally, both results are summed by Ex. 4.6. □

The cost analysis of these gadgets carries forward similarly to the example highlighted in Appx. H, depending on which corrective protocol is chosen.

5 Discussion and conclusion

At a practical level this work gives a modular construction, rooted in QSP and QSVT, for achieving highly expressive multivariable block encodings for commuting linear operators. This construction relies on special *corrective* subroutines which restore the *embeddability* (Def. 2.2) of QSP- and QSVT-like protocols in a black-box way. Recovering embeddability allows assemblages of such protocols, which can take many unitary oracles as input and produce many unitary outputs, to be simply wired together *at the level of the functions they apply* to matrix-invariant quantities; that this is a true modular process is demonstrated in the statements of the work's main theorems: Thms. 3.1 and 3.2, which indicate how to intersperse independently defined correction protocols (Thm. 2.1) between modules. Such modules are named *gadgets* (Def. 2.4), and their valid combinations are shown to follow a simple grammar, and generate both the natural commutative ring and composition monoid over multivariable polynomials. In the terminology of programming language theory, the modularity of gadgets is formalized in identifying them as *monadic types*, with the correction protocol furnishing a previously missing component of the monadic function *bind* (Appx. I).

We can phrase our results at three level, in order of increasing burden of proof and abstractness. **(a)** We provide improved, constructive upper bounds on the resource requirements (space, gate, and query complexity) of multivariable polynomial block encodings, and make comparisons to previously known methods (see Appx. B), with inspiration from older results in modular classical filter design [34, 61] and composite pulse sequences [32, 39, 46]. **(b)** We provide methods for interpretable constructions of coherent (quantum) control flow, with no obvious incoherent counterpart. Specifically, we work in a coherent-access model, analogous to that of recent work on the benefit of measurement-free oracular access [3, 30], and show that the action of QSVT-like protocols can be coherently conditioned on or coupled with the action of other QSVT-like protocols, *simultaneously within invariant subspaces* related to the singular values and vectors of (possibly many) block-encoded operators. **(c)** We apply methods in functional programming, type theory, and category theory to re-situate QSP and QSVT as quantum *types*, where our results instantiate a *monadic type* over real-valued, real-argument multivariable polynomials (see Appx. I). Our methods thus enable high-level quantum algorithm design, generalizing the work of [60] and providing a simpler path for bootstrapping M-QSP [59] to achieve wider and less constrained multivariable functional transforms.

Our construction of gadgets couples to the design of quantum algorithms on two levels, inspired by a natural division in formal and natural languages: semantics and syntax [35, 62, 63]. On the first, gadgets specify which polynomial transforms can be treated functionally, serving as basic units of computation; formally this is summarized in the instantiation of a monad, and characterizes the basic *semantics* of the resulting system. At the second level, we define a formal grammar enumerating valid ways to connect gadgets, which characterizes *syntax* of our system. While the ultimate relation between our construction's syntax and semantics is more complicated than this simple division implies (depending on various hybrid structures such as attribute grammars [35]) these two levels help to situate this work in classical programming language theory, where division into semantics and syntax is considered a first step along the path of parsing, optimizing over, and automatically generating programs for concrete tasks under concrete constraints.

To restate a crucial point, this work considers gadgets comprising QSP/QSVT protocols that can be linked, following intermediary processing, to other gadgets *such that* their embedded functional transforms are combined in a *semantically clear way*. This intermediary processing is generally necessary, non-trivial, and strictly subsumes the single-variable case [60], which itself subsumed various techniques in the recursive application of quantum algorithms [32, 39, 46]. The key benefit of our method is that each gadget inherits the approximative efficiency and ancilla-efficient character of QSP, meaning that the complexity of achieving a desired polynomial transform is no longer coupled directly to term number, polynomial degree, or polynomial norm, as in LCU or standard QSVT [52], but instead sophisticated results in the theory of multivariable polynomial decomposition [5, 19, 22, 37], and multivariable quantum signal processing (discussed in detail in Appx. B). In turn, this allows new questions in the algorithmic complexity (or Solomonoff–Kolmogorov–Chaitin complexity) [8, 36, 65] of certain approximate functional transforms under the composition of gadgets, for which our results provide an upper bound. The benefit of this approach to quantum algorithm design is multiform: numerical (given our ability to re-apply pre-computed gadgets elsewhere), experimental (given the reduced number of distinct QSP phases to compute), conceptual (given the ability to reason at the level of abstracted, useful quantum subroutines), and finally pedagogical (given the treatment of disparate quantum algorithms through a single lens of block-encoding manipulations).

We stress, however, that there remain significant open questions on the optimality of method offered here, especially in light of the dependence of a gadget's complexity on sophisticated results in the theory of polynomial decompositions [5, 19, 22, 37]. Consequently existing methods for multivariable block encoding, such as LCU, continue to offer diverse, context-specific utility; in fact, as we showed, LCU offers one method for instantiating gadgets and thus provides an upper bound on the complexity of any given multivariable transformation—critically, for a wide class of functions, summarized in Fig. 3, we show that we can improve on these LCU-provided bounds arbitrarily, recovering the constant-space and improved infinity-norm dependence of QSP and QSVT. Investigating richer classes of target functions for which robust separations in space and query complexity can be shown between gadgets and LCU is a promising avenue for future work.

Whether viewed in terms of highly-efficient block encodings, or as the instantiation of a monadic type [71–73] over higher-order quantum processes composable in accordance to a simple grammar, this work instantiates new and expressive QSP/QSVT-based quantum algorithms, with function-first interpretability and a declarative style. To support this claim, we provide multiple examples and an associated code repository allowing for the automated construction and analysis of gadgets (located at https://github.com/ichuang/pyqsp/tree/beta). In that repository we provide block encodings of algorithmically useful multivariable polynomials with improved query and space complexity over alternative methods for non-trivial classes of target functions, contained in an extensive test suite. We hope that these concrete numerical tools lower the barrier toward further experimentation—e.g., we noted in this work that the ability to semantically combine QSP/QSVT as modules in a black-box way relied strongly on a restricted circuit form, leaving the question of the existence of larger families of (less) structured parameterized quantum circuits for which functional programming abstractions can still be built.

6 Acknowledgements

Z.R. was supported in part by the NSF EPiQC program. J.C. thanks Prof. Nathan Wiebe for valuable conversations, and for catalyzing and funding an internship during the summer of 2023 at MIT, where the majority of the work on this project was conducted. J.C. also thanks the MIT Research Laboratory of Electronics (RLE) for their hospitality during his time at MIT. I.C. was supported in part by the U.S. DoE, Office of Science, National Quantum Information Science Research Centers, and Co-design Center for Quantum Advantage (C2QA) under contract number DE-SC0012704.

References

[1] Scott Aaronson. Read the fine print. *Nat. Phys.*, 11(4):291–293, 2015. URL https://doi.org/10.1038/nphys3272.

[2] Scott Aaronson. How much structure is needed for huge quantum speedups? *arXiv preprint, arXiv:2209.06930*, 2022. URL https://doi.org/10.48550/arXiv.2209.06930.

[3] Dorit Aharonov, Jordan Cotler, and Xiao-Liang Qi. Quantum algorithmic measurement. *Nat. Commun*, 13(1), 2022. DOI: 10.1038/s41467-021-27922-0. URL https://doi.org/10.1038%2Fs41467-021-27922-0.

[4] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A*, 52:3457–3467, Nov 1995. DOI: 10.1103/PhysRevA.52.3457. URL https://doi.org/10.1103/PhysRevA.52.3457.

[5] David R. Barton and Richard Zippel. Polynomial decomposition algorithms. *J. Symb. Comput.*, 1 (2):159–168, 1985. URL https://doi.org/10.1016/S0747-7171(85)80012-2.

[6] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, 2015. DOI: 10.1109/FOCS.2015.54. URL https://doi.org/10.1109/FOCS.2015.54.

[7] Yonah Borns-Weil, Tahsin Saffat, and Zachary Stier. A quantum algorithm for functions of multiple commuting Hermitian matrices. *arXiv preprint, arXiv:2302.11139*, 2023. URL http://dx.doi.org/10.48550/arXiv.2302.11139.

[8] Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969. ISSN 0004-5411. DOI: 10.1145/321526.321530. URL https://doi.org/10.1145/321526.321530.

[9] Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. Finding angles for quantum signal processing with machine precision. *arXiv preprint, arXiv:2003.02831*, 2020. URL https://doi.org/10.48550/arXiv.2003.02831.

[10] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 387–400, NY, USA, 2020. ACM. DOI: 10.1145/3357713.3384314. URL https://doi.org/10.1145/3357713.3384314.

[11] Andrew M. Childs and Wim van Dam. Quantum algorithms for algebraic problems. *Rev. Mod. Phys.*, 82:1–52, 2010. DOI: 10.1103/RevModPhys.82.1. URL https://doi.org/10.1103/RevModPhys.82.1.

[12] G. Chiribella, G. M. D'Ariano, and P. Perinotti. Quantum circuit architecture. *Phys. Rev. Lett.*, 101:060401, 2008. DOI: 10.1103/PhysRevLett.101.060401. URL https://doi.org/10.1103/PhysRevLett.101.060401.

[13] Giulio Chiribella, Giacomo Mauro D'Ariano, and Paolo Perinotti. Theoretical framework for quantum networks. *Phys. Rev. A*, 80:022339, 2009. DOI: 10.1103/PhysRevA.80.022339. URL https://doi.org/10.1103/PhysRevA.80.022339.

[14] N. Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 2(3):113–124, 1956. DOI: 10.1109/TIT.1956.1056813. URL https://doi.org/10.1109/TIT.1956.1056813.

[15] B. Claudon, J. Zylberman, C. Feniou, F. Debbasch, A. Peruzzo, and J. Piquemal. Polylogarithmic-depth controlled-NOT gates without ancilla qubits. *Nat. Comm.*, 15, 2024. URL https://doi.org/10.1038/s41467-024-50065-x.

[16] Robert M. Corless, Mark W. Giesbrecht, David J. Jeffrey, and Stephen M. Watt. Approximate polynomial decomposition. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*, ISSAC '99, pages 213–219, New York, NY, USA, 1999. Association for Computing Machinery. DOI: 10.1145/309831.309939. URL https://doi.org/10.1145/309831.309939.

[17] Sefa Demirtas, Guolong Su, and Alan V. Oppenheim. Exact and approximate polynomial decomposition methods for signal processing applications. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5373–5377, 2013. DOI: 10.1109/ICASSP.2013.6638689. URL https://doi.org/10.1109/ICASSP.2013.6638689.

[18] Matthew T Dickerson. *The functional decomposition of polynomials*. PhD thesis, Cornell University, 1989. URL https://dl.acm.org/doi/10.5555/866387.

[19] Matthew T Dickerson. General polynomial decomposition and the s-1-decomposition are NP-hard. *Int. J. Found. Comput. Sci.*, 4(02):147–156, 1993. DOI: 10.1142/S0129054193000109. URL https://doi.org/10.1142/S0129054193000109.

[20] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. Efficient phase-factor evaluation in quantum signal processing. *Phys. Rev. A*, 103(4), 2021. ISSN 2469-9934. DOI: 10.1103/physreva.103.042419. URL https://doi.org/10.1103/PhysRevA.103.042419.

[21] Yulong Dong, Lin Lin, Hongkang Ni, and Jiasu Wang. Infinite quantum signal processing. *Quantum*, 8:1558, December 2024. ISSN 2521-327X. DOI: 10.22331/q-2024-12-10-1558. URL https://doi.org/10.22331/q-2024-12-10-1558.

[22] Jean-Charles Faugère and Ludovic Perret. An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography. *J. Symb. Comput.*, 44(12):1676–1689, 2009. URL http://dx.doi.org/10.1016/j.jsc.2008.02.005.

[23] Simon J. Gay. Quantum programming languages: survey and bibliography. *Math. Struct.*, 16(4): 581–600, 2006. ISSN 0960-1295. DOI: 10.1017/S0960129506005378. URL https://doi.org/10.1017/S0960129506005378.

[24] J. Geronimo and Hugo Woerdeman. Positive extensions, Fejér-Riesz factorization and autoregressive filters in two variables. *Ann. Math.*, 160:839–906, Nov 2004. DOI: 10.4007/annals.2004.160.839. URL https://doi.org/10.4007/annals.2004.160.839.

[25] Mark Giesbrecht and John May. New algorithms for exact and approximate polynomial decomposition. In *Symbolic-Numeric Computation*, pages 99–112. Springer, 2007. URL https://doi.org/10.1007/978-3-7643-7984-1_7.

[26] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 2019. DOI: 10.1145/3313276.3316366. URL http://dx.doi.org/10.1145/3313276.3316366.

[27] Jonathan Grattage. A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, LICS '05, page 249–258, USA, 2005. IEEE Computer Society. ISBN 0769522661. DOI: 10.1109/LICS.2005.1. URL https://doi.org/10.1109/LICS.2005.1.

[28] Lov K Grover. Fixed-point quantum search. *Phys. Rev. Lett.*, 95(15):150501, 2005. DOI: 10.1103/PhysRevLett.95.150501. URL https://doi.org/10.1103/PhysRevLett.95.150501.

[29] Jeongwan Haah. Product decomposition of periodic functions in quantum signal processing. *Quantum*, 3:190, Oct 2019. ISSN 2521-327X. DOI: 10.22331/q-2019-10-07-190. URL http://dx.doi.org/10.22331/q-2019-10-07-190.

[30] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, et al. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 2022. URL https://doi.org/10.1126/science.abn7293.

[31] Sami Husain, Minaru Kawamura, and Jonathan A Jones. Further analysis of some symmetric and antisymmetric composite pulses for tackling pulse strength errors. *J. Magn. Reson.*, 230:145–154, 2013. URL https://doi.org/10.1016/j.jmr.2013.02.007.

[32] Jonathan A Jones. Nested composite NOT gates for quantum computation. *Phys. Lett. A*, 377(40): 2860–2862, 2013. URL https://doi.org/10.1016/j.physleta.2013.08.040.

[33] Camille Jordan. Essai sur la géométrie à n dimensions. *Bulletin de la Société mathématique de France*, 3:103–174, 1875.

[34] J Kaiser and R Hamming. Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(5):415–422, 1977. URL https://doi.org/10.1109/TASSP.1977.1162980.

[35] Donald E Knuth. Semantics of context-free languages. *Math. Syst. Theory*, 2(2):127–145, 1968. URL https://doi.org/10.1007/BF01702865.

[36] A. N. Kolmogorov. Three approaches to the definition of "the quantity of information". *Problemy peredachi informatsii*, 1(1):3–11, 1965. URL https://doi.org/10.1080/00207166808803030.

[37] Dexter Kozen and Susan Landau. Polynomial decomposition algorithms. *J. Symb. Comput.*, 7(5): 445–456, 1989. URL https://doi.org/10.1016/S0747-7171(89)80027-6.

[38] Dennis Kretschmann and Reinhard F Werner. Quantum channels with memory. *PRA*, 72(6):062323, 2005. URL https://doi.org/10.1103/PhysRevA.72.062323.

[39] Elica Kyoseva and Nikolay V Vitanov. Arbitrarily accurate passband composite pulses for dynamical suppression of amplitude noise. *Phys. Rev. A*, 88(6):063410, 2013. URL https://doi.org/10.1103/PhysRevA.88.063410.

[40] Monica Lam, Ravi Sethi, Jeffrey D Ullman, and Alfred Aho. Compilers: principles, techniques, and tools. *Pearson Education*, 2006.

[41] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

[42] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer New York, NY, 2nd edition, 1978. URL https://doi.org/10.1007/978-1-4757-4721-8.

[43] G. H. Low and I. L. Chuang. Optimal Hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, 2017. DOI: 10.1103/PhysRevLett.118.010501. URL https://doi.org/10.1103/PhysRevLett.118.010501.

[44] G. H. Low and I. L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019. DOI: 10.22331/q-2019-07-12-163. URL http://dx.doi.org/10.22331/q-2019-07-12-163.

[45] G. H. Low, T. J. Yoder, and I. L. Chuang. Methodology of resonant equiangular composite quantum gates. *Phys. Rev. X*, 6:041067, 2016. DOI: 10.1103/PhysRevX.6.041067. URL https://doi.org/10.1103/PhysRevX.6.041067.

[46] Guang Hao Low, Theodore J Yoder, and Isaac L Chuang. Optimal arbitrarily accurate composite pulse sequences. *Phys. Rev. A*, 89(2):022341, 2014. URL https://doi.org/10.1103/PhysRevA.89.022341.

[47] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4):040203, 2021. URL https://doi.org/10.1103/PRXQuantum.2.040203.

[48] Kaoru Mizuta and Keisuke Fujii. Recursive quantum eigenvalue and singular-value transformation: Analytic construction of matrix sign function by Newton iteration. *Phys. Rev. Res.*, 6:L012007, 2024. DOI: 10.1103/PhysRevResearch.6.L012007. URL https://doi.org/10.1103/PhysRevResearch.6.L012007.

[49] Eugenio Moggi. *Computational lambda-calculus and monads*. University of Edinburgh, 1988.
[50] Eugenio Moggi. *An abstract view of programming languages*. University of Edinburgh, 1989.
[51] Ashley Montanaro. Quantum algorithms: an overview. *Npj Quantum Inf.*, 2(1):1–8, 2016. URL https://doi.org/10.1038/npjqi.2015.23.
[52] Danial Motlagh and Nathan Wiebe. Generalized quantum signal processing. *PRX Quantum*, 5:020368, 2024. DOI: 10.1103/PRXQuantum.5.020368. URL https://doi.org/10.1103/PRXQuantum.5.020368.
[53] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, USA, 10th edition, 2011. ISBN 1107002176. URL https://doi.org/10.1017/CBO9780511976667.
[54] Tatsuki Odake, Hlér Kristjánsson, Akihito Soeda, and Mio Murao. Higher-order quantum transformations of Hamiltonian dynamics. *Phys. Rev. Res.*, 6:L012063, 2024. DOI: 10.1103/PhysRevResearch.6.L012063. URL https://doi.org/10.1103/PhysRevResearch.6.L012063.
[55] Michele Pagani, Peter Selinger, and Benoît Valiron. Applying quantitative semantics to higher-order quantum computing. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 647–658, 2014. URL https://doi.org/10.1145/2578855.2535879.
[56] George Polya and Gabor Szegö. *Problems and Theorems in Analysis II*. Springer, Berlin, 1998. URL https://doi.org/10.1007/978-3-642-61905-2.
[57] Oded Regev. Fast amplification of QMA (lecture notes), 2006. https://cims.nyu.edu/~regev/teaching/quantum_fall_2005/ln/qma.pdf.
[58] Joseph Fels Ritt. Prime and composite polynomials. *Trans. Amer. Math. Soc.*, 23(1):51–66, 1922. URL https://doi.org/10.1090/S0002-9947-1922-1501189-9.
[59] Zane M. Rossi and Isaac L. Chuang. Multivariable quantum signal processing (M-QSP): prophecies of the two-headed oracle. *Quantum*, 6:811, Sep 2022. DOI: 10.22331/q-2022-09-20-811. URL https://doi.org/10.22331%2Fq-2022-09-20-811.
[60] Zane M. Rossi and Isaac L. Chuang. Semantic embedding for quantum algorithms. *Journal of Mathematical Physics*, 64(12):122202, 12 2023. ISSN 0022-2488. DOI: 10.1063/5.0160910. URL https://doi.org/10.1063/5.0160910.
[61] Tapio Saramaki. Design of FIR filters as a tapped cascaded interconnection of identical subfilters. *IEEE Trans. Circuits Syst.*, 34(9):1011–1029, 1987. URL https://doi.org/10.1109/TCS.1987.1086263.
[62] Peter Selinger. Towards a semantics for higher-order quantum computation. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages, TUCS General Publication*, volume 33, pages 127–143, 2004.
[63] Peter Selinger. Towards a quantum programming language. *Math. Struct.*, 14(4):527–586, 2004. URL https://doi.org/10.1017/s0960129504004256.
[64] L Sheridan, D Maslov, and M Mosca. Approximating fractional time quantum evolution. *J. Phys. A Math. Theor.*, 42(18):185302, 2009. DOI: 10.1088/1751-8113/42/18/185302. URL https://dx.doi.org/10.1088/1751-8113/42/18/185302.
[65] R.J. Solomonoff, United States. Air Force. Office of Scientific Research, and Zator Company. *A Preliminary Report on a General Theory of Inductive Inference*. AFOSR TN-60-1459. United States Air Force, Office of Scientific Research, 1960.
[66] Andrew K Tan, Yuan Liu, Minh C Tran, and Isaac L Chuang. Error correction of quantum algorithms: Arbitrarily accurate recovery of noisy quantum signal processing. *arXiv preprint, arXiv:2301.08542*, 2023. URL https://doi.org/10.48550/arXiv.2301.08542.
[67] Ewin Tang and Kevin Tian. A CS guide to the quantum singular value transformation. *arXiv preprint, arXiv:2302.14324*, 2023. DOI: 10.48550/arxiv.2302.14324. URL https://doi.org/10.48550/arXiv.2302.14324.
[68] F. M. Toyama, S. Kasai, W. van Dijk, and Y. Nogami. Matched-multiphase Grover algorithm for a small number of marked states. *Phys. Rev. A*, 79:014301, Jan 2009. DOI: 10.1103/PhysRevA.79.014301. URL https://doi.org/10.1103/PhysRevA.79.014301.
[69] Adriaan van Wijngaarden. Orthogonal design and description of a formal language. *Stichting Mathematisch Centrum. Rekenafdeling*, 1965.
[70] Adriaan Van Wijngaarden, Barry J Mailloux, John EL Peck, Cornelis HA Koster, Charles Hodgson Lindsey, Michel Sintzoff, Lambert GLT Meertens, and Richard G Fisker. *Revised report on the algorithmic language ALGOL 68*. Springer Science & Business Media, 2012.

[71] Philip Wadler. Comprehending monads. In *Proceedings of the 1990 ACM Conference on LISP and Functional Programming*, pages 61–78, 1990. URL https://doi.org/10.1145/91556.91592.

[72] Philip Wadler. The essence of functional programming. In *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–14, 1992. URL https://doi.org/10.1145/143165.143169.

[73] Philip Wadler. Monads for functional programming. In *Advanced Functional Programming: First International Spring School on Advanced Functional Programming Techniques Båstad, Sweden, May 24–30, 1995 Tutorial Text 1*, pages 24–52. Springer, 1995. URL https://doi.org/10.1007/978-3-662-02880-3_8.

[74] Jiasu Wang, Yulong Dong, and Lin Lin. On the energy landscape of symmetric quantum signal processing. *Quantum*, 6:850, 2022. DOI: 10.22331/q-2022-11-03-850. URL https://doi.org/10.22331/q-2022-11-03-850.

[75] Xin Wang, Youle Wang, Zhan Yu, and Lei Zhang. Quantum phase processing: Transform and extract eigen-information of quantum systems. *arXiv preprint, arXiv:2209.14278*, 2022. URL https://doi.org/10.48550/arXiv.2209.14278.

[76] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. Fixed-point quantum search with an optimal number of queries. *Phys. Rev. Lett.*, 113(21):210501, 2014. URL https://doi.org/10.1103/PhysRevLett.113.210501.

[77] Zhan Yu, Hongshun Yao, Mujin Li, and Xin Wang. Power and limitations of single-qubit native quantum neural networks. *Advances in Neural Information Processing Systems*, 35:27810–27823, 2022. URL http://dx.doi.org/10.48550/arXiv.2205.07848.

Supplemental Materials

In what follows we provide a series of appendices. In turn, these detail proofs of theorems in the main body (Appx. A), provide an overview of related prior techniques with performance comparisons (Appx. B), provide detailed analysis of the non-obvious method for computing gadget cost (Appx. C), provide a review of the major results of QSP, M-QSP, and QSVT (Appx. D), provide analysis of *correction protocols* (Appxs. E and F), cover some caveats in the theory of functional approximation by polynomials (Appx. G), compare certain concrete example gadgets for specific problems to their best known counterpart using other quantum algorithms (Appx. H), discuss how gadgets instantiate monadic functions and relate to functional programming (gadget *semantics*, Appxs. I.1 and I.2), and finally provide a formal language for gadgets constituted by an attribute grammar (gadget *syntax*, Appx. I.3).

A Proofs of main results

We begin by providing proofs of the main results of Sec. 2. Note that $\widetilde{\mathcal{O}}$ denotes scaling to leading order. In addition, when using $\mathcal{O}$, we will often suppress scaling beyond leading and second-order (i.e. given leading-order polynomial scaling in parameter p, we keep $\log(p)$ terms and suppress terms of order $\log\log(p)$ and greater).

Lemma 2.1 (Efficient correction of ε-twisted embeddable unitaries)**.** Let U be a ν-twisted embeddable unitary, so U is ν-close to $U' = e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}$. Suppose $\cos(\theta) \in [-1+\delta, 1-\delta]$. Let $\varepsilon > 0$. Then, given controlled access to U, there exists a quantum circuit using $\zeta = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/2}))$ black-box calls to U and a single ancilla qubit which yields a $(\zeta\nu+\varepsilon)$-embeddable unitary V, which is at most $(\zeta\nu+\varepsilon)$-far from $e^{i\theta\sigma_x}$ (the embeddable component of U') with success probability at least $(1-(\zeta\nu+\varepsilon))^2$. Alternatively, given access to two ancilla qubits and uncontrolled oracle access to U, it is possible to implement a circuit which achieves the same $(\zeta\nu+\varepsilon)$-approximation with the same asymptotic query complexity and success probability, on the domain $\cos(\theta_k) \in [\delta, 1-\delta]$. Finally, if U is ν-half-twisted embeddable, and satisfies the same conditions, with the additional constraint that $\cos(\varphi) \in [\gamma, \sqrt{1-\gamma^2}]$ (or $\varphi \in [-\pi/2+2\gamma, -2\gamma] \cup [2\gamma, \pi/2-2\gamma]$), there exists a quantum circuit using $\zeta' = \mathcal{O}(\delta^{-1}\gamma^{-2}\log^2(\gamma^{-4}\varepsilon^{-2}\delta^{-1/2}))$ black-box calls to U and zero ancilla qubits which yields an $(\zeta'\nu+\varepsilon)$-embeddable unitary, at most $(\zeta'\nu+\varepsilon)$-far from the embeddable component $e^{i\theta\sigma_x}$, with unit success. In all cases, the circuits effectuating the corrections can be efficiently and constructively specified. □

Proof. We will prove each of the complexities corresponding to the three different methods for performing the correction of an ε-twisted embeddable unitary. We will begin with the simplest case: the scenario where we are provided access to ancilla qubits as well as controlled oracle access to U.

From Thm. E.1, we know that there exists a single-qubit protocol (effectuated by the superoperator $\mathcal{E}$) making $\zeta_1 = \mathcal{O}(\delta^{-1}\log(\varepsilon_1^{-2}\delta^{-1/2}))$ black-box calls to the unitary U' which yields an ε_1-approximation of $e^{i\varphi\sigma_z}$, assuming $\cos(\theta) \in [-1+\delta, 1-\delta]$. We then have

$$||\mathcal{E}[U] - e^{i\varphi\sigma_z}|| \leq ||\mathcal{E}[U] - \mathcal{E}[U']|| + ||\mathcal{E}[U'] - e^{i\varphi\sigma_z}|| \leq ||\Phi[U] - \Phi[U']|| + \varepsilon_1, \tag{22}$$

where Φ is a QSP protocol. Due to Lem. D.4, we know that $||\Phi[U] - \Phi[U']|| \leq \zeta_1\nu$, implying that the overall error is $\zeta_1\nu + \varepsilon_1$. Due to Cor. F.2.1, there exists a protocol $\mathcal{R}$ making a single black-box call to controlled-$\mathcal{E}[U]$ (which we have assumed we can access, as we have controlled-U access) and using an extra qubit, such that $(\mathcal{R}\circ\mathcal{E})[U]$ will be an $(\zeta_1\nu+\varepsilon_1)$-approximation of $e^{i\varphi/2}e^{i\varphi\sigma_z/2}$, with probability at least $1-(2\zeta_1\nu+2\varepsilon_1)$. It follows, in the case of success, that

$$\begin{aligned}
||(\mathcal{R}\circ\mathcal{E})[U]^\dagger U(\mathcal{R}\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| &\leq ||(\mathcal{R}\circ\mathcal{E})[U]^\dagger e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}(\mathcal{R}\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| + \nu && (23)\\
&\leq 2||e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}(\mathcal{R}\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| + \nu && (24)\\
&\leq 2||(\mathcal{R}\circ\mathcal{E})[U] - e^{i\varphi\sigma_z/2}|| + \nu && (25)\\
&\leq (2\zeta_1+1)\nu + 2\varepsilon_1, && (26)
\end{aligned}$$

where we can drop the overall phases $e^{i\varphi/2}$ within the operator norm, via cancellation (see Rem. F.4). Since the ancilla qubits can be re-used with each non-nested action of $\mathcal{R}$ (see Rem. F.3), the overall protocol will use two ancilla qubits, and makes $\zeta = 2\zeta_1 + 1$ black-box calls to U. Thus, setting $\varepsilon_1 = \varepsilon/2$, so the approximation error is $\zeta\nu+\varepsilon$, the success probability is greater than $(1-(\zeta\nu+\varepsilon))^2$ (as $\mathcal{R}$ is called twice), and $\zeta = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/2}))$ yields the desired result.

In the case where we are provided ancillae, but do not have access *a priori* to the signal oracles, we make use of the protocol of Thm. F.1 to get, from U, an approximation of its controlled-variant. We then make use of the previous protocol $\mathcal{R}$ to achieve an approximation of the desired σ_z-square root. We call this composite protocol $\mathcal{R}'$. Because the first part of the protocol has the exact same asymptotic complexity as the protocol $\mathcal{R}$, and is effectively the implementation of two parallel QSP sequences interspersed by CSWAP gates, the error analysis will proceed in exactly the same way as above.

Finally, from Thm. F.3, not that there exists a single-qubit protocol $\mathcal{R}''$ making $\zeta_2 = \mathcal{O}(\gamma^{-2}\log(\varepsilon_2^{-2}\gamma^{-1/2}))$ black-box calls to $e^{i\varphi\sigma_z}$ such that $\mathcal{R}''[e^{i\varphi\sigma_z}]$ is an ε_2-approximation to $e^{i\varphi\sigma_z/2}$, assuming $\cos(\varphi) \in [\gamma, \sqrt{1-\gamma^2}]$. Of course, since $\mathcal{R}''$ is effectively a QSP protocol, we can once again make use of Lem. D.4 to bound the accumulation of errors once again. We have

$$
\begin{aligned}
||(\mathcal{R}''\circ\mathcal{E})[U]^\dagger U(\mathcal{R}''\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| &\leq ||(\mathcal{R}''\circ\mathcal{E})[U]^\dagger e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}(\mathcal{R}''\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| + \nu && (27)\\
&\leq 2||e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2}(\mathcal{R}''\circ\mathcal{E})[U] - e^{i\theta\sigma_x}|| + \nu && (28)\\
&\leq 2||(\mathcal{R}''\circ\mathcal{E})[U] - e^{i\varphi\sigma_z/2}|| + \nu && (29)\\
&\leq 2||\mathcal{R}''[e^{i\varphi\sigma_z}] - e^{i\varphi\sigma_z/2}|| + 2||\mathcal{R}''[e^{\varphi i\sigma_z}] - \mathcal{R}''[\mathcal{E}[U]]|| + \nu && (30)\\
&\leq 2\varepsilon_2 + 2\zeta_2\varepsilon_1 + (2\zeta_2\zeta_1 + 1)\nu && (31)
\end{aligned}
$$

Clearly, a total of $\zeta' = 2\zeta_2\zeta_1 + 1$ black-box calls are made to U throughout the entire protocol. We set $\varepsilon_2 = \varepsilon/4$ and $\varepsilon_1 = \varepsilon/4\zeta_2 = \widetilde{\mathcal{O}}(\varepsilon\gamma^2)$, where we drop the log factor of $\log(\varepsilon^{-2}\gamma^{-1/2})^{-1}$ (as this will only contribute a log log factor when plugged into the protocol depth upper-bound). Therefore, $\zeta_1 = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\gamma^{-4}\delta^{-1/2}))$, and

$$
\zeta' = 2\zeta_2\zeta_1 + 1 = \mathcal{O}\left(\frac{1}{\gamma^2\delta}\log^2\left(\frac{1}{\gamma^4\varepsilon^2\delta^{1/2}}\right)\right). \tag{32}
$$

The approximation error is then $\zeta'\nu + \varepsilon$, and the proof is complete. □

With this result, the main theorem characterizing gadget correction follows fairly immediately.

Theorem 2.1 (Efficient correction of unitaries produced by (a,b) gadgets)**.** Let $\varepsilon, \delta > 0$ and $\mathfrak{G}$ an (a,b) gadget whose output unitaries are guaranteed to be ν-twisted-embeddable: they are ν-close to unitaries $e^{i\varphi\sigma_z/2}e^{i\theta_k\sigma_x}e^{-i\varphi\sigma_z/2}$ whose embeddable components encode $\cos\theta_k, k \in [b]$ within $[-1+\delta, 1-\delta]$. Assuming an access model in which one can query the *controlled* single-qubit unitaries achieved by the output legs of $\mathfrak{G}$, then given $k \in [b]$, there exists a computable $\nu' = \widetilde{\mathcal{O}}(\delta\varepsilon)$ and a quantum circuit using $\zeta = \widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$ black-box calls to the controlled unitaries produced by executing $\mathfrak{G}$ on the input oracles, as well as one ancilla qubit, such that if $\nu \leq \nu'$ then the circuit achieves an ε-approximation of $U_k' = e^{i\theta_k\sigma_x}$ (the embeddable component of the k-th output unitary of $\mathfrak{G}$), with success probability at least $(1-\varepsilon)^2$, for all k. Alternatively, given access to two ancilla qubits and access only to uncontrolled U queries, the same outcome can be achieved with the same asymptotic complexity and success probability over the domain $\cos(\theta_k) \in [\delta, 1-\delta]$. Finally, in the case that the output legs are all ν-half twisted embeddable with $\cos(\varphi) \in [\gamma, \sqrt{1-\gamma^2}]$, the same can be done with no extra space and unit success probability using $\zeta = \widetilde{\mathcal{O}}(\delta^{-1}\gamma^{-2}\log^2(\varepsilon^{-1}))$ black-box calls to $\mathfrak{G}$ and the guarantee that $\nu \leq \nu''$, for a computable $\nu'' = \widetilde{\mathcal{O}}(\delta\gamma^2\varepsilon)$. Each protocol described above can be constructed efficiently and obliviously to the output unitaries of the gadget. □

Proof. This follows immediately from invoking Lem. 2.1 for each of the output gadgets, enforcing the condition that $\zeta\nu = \varepsilon \Rightarrow \nu = \varepsilon/\zeta = \widetilde{\mathcal{O}}(\varepsilon\delta)$ in the first and second cases, and $\zeta\nu = \varepsilon \Rightarrow \nu = \widetilde{\mathcal{O}}(\delta\gamma^2\varepsilon)$ in the final case, so that the total approximation error in both cases is $2\varepsilon = \mathcal{O}(\varepsilon)$. □

Remark A.1 (On the acceptable efficiency of correction protocols)**.** Lemma 2.1 shows that, given an approximately embeddable unitary, the error accumulated during its correction is (1) proportional to the error of the input unitary and (2) the constant of proportionality is linear in the length of the correction procedure. We clarify in what follows that this scaling is not as poor as it appears, and connect this to natural but implicit assumptions for protocols given in this work:

(a) Firstly, the error in the input approximately embeddable unitary (ν in Lemma 2.1) arises *only from previous correction protocols*, and thus can be suppressed by a factor of ζ, for instance, by increasing the length of the previous correction procedure by a factor only *logarithmic* in ζ (noting the ε-dependence in Lemma 2.1, and that the argument is more subtle when corrections are changed). In

other words, ν can be made small quite easily, inheriting the rapid convergence of QSP polynomials to desired smooth functions.

(b) Secondly, analogously to previous work on the self-composition of QSP and general classical signal processing [32, 34, 39, 46, 61], this work considers circuits comprising logarithmically-many protocol-nesting operations, and thus only logarithmically-many uses of correction protocols in the length of a given 'base protocol.' While such scaling is more difficult to define where protocols of varying length are used, rather than where one protocol is recursively self-composed, this statement will be true even if scaling is taken with respect to the length of the longest 'base protocol.' Ultimately, this restriction is equivalent to requiring that our composite protocols have length *polynomial in the length of their smallest sub-protocols.*

(c) Taken together the (a) rapid convergence of polynomial approximation in QSP and the (b) natural restriction to logarithmic-depth composite *gadgets* (Def. 2.3; for now to be thought of as parallel circuits with the basic form of QSP/M-QSP), means the ultimate length of protocols scales, at worst, inverse polynomially in the desired output error.

While the precise cost of a specific composite gadget can be involved to compute (Appx. C), and can change depending on properties of the achieved functions it comprises (Rems. H.1 and H.2), we can assert that QSP-based computations are *acceptable* (in a colloquial sense) if their required length scales only inverse-polynomially in the desired error. While this is worse than the best case scenario of using QSP or M-QSP to approximate smooth functions, it is no worse than the scaling appearing when approximating common discontinuous functions. Note also this error is distinct from the error of functional approximation within a standard QSP or M-QSP protocol, but is of the same order, and can be independently manipulated in minimizing gadget depth. □

We are now able to discuss the cost associated with composing gadgets. First, a minor result, showing the simple relationship between unitary approximations and function approximations achieved with gadgets.

Lemma A.1. Let $\mathfrak{G}$ be an (a, b) gadget which achieves unitaries U'_k which ε-approximate unitaries V_k for $k \in [b]$ over all $(x_0, \ldots, x_{a-1}) \in \mathcal{D}$. Then $\mathfrak{G}$ achieves functions f_k which ε-approximate functions $g_k = \langle 0|V_k|0\rangle$ for $k \in [b]$ over all $(x_0, \ldots x_{a-1}) \in \mathcal{D}$. □

Proof. Note that $|f_k - g_k| = |\langle 0|U'_k|0\rangle - \langle 0|V_k|0\rangle| = |\langle 0|U'_k - V_k|0\rangle| \leq ||U'_k - V_k|| \leq \varepsilon$. □

Theorem 3.1 (Composing a gadget with an atomic gadget)**.** Let $\varepsilon, \delta > 0$, $\mathfrak{G}$ be an (a, b) gadget and (Ξ, S) an antisymmetric atomic (c, d) gadget, where $\Xi \equiv \{\Phi_0, \ldots, \Phi_{d-1}\}$ and $S \equiv \{s_0, \ldots, s_{d-1}\}$ for (Ξ, S). Suppose the gadget $\mathfrak{G}$ achieves

$$F(x) \equiv \{f_0(x_0, \cdots, x_{a-1}), f_1(x_0, \cdots, x_{a-1}), \ldots, f_{b-1}(x_0, \cdots, x_{a-1})\} \in (\mathbb{R}^a \to \mathbb{R}^b) \tag{5}$$

over $x \in [-1, 1]^a$, and the atomic gadget (Ξ, S) achieves

$$G(y) \equiv \{g_0(y_0, \cdots, y_{c-1}), g_1(y_0, \cdots, y_{c-1}), \ldots, g_{d-1}(y_0, \cdots, y_{c-1})\} \in (\mathbb{R}^c \to \mathbb{R}^d) \tag{6}$$

over $y \in [-1, 1]^c$. Let $\mathfrak{I} = (B, C, W)$ be an interlink between these gadgets. Then, there exists a gadget $\mathfrak{G}'$ which ε-approximately achieves

$$H(x, y') \equiv \bigcup_{k\in[d]} g_k \left(\bigcup_{j\in B} f_{W(j)}(x_0, \ldots, x_a) \cup \bigcup_{k\notin C} y_k \right) \cup \bigcup_{k\notin B} f_k(x_0, \ldots, x_a) \tag{7}$$

over $(F(x), y') \in \mathcal{D}$, where y' is the subset of y_k such that $k \notin C$ and $\mathcal{D}$ is a domain determined by the correction procedure utilized. The set union symbol is abused to mean concatenation of lists with respect to the pre-established order of the labels of the relevant input and output *lists* (not sets), and where $W(j)$ is the result of the application of the specified permutation applied to the *index* of the subset member j in the ordered sublist C of $[c]$. Moreover, $\mathfrak{G}'$ can be constructed efficiently, and uses only a description of (Ξ, S) and a total of $\widetilde{\mathcal{O}}(d|\Xi|_\infty \zeta)$ black-box calls to the *unitaries produced by running* $\mathfrak{G}$ to realize the function H.

Note ζ, the cost of correcting the gadget $\mathfrak{G}$, is precisely the cost to make $\mathfrak{G}$ *snappable*, and this snappability enables the simple compositional form of Eq. 7. ζ is one among two choices presented

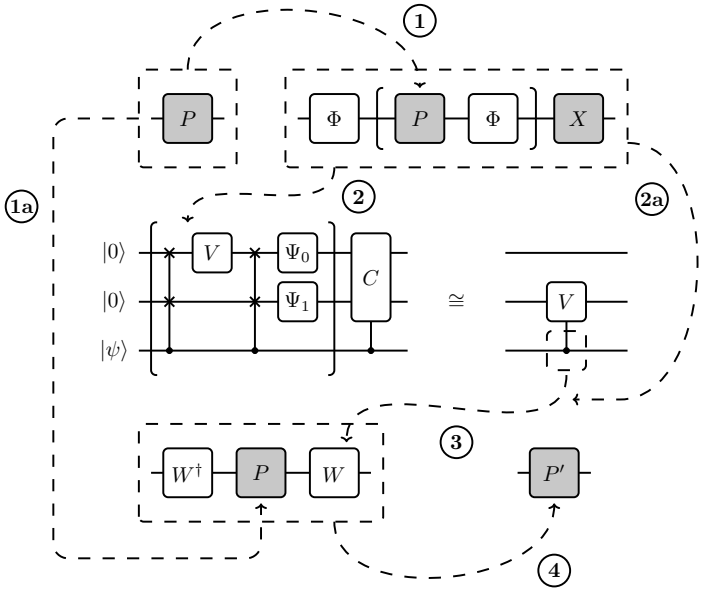


Figure 5: High-level breakdown of the correction protocol of Thm. 2.1. We depict multiple pipelines for preparing, given a gadget's output leg which embeds the polynomial transform P, a leg which embeds P in the same way but whose rotation axis is approximately fixed and known (denoted P'). Dotted boxes indicate substitution of the enclosed circuit as a module into another circuit, and $\cong$ indicates circuits with approximately identical action (in the operator norm). Here (1) gives substitution into a QSP protocol, which generates V (an approximate z-rotation). Using a pair of Fredkin gates and two additional qubits in fixed, known states, through (2) we can approximately apply the square root of this z-rotation to a qubit of our choice by creating approximate controlled-V access using the pair of QSP protocols parameterized by Φ_0, Φ_1 as in Thm. F.1; here C is a fixed, known correction unitary. If a different access assumption is made (see Tab. 3), (2) can be skipped, as in (2a) one assumes either direct access to controlled-V per Cor. 72 of [26], or a domain restriction on the possible V allows ancilla-free square roots to be taken. In (3) one conjugates the original protocol P (1a), which is precisely, through (4), what was desired of P'. Note both W and $W^\dagger$ are preparable as gadget outputs are obliviously invertible for all arguments: $ZPZ \equiv P^\dagger$. Here boxes Φ, Ψ_0, Ψ_1 are in general different, controllable z-rotations, and brackets indicate finite repetition of a block with possibly different phases.

in Thm 2.1, with the variables ε and δ carrying over. Generally, $\zeta = \widetilde{O}(\text{polylog}(\varepsilon^{-1})\text{poly}(\delta^{-1}))$, with possible polynomial scaling in a parameter γ, given in Thm. 2.1, as well. This choice also dictates whether $\mathcal{O}(1)$ or zero ancilla qubits are required to perform this composition, and whether the success probability of achieving each individual function is at least $(1-\varepsilon)^2$, or 1. $|\Xi|_\infty$ is the maximum length of lists within Ξ. For a proof of this result, see Appx. A. □

Proof. We must correct each of the gadget $\mathfrak{G}$'s e outputs involved in the interlink, via the protocol of Thm. 2.1, before composition. The overall complexity will simply amount to the cost of the corrective protocol, multiplied by the total query complexity required to implement all of the output legs of the atomic (c,d) gadget involved in the interlink, given its black-box use of input gadgets (a quantity upper-bounded by $d|\Xi|_\infty$). Of course, this query complexity will often be much lower, since the sequence of Ξ will often only utilize a small number of queries to input legs involved in the interlink, which are the only legs which require correction.

Moreover, given that corrective protocols used in Thm. 2.1 only have logarithmic dependence on ε, and accumulate errors linearly in circuit depth, making the transformation of $\varepsilon \mapsto \varepsilon/|\Xi|_\infty$ introduce only logarithmic factors in $|\Xi|_\infty$. This implies that we require the given $\widetilde{\mathcal{O}}(d|\Xi|_\infty\zeta)$ query-complexity (for ζ of Thm. 2.1) to ε-achieve all of the unitaries of the M-QSP protocols/atomic gadget achieving functions $g_0,\ldots,g_{d-1}$, composed with the signal operators $e^{i\arccos(f_1(x))\sigma_x},\ldots,e^{i\arccos(f_{b-1}(x))\sigma_x}$, for x over a prescribed domain (depending on the correction protocol), with respect to the interlink. Lem. A.1 implies that the composite gadget will ε-approximate the desired function composition, H. □

Remark A.2 (Counting gadget interlinks)**.** Note that given $\mathfrak{G}$ and (Ξ,S) as (a,b) and (c,d) gadgets respectively, then there are

$$L_{(a,b),(c,d)} \equiv \sum_{e=0}^{\min(b,c)} \binom{b}{e}\binom{c}{e}(e!) \tag{33}$$

valid interlinks between these gadgets, and $\binom{b}{e}\binom{c}{e}(e!)$ interlinks which induce the same type transformation, but which may not in general be equivalent. □

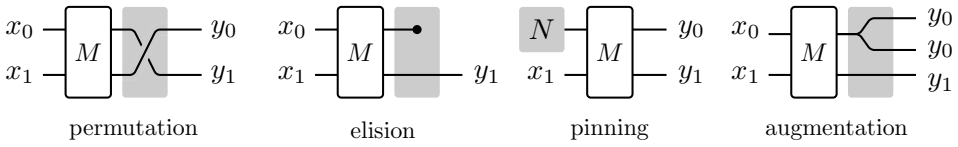
Definition A.1 (Augmenting, eliding, pinning, and permuting gadgets)**.** There exist a number of simple operations over gadgets that take only one argument, and we discuss them here, terming them *augmentation*, *elision*, *pinning*, and *permutation*. Graphically, these correspond to adding or removing out-going legs, fixing in-going legs, or permuting out-going legs in the picture discussed in Fig. 9. These have the following types

$$(a,b) \to (a,b+c), \quad \text{augmentation}, \tag{34}$$
$$(a,b) \to (a,b-c), \quad \text{elision}, \tag{35}$$
$$(a,b) \to (a-c,b), \quad \text{pinning}, \tag{36}$$
$$(a,b) \to (a,b), \quad \text{permutation}. \tag{37}$$

Given an (a,b) gadget that achieves $\{f_0(x_0,\cdots,x_{a-1}), f_1(x_0,\cdots,x_{a-1}),\ldots,f_{b-1}(x_0,\cdots,x_{a-1})\}$, the result of the above actions are gadgets which achieve the following:

$$\bigcup_{k\in B'} f_k(x_0,\cdots,x_{a-1}), \tag{38}$$
$$\bigcup_{k\in B''} f_k(x_0,\cdots,x_{a-1}), \tag{39}$$
$$\bigcup_{k\in B} f'_k(x_{j_0},\cdots,x_{j_{a-c-1}}), \tag{40}$$
$$\bigcup_{k\in B} f_{W(k)}(x_0,\cdots,x_{a-1}), \tag{41}$$

where $W(k)$ is the action of a permutation over $[b]$ on the index of an out-going leg of the gadget, f'_k is the function f_k with some size-c subset of its input arguments fixed, B' is a list containing $[b]$ in order, possibly interspersed with repeated elements from $[b]$, and B'' is an ordered sublist of B. We can again simply depict these operators graphically:



where M is chosen here to be a $(2,2)$ gadget, whose legs can be permuted, elided, pinned, or augmented, resulting in $(2,2)$, $(2,1)$, $(1,2)$, and $(2,3)$ gadgets respectively. Here N is some superoperator which returns a fixed unitary (often chosen by the computing party). □

The cost of an operator over a gadget is most clearly expressed in the number of queries to the subsidiary gadget(s) (as their outputs are used in a black-box way), as well as any additional space required. Given an interlink (Def. 3.1) defined by $\mathfrak{I} \equiv (B, C, W)$, it is evident that the permutation requires at most $\mathcal{O}(|B|)$ two-qubit gates. Moreover, for each of the $|B|$ out-going legs, the assumptions of Thm. 2.1 do not distinguish between their corrections, meaning that the same superoperator is applied to each, with the same asymptotic query complexity. This induces a cost dependent only on the out-going leg number and the maximum length among sub-components of the outer gadget. For augmentations (Def. A.1) of size c, a total of c calls to the first gadget are necessary and sufficient (with additional space linear in c), while elisions and pinnings of size c require only a single call, and no additional space.

To conclude the proofs of the main results, we revisit the equivalence theorem, Thm. 3.2.

Theorem 3.2 (Efficient equivalence of polynomial compositions and gadget assemblages)**.** Let the expression $\mathcal{L}(x_1, \ldots, x_n)$ denote the set of all polynomials achievable by atomic gadgets in the variables $x_1, \ldots, x_n$. Suppose $P(x_1, \ldots, x_n)$ is a polynomial of degree D and can be split into a tower of $m = \mathcal{O}(\log(D))$ *interlinked polynomials* (Rem. 3.2),

$$P = P^{(m-1)} \circ_{\mathfrak{I}_{m-2}} \left(P^{(m-2)} \circ_{\mathfrak{I}_{m-3}} \circ \left(\cdots \left(P^{(1)} \circ_{\mathfrak{I}_0} P^{(0)} \right) \cdots \right) \right) \tag{9}$$

such that $P_i^{(j)} \in \mathcal{L}(x_1, \ldots, x_n)$ for all i and j. Assume that the composition satisfies the domain condition (Rem. C.1) with P_k supported on $\mathcal{D}_k$ such that each $\mathcal{D}_k$ is separated from singular points ± 1 (and possibly 0) by some length $\delta \in \mathcal{O}(1)$ for all k. Then, there exists an assemblage of m atomic, snappable gadgets which ε-approximates the polynomial P on the domain yielded from the $\mathcal{D}_k$, with cost $\widetilde{O}(\mathrm{poly}(D)\,\mathrm{polylog}(\varepsilon))$. □

Proof. To construct the desired polynomial, we will perform an m-fold composition of atomic gadgets, each of which have had the correction procedure applied to their output legs. We are able to bound the accumulated error within the gadget via Thm. C.2, and we are able to guarantee that the desired polynomial is achieved to ε-accuracy via Thm. 3.1. Since each gadget being considered is atomic and achieves polynomial is the variables $x_1, \ldots, x_n$ (it could, of course, be constant in some of these variables), each gadget can have at most n input legs. Thus, we think of each interlink $\mathfrak{I}_k$ as being between two sets of *precisely* n legs, where the action of a gadget on some of these legs may merely be the identity. This viewpoint, while leading to sub-optimal scaling in particular cases when interlinks are sparse, will simplify the proof of this theorem.

Let $\mathfrak{G}^{(k)}$ be the gadget ν-achieving polynomial $P^{(k)}$, with the correction procedure appended to the output legs. For the sake of simplicity, we are using the correction procedure which requires ancilla qubits, so there is no concern about half-twisted embedability. It follows that to implement a single output leg of this gadget on $\mathcal{D}_k$ requires circuit depth $\widetilde{O}(d_k \delta^{-1} \mathrm{polylog}(\nu^{-1}))$, where $d_k = \max_j \deg(P_j^{(k)})$ is the depth of the uncorrected gadget achieving $P^{(k)}$ (Thm. 2.1). Because each of the gadgets $\mathfrak{G}^{(k)}$ has ν-embeddable output, they can be interlinked in a way such that the polynomials will (approximately) compose as expected. In particular, we know that $\mathfrak{G}^{(k)}$ will ν-approximate the unitary function $\mathfrak{F}^{(k)}$ with

$$\mathfrak{F}^{(k)} \left[e^{i \arccos(x_0)\sigma_x}, \ldots, e^{i \arccos(x_n)\sigma_x} \right] = \left(e^{i \arccos(P_1^{(k)}(x_1,\ldots,x_n))\sigma_x}, \ldots, e^{i \arccos(P_n^{(k)}(x_1,\ldots,x_n))\sigma_x} \right) \tag{42}$$

for $x \in \mathcal{D}_k$. Thus, by Thm. C.2, it follows that the error between $\mathfrak{F}$: the composition of all the $\mathfrak{F}^{(k)}$ and the function achieved by $\mathfrak{G}$: the gadget which is a full interlink of the sequence of $\mathfrak{G}^{(k)}$ is given by

$$\begin{aligned} &\left\| \mathfrak{G} \left[e^{i \arccos(x_0)\sigma_x}, \ldots, e^{i \arccos(x_n)\sigma_x} \right] - \mathfrak{F} \left[e^{i \arccos(x_0)\sigma_x}, \ldots, e^{i \arccos(x_n)\sigma_x} \right] \right\| \\ &\leq \sum_{k=-1}^{m-1} v_\nu^T \prod_{k<\ell\leq m-1} C^{(\ell)} \end{aligned} \tag{43}$$

where $C^{(\ell)}$ is the cost matrix of the ℓ-th gadget in the composition. Note that each of the columns of $C^{(\ell)}$ sum to some $d'_\ell = \widetilde{O}(d_\ell \text{polylog}(\nu^{-1}))$, and each entry of v_ν is equal to ν. Therefore,

$$\sum_{k=-1}^{m-1} v_\nu^T \prod_{k<\ell\leq m-1} C^{(\ell)} = \widetilde{\mathcal{O}}\left(\nu d_1 \cdots d_m \left(K\delta^{-1}\text{polylog}(\nu^{-1})\right)^m\right) \tag{44}$$

for some constant K. Note that $D = d_1 \cdots d_m$. In addition, since $m = \mathcal{O}(\log(D))$,

$$\widetilde{O}\left((K'\delta^{-1}\text{polylog}(\nu^{-1}))^m\right) = \mathcal{O}\left(\exp\left(K'' \log(D) \log\log(\nu^{-1})\right)\right), \tag{45}$$

and thus an error of ν on an input generates an error of the asymptotic form $\widetilde{\mathcal{O}}(\nu D^{K'' \log\log \nu^{-1}})$ on an output. If we want to achieve error at maximum some desired ε, one then finds that the required initial ν must satisfy the implicit equation

$$-\log \nu^{-1} + K'' \log\log \nu^{-1} \log D = -\log \varepsilon^{-1}, \tag{46}$$

which can be rearranged into the suggestive form

$$-e^{\log\log \nu^{-1}} + K'' \log D \log\log \nu^{-1} = -\log \varepsilon^{-1} \tag{47}$$

This permits us to express $\log\log \nu^{-1}$ in terms of the Lambert W function (taking the W_{-1} branch, which is valid for our intended range of small ε), namely

$$\log\log \nu^{-1} = -\frac{\log \varepsilon^{-1}}{K'' \log D} - W_{-1}\left(-\frac{e^{-\log \varepsilon^{-1}/K'' \log D}}{K'' \log D}\right). \tag{48}$$

For small ε the the Lambert W function has simple behavior, approaching $\log(-x)$ for small argument x, meaning that the whole expression simplifies to $\log\log \nu^{-1} = \log K'' D$ at leading order. As such we see that for very small ε, $\log \nu^{-1}$ grows only as $K'' D$; as this is the only multiplicative factor from the precision ν entering into the complexity of each sub-gadget, we see that their length does not grow superpolynomially in the logarithm of ε^{-1}. This yields the desired $\widetilde{\mathcal{O}}(\text{poly}(D)\,\text{polylog}(\varepsilon^{-1}))$ scaling. □

B Related prior techniques and performance comparisons

The resource requirements for enacting a multivariable polynomial transformation on the mutual singular values of a series of commuting block encodings requires some scaffolding to analyze, which we present here. We will call d the (generalized) functional degree of the achieved multivariable polynomial, r the number of variables within this transform, and s is the number of additional qubits used for the original block encoding(s). In compressing these values to single scalars, we usually mean the maximum over an implicit tuple, e.g., of the degree of the desired polynomial in each variable, or the required auxiliary space of each block-encoded input. Note that the generalized degree of the implemented function d generally scales as the polylogarithm of the inverse uniform approximation error ε to some idealized continuous function. We will suppress this factor in the discussion below, though one is free to substitute $\text{poly}(d)\text{polylog}(\varepsilon^{-1})$ for $\text{poly}(d)$ where it appears, if one is trying to achieve only a polynomial approximation to a desired, ideal functional transform. We also note that δ is sometimes introduced to bound uniform approximation away from regions in which the function has an $\mathcal{O}(1)$ jump discontinuity, or a hard boundary condition; for this, $\text{poly}(d)\text{polylog}(\varepsilon^{-1})\text{poly}(\delta^{-1})$ can be substituted in the same way.

We give the defining complexities of multivariable quantum eigenvalue transformation (M-QET), which relies on techniques from linear combination of unitaries (LCU) based methods, from [7]. Their work does not consider successive composition of protocols; moreover works which do consider such compositions, e.g., [48], do so only only in the single-variable setting, and only in the case of recursive function application. In this way our work is not directly comparable to either of these methods, as our correction procedure (Thm. 2.1) confers an additional ability to not just compose functions or block encode multivariable polynomial transforms, but to link and partially-compose gadgets achieving such transforms.

Consequently, when we compare our methods to previous works below, we are doing so on the level of individual multivariable quantum eigenvalue transforms (a restriction of QSVT to normal operators). On this front both our work and [7] demonstrate near arbitrary block encodings of multivariable polynomials in commuting operators, allowing for general comparison.

Namely, we consider the resource costs of the basic units of our work: gadgets (Def. 2.3), each of whose out-going legs yield an embeddable unitary which, if the gadget was built from the composition of atomic gadgets, can be seen to achieve a given (approximation to) a multivariable polynomial transform of the input leg variables. Viewed in this way our work approaches the problem of multivariable polynomial block encodings from a starkly different direction than [7]. Rather than considering the efficiency of an algorithmic method that achieves arbitrary functional block encodings, we investigate the closure of a highly-resource-efficient *class of subroutines* under *block encoding-preserving* superoperators, and show that we can achieve unexpectedly expressive multivariable block encodings with improved resource scaling, i.e., without the space use and post-selection requirements of LCU. In this way, our methods achieve a synthesis of the aims of [7] and [48], albeit using tools from M-QSP [59] language from work on semantic embedding [60], and abstractions from the classical theory of functional programming languages. The ultimate result is significant savings in space complexity and success probability at the cost of immediate generality of embeddable functions. It is also important to observe that, just as in initial comparisons between LCU and QSVT methods for Hamiltonian simulation [6, 26, 43], the *query-complexity's* dependence on polynomial degree and approximation error are both within polynomial of optimal already. Our benefit is however not just tied to the query complexity, but in the modular re-use of pre-computed gadgets, substantial auxiliary space savings, and ability to leverage the powerful functional-analytic results underlying the quite mature theory of QSP-based algorithms.

Comparisons of our methods along major resource axes in terms of d, r, s are made in Table 2. We also emphasize that our methods are not algorithmically incompatible with those of [7], just resource incomparable. I.e., our gadgets are agnostic to the circuit having constructed a given block encoding used as input, and thus similar constructions to those of [7] can furnish immediately applicable upper bounds for certain multivariable eigenvalue transforms (following amplifying into the proper block), and in turn be fed into our gadgets if desired. In this way, we hope multiple methods for efficient block encoding can mutually inform one another, and settle sophisticated questions of the minimum-required space and query complexity to achieve certain (approximations to) desired functional transforms.

Method	Query complexity	Space complexity	Subnormalization
[7]	$\mathcal{O}(\text{poly}(r)d^{\text{poly}(r)})$	$\mathcal{O}(\text{poly}(r)[\text{poly}(d)+\text{poly}(s)])$	$\mathcal{O}(\text{poly}(d)^{\text{poly}(r)})$
This work	$\mathcal{O}(d)$	$\mathcal{O}(s+c)$	$\mathcal{O}(1)$

Table 2: Asymptotic query and space complexity, as well as block encoding sub-normalization factor for this work's multivariable block encodings and those of other methods based on LCU methods. Note that the ε, δ dependencies (i.e., $\mathcal{O}(\varepsilon)$-uniformly approximating functions on regions at most $\mathcal{O}(\delta)$-close to discontinuities or singularities) are here suppressed, but are in general identical between methods [26], and that the polynomial scaling described for LCU methods are often low-degree or linear in practice. Here subnormalization relates roughly linearly to the number of runs required to measure the proper sub-block of the block encoding. Note also that this comparison does not consider the new ability, proposed in this paper, to curry gadgets comprising M-QSP protocols in a semantically clear way, and instead considers the complexity of an initial atomic gadget's construction. Note that the variable c is a constant that can be zero for half-twisted-embeddable protocols (Def. 2.2).

Beyond query and space complexity, we take some time to discuss additional benefits to our method in relation to observations made by previous works. One of these relates to the fact that, if a gadget uses a discrete set of phases, and a composite computation is built from repeated interlinks of that gadget with itself (or some other finite set of gadgets), then the phases used in the resulting circuit necessarily come from the union of phases used by the constituting gadgets. As discussed by [48], there exist settings [66] in which the number of distinct QSP phases used in a protocol defines the complexity of certain error-correction procedures on that protocol. It is an intriguing open question whether the limited distinct phase character possible with our methods (already an experimental advantage) can be exploited in the same way as those of the single variable setting [66]. Moreover, even if a variety of gadgets are used, those most often repeated, or prone to error, can be selectively optimized, given their ubiquity, and applied wherever given our method's preservation of gadget contiguity as subroutines.

As a concluding note, we repeat that the comparison made here cannot be entirely fair, due to the different goals of the previous works considered and our own. We assume, within Table 2, that a given multivariable polynomial transformation *can* be reached by both methods, and then show that the query complexities, space complexities, and subnormalization factors differ. Nevertheless, the transforms possible with our method are sufficiently expressive to warrant comparison within this implied overlap, and moreover, a wealth of results exist in classical signal processing showing the success of considering composite protocols comprising multiple uses of inter-related and simpler sub-filters [34, 61].

Remark B.1 (On the general inequivalence of composite gadgets)**.** As discussed previously, as well as in the caption of Tab. 2, there exist non-obvious conditions on polynomial transforms in M-QSP which effect the space and query complexity of the correction procedures used to compose gadgets. This is in tandem with the (approximate) polynomial decomposition theorems [16, 17, 25, 37] specifying how to break a composite polynomial into smaller sub-protocols combined through composition. Both of these subtleties to understanding the resource requirements of achieving a given transform are further confounded by the non-trivial inequivalence of gadgets which apparently achieve the same transform. For instance, consider the simple $(1,1)$ product gadget (Ex. 4.3, with a pre-applied square root) which outputs x_0^2, as well as the $(1,1)$ gadget which squares an oracle, achieving $2x_0^2-1$, and averages it with the $(1,1)$ trivial gadget achieving 1 (using the $(2,1)$ gadget of Ex. 4.6) to achieve the same x_0^2. The averaging gadget, by merit of using the product gadget as a subroutine, has strictly higher query complexity, complicating the question of determining the minimal composition of gadgets achieving a given function. Evidently, the resources required for a given functional transform (or its approximate version) are sensitive to small changes in objective or constitutive subroutines. □

The relative incomparability of our work and those of others extends further, considering that our work explicitly characterizes the cost of gadget composition (Sec. C), in direct generalization of techniques discussed in [48, 59], which considered the simpler single-variable setting. Modular filter design, from which our work draws some inspiration, is classically well-developed, as mentioned, and undergirded by a correspondingly rich theory of polynomial decomposition theorems, whose approximate forms and associated algorithms [17, 25, 37], can be used to approach the generality offered [7]. By focusing on smaller, modular, and well-characterized subsets of possible transforms, algorithmic clarity is improved, automatic optimization and verification becomes possible, and experimental realization is simpler.

C On computing the cost of gadgets

General gadgets are composite objects with multiple inputs and outputs, and consequently defining their implementation cost requires some bookkeeping. Here, we define basic objects associated with this cost, such that the cost of compositions of (possibly composite) gadgets is simple to compute.

Definition C.1 (Gadget cost matrix)**.** Let $\mathfrak{G}$ be an (m,n) gadget. Then there is a $m \times n$ *cost matrix* C associated with $\mathfrak{G}$

$$C = \begin{array}{c} \\ x_0 \\ x_1 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{cccc} x'_0 & x'_1 & \cdots & x'_m \end{array} \\ \begin{pmatrix} c_{00} & c_{01} & \cdots & c_{0m} \\ c_{10} & c_{11} & \cdots & c_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n0} & c_{n1} & \cdots & c_{nm} \end{pmatrix} \end{array}, \quad \text{characterizing} \quad \begin{array}{c} \\ x_0 \\ x_1 \\ \\ x_n \end{array} \begin{array}{c} \\ \\ \mathfrak{G} \\ \\ \\ \end{array} \begin{array}{c} \\ x'_0 \\ x'_1 \\ \\ x'_m \end{array} \tag{49}$$

where element c_{jk} indicates the required number of queries to the input leg encoding x_j to achieve an output leg encoding x'_k up to some precision $\varepsilon_k \geq 0$ for $x_j, j \in [n]$ in a particular domain $\mathcal{D}_j$. □

It is not so difficult to determine these cost matrices in common cases. For instance, let (Ξ, S) define an atomic gadget $\mathfrak{G}$; then the cost matrix C of $\mathfrak{G}$ has elements $c_{jk} = S_{jk}$, where S_{jk} is the number of occurrences of j in the k-th element of S. Furthermore, if one considers the correction procedure defined previously, then the cost matrix of a corrected gadget has each c_{jk} for the non-corrected gadget multiplied by a factor ζ determined by Thm. 2.1, for a desired precision ε to an embeddable output, over a desired range. For our purposes, we care primarily about the asymptotic behavior of elements of the cost matrix, and will thus treat them up to logarithmic factors. Cost matrices provide a clean formalism in which we can compute the cost of implementing composite gadgets.

Lemma C.1 (Cost matrix for a composite gadget)**.** We compute the cost matrix of a composite gadget. Let $\mathfrak{G}$ be an (a,b) gadget and $\mathfrak{G}'$ a (c,d) gadget. Given an interlink $\mathfrak{I} \equiv (\tilde{B}, \tilde{C}, W)$ (Def. 3.1, where tildes have been added to prevent variable overloading), we know that the resulting gadget has type $(a+c-e, b+d-e)$ where $e \equiv |\tilde{B}| = |\tilde{C}|$ is the number of contracted legs. We can define block versions of the cost matrices C, C'

$$C = \left[\begin{array}{c|c} A & A' \end{array}\right], \quad C' = \left[\begin{array}{c} B \\ \hline B' \end{array}\right], \tag{50}$$

where A is $a \times e$, A' is $a \times (b-e)$, B is $e \times d$, and B' is $(c-e) \times d$; this will result in C'' as defined below as $(a+c-e) \times (d+b-e)$, as expected. Note that input and output variables have been permuted such that A, B encompass the marked subsets $\tilde{B}, \tilde{C}$ of the interlink. The block version of the cost matrix C'' for the composition according to the interlink $\mathfrak{I}$ can then be simply defined in terms of C, C':

$$C'' = \left[\begin{array}{c|c} W(A)B & A' \\ \hline B' & 0 \end{array}\right]. \tag{51}$$

Here W, from the interlink, possibly permutes the columns of A, and $W(A)B$ is the standard matrix product between $W(A)$ and B. The bottom right block is all zeros. Note that in the case that $\mathfrak{G}'$ is an atomic gadget, the cost matrix C' can be explicitly computed as discussed previous to this lemma. □

It is apparent that the general cost matrix of a series of interlinked gadgets can be quite complex, with elements that can grow quickly in the number of compositions. It follows that in the generic case, using cost matrices to determine the required input-leg precision to achieve a desired output-leg precision is a task very particular to individual gadget networks. However, certain generalizations can be made, provided a gadget is of a particular form. To conclude this section, we present a large class of gadgets for which the cost matrix formalism allows computation of accumulating errors upon iterated, complete gadget composition to go forward easily.

Definition C.2 (Circuit gadget)**.** Let an (a, b) gadget $\mathfrak{G}$ be a *circuit gadget* if it achieves unitaries U'_k via mapping input unitaries U_k to an m-qubit quantum circuit, where the unitary U'_k is achieved via $(c_{0k}, \ldots, c_{(a-1)k})$ (the k-th column of a cost matrix) queries to $(U_0, \ldots, U_{a-1})$ on any of the m qubits, interspersed with fixed m-qubit unitaries. □

Clearly, antisymmetric atomic gadgets are a particular type of circuit gadget. There is a simple upper-bound on the error accumulated within circuit gadgets.

Theorem C.1 (Linear error accumulation in circuit gadgets)**.** Suppose $\mathfrak{G}$ is an (a, b) circuit gadget. In addition, suppose V_k and V'_k are sets of unitaries such that $||V_k - V'_k|| \leq \varepsilon_k$, for each $k \in [a]$. Then the output error of the k-th output unitary achieved by the gadget can be upper-bounded as

$$||\mathfrak{G}[V_0, \ldots, V_{a-1}]_k - \mathfrak{G}[V'_0, \ldots, V'_{a-1}]_k|| \leq \sum_{j=0}^{a-1} c_{jk}\varepsilon_j = v_\varepsilon^T C e_k \tag{52}$$

where $v_\varepsilon = (\varepsilon_0, \ldots, \varepsilon_{a-1})$ and e_k is the k-th standard basis vector. □

Proof. By definition of the circuit gadget, we can write

$$\mathfrak{G}[V_0, \ldots, V_{a-1}]_k = W_0 \prod_{j=1}^{M} V_{s_j} W_j \tag{53}$$

for some length-M sequence $s \in [b]^M$, where $M = \sum_j C_{jk}$ (the total number of queries to input oracles characterizing the k-th output leg), and each W_j is a fixed M-qubit unitary. It follows that we can bound the difference of gadget actions via a telescoping series,

$$\begin{aligned}
\left|\left|\mathfrak{G}[V_0, \ldots, V_{a-1}]_k - \mathfrak{G}[V'_0, \ldots, V'_{a-1}]_k\right|\right| &= \left|\left| W_0 \prod_{j=1}^{M} V_{s_j} W_j - W_0 \prod_{j=1}^{M} V'_{s_j} W_j \right|\right| \\
&= \left|\left| \sum_{i=1}^{M} \left[\prod_{j<i} V_{s_j} W_j\right] (V_{s_i} W_i - V'_{s_i} W_i) \left[\prod_{j>i} V'_{s_j} W_j\right] \right|\right| \\
&\leq \sum_{i=1}^{M} \left|\left| V_{s_i} W_i - V'_{s_i} W_i \right|\right| \\
&= \sum_{i=1}^{M} ||V_{s_i} - V'_{s_i}|| \leq \sum_{j=1}^{a-1} C_{jk}\varepsilon_j = v_\varepsilon^T C e_k,
\end{aligned} \tag{54}$$

and the proof is complete. □

In general, in a sequence of gadget compositions, there are two sources of error: error in the unitaries inputted to a particular gadget, and error due to the imperfections in the gadget protocol itself. Both types of error can be understood in the the following results. First, consider what is essentially a corollary of the previous theorem.

Corollary C.1.1 (Output error in circuit gadgets)**.** Let $\mathfrak{G}$ be an (a,b) circuit gadget with cost matrix C which achieves, in its j-th leg, an ε'_j-approximation of the function $\mathfrak{F}_j$, mapping a individual unitaries to a single unitary, over the domain $\mathcal{D}$, so for $W_0,\dots,W_{a-1}\in\mathcal{D}$,

$$||\mathfrak{G}[W_0,\dots,W_{a-1}]_j - \mathfrak{F}_j[W_0,\dots,W_{a-1}]|| \leq \varepsilon'_j. \tag{55}$$

for each j. Then, given sets of unitaries $V_0,\dots,V_{a-1}$ and $V'_0,\dots,V'_{a-1}$ for which $||V_k - V'_k|| \leq \varepsilon_k$ for $k\in[a]$ with $V'_k\in\mathcal{D}$, $\mathfrak{G}[V_0,\dots,V_{a-1}]_j$, for all $j\in[b]$, $(v_\varepsilon^T C e_j + \varepsilon'_j)$-approximately achieves $\mathfrak{F}_j[V'_0,\dots,V'_{a-1}]$, where $v_\varepsilon = (\varepsilon_0,\dots,\varepsilon_{a-1})$. □

Proof. Via Thm. C.1, $||\mathfrak{G}[V_0,\dots,V_{a-1}]_j - \mathfrak{G}[V'_0,\dots,V'_{a-1}]_j|| \leq v_\varepsilon C e_j$. Then, from a triangle inequality,

$$\begin{aligned}||\mathfrak{G}[V_0,\dots,V_{a-1}]_j - \mathfrak{F}_j[V'_0,\dots,V'_{a-1}]|| &\leq ||\mathfrak{G}[V_0,\dots,V_{a-1}]_j - \mathfrak{G}[V'_0,\dots,V'_{a-1}]_j|| \\ &\quad + ||\mathfrak{G}[V'_0,\dots,V'_{a-1}]_j - \mathfrak{F}_j[V'_0,\dots,V'_{a-1}]|| \leq v_\varepsilon C e_j + \varepsilon'_j,\end{aligned} \tag{56}$$

which completes the proof. □

Immediately, we are able to prove more sophisticated results about the error accumulation in n-fold gadget compositions.

Theorem C.2 (Error in n-fold composition of circuit gadgets)**.** Let $\mathfrak{G}^{(k)}$ be (a_k,a_{k+1}) circuit gadgets for k ranging from 0 to $n-1$ with cost matrices $C^{(k)}$. Let $\mathfrak{G}$ be the circuit gadget obtained from performing full composition of gadget legs, $\mathfrak{G}=\mathfrak{G}^{(n-1)}\circ\cdots\circ\mathfrak{G}^{(0)}$. Suppose the gadget $\mathfrak{G}^{(k)}$ achieves, in its j-th leg, an $\varepsilon_j^{(k)}$-approximation of the unitary-valued function $\mathfrak{F}_j^{(k)}$ over the domain $\mathcal{D}^{(k)}$. This is to say that for all $W_0,\dots,W_{a_k-1}\in\mathcal{D}^{(k)}$,

$$\left|\left|\mathfrak{G}^{(k)}[W_0,\dots,W_{a_k-1}]_j - \mathfrak{F}_j^{(k)}[W_0,\dots,W_{a_k-1}]\right|\right| \leq \varepsilon_j^{(k)}, \tag{57}$$

for each j. Moreover, suppose $\mathfrak{F}_j^{(k)}[\mathcal{D}^{(k)}]\subset\mathcal{D}^{(k+1)}$ for all j and all k. Let $\mathfrak{F}^{(k)}=(\mathfrak{F}_0^{(k)},\dots,\mathfrak{F}_{a_{k+1}}^{(k)})$ be the multi-output unitary function. Let $\mathfrak{F}=\mathfrak{F}^{(n-1)}\circ\cdots\circ\mathfrak{F}^{(0)}$. Consider sets of unitaries $V_0,\dots,V_{a_0-1}$ and $V'_0,\dots,V'_{a_0-1}$ with $V'_k\in\mathcal{D}^{(0)}$, and $||V_j - V'_j||\leq\varepsilon_j^{(-1)}$ for each j. Note that $k=-1$ indexing is used for notational convenience. The composition error is bounded via multiplication and summation of cost matrices, as

$$\left|\left|\mathfrak{G}[V_0,\dots,V_{a_0-1}]_j - \mathfrak{F}[V'_0,\dots,V'_{a_0-1}]_j\right|\right| \leq \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right] e_j, \tag{58}$$

where $v_{\varepsilon^{(k)}}=(\varepsilon_0^{(k)},\dots,\varepsilon_{a_{k+1}-1}^{(k)})$. □

Proof. We can proceed via induction. Cor. C.1.1 proves the case of $n=1$ immediately. We assume the case of n, and prove the case of $n+1$ via induction. Letting $\mathfrak{G}'=\mathfrak{G}^{(n-1)}\circ\cdots\circ\mathfrak{G}^{(0)}$ and $\mathfrak{F}'=\mathfrak{F}^{(n-1)}\circ\cdots\mathfrak{F}^{(0)}$, note that $\mathfrak{F}'[V'_0,\dots,V'_{a_0-1}]\in\mathcal{D}^{(n)}$. Thus,

$$\left|\left|\mathfrak{G}^{(n)}[\mathfrak{F}'[V'_0,\dots,V'_{a_0-1}]]_j - \mathfrak{F}_j^{(n)}[\mathfrak{F}'[V'_0,\dots,V'_{a_0-1}]]\right|\right| \leq \varepsilon_j^{(n)} \tag{59}$$

for each j. By the inductive hypothesis, we know that

$$||\mathfrak{G}'[V_0,\dots,V_{a_0-1}]_j - \mathfrak{F}'[V'_0,\dots,V'_{a_0-1}]_j|| \leq \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right] e_j \tag{60}$$

for each j. Let v be the vector containing each of these errors, so

$$v^T = \left(\sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right] e_0,\dots,\sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right] e_{a_n-1}\right) \tag{61}$$

$$= \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right]. \tag{62}$$

It follows from Thm. C.1 that

$$||\mathfrak{G}^{(n)}[\mathfrak{F}'[V'_0,\ldots,V'_{a_0-1}]]_j - \mathfrak{G}^{(n)}[\mathfrak{G}'[V_0,\ldots,V_{a_0-1}]]_j|| \leq v^T C^{(n)} e_j \tag{63}$$

$$\leq \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n-1} C^{(\ell)}\right] C^{(n)} e_j \leq \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n} C^{(\ell)}\right] e_j, \tag{64}$$

so via a triangle inequality,

$$\left|\left|\mathfrak{G}^{(n)}[\mathfrak{G}'[V_0,\ldots,V_{a_0-1}]]_j - \mathfrak{F}_j^{(n)}[\mathfrak{F}'[V'_0,\ldots,V'_{a_0-1}]]\right|\right| \leq \varepsilon_j^{(n)} + \sum_{k=-1}^{n-1} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n} C^{(\ell)}\right] e_j \tag{65}$$

$$= \sum_{k=-1}^{n} v_{\varepsilon^{(k)}}^T \left[\prod_{k<\ell\leq n} C^{(\ell)}\right] e_j. \tag{66}$$

This completes the proof. □

Remark C.1 (Domain nesting property)**.** As a final point, it is important to note that when gadgets are composed in networks, they will often have be required to have different domain constraints on which they achieve their corresponding functions, in order to achieve a valid n-fold composition. In particular, if we wish to compose the functions induced by gadgets $\mathfrak{G}^{(0)},\ldots,\mathfrak{G}^{(n-1)}$, which achieve functions $f_0,\ldots,f_{n-1}$ on domains $\mathcal{D}_0,\ldots,\mathcal{D}_{n-1}$ respectively, via interlinks $\mathfrak{I}_0,\ldots,\mathfrak{I}_{n-2}$ the domains must satisfy the *nesting property* in order to induce the function $f_{n-1}\circ_{\mathfrak{I}_{n-2}}\cdots\circ_{\mathfrak{I}_0} f_0$ over the domain $\mathcal{D}=\mathcal{D}_0$: that is to say that the sequence of sets $\mathcal{D}'_k = (f_k\circ_{\mathfrak{I}_{k-1}}\cdots\circ_{\mathfrak{I}_0} f_0)(\mathcal{D}_0)$ for all k from 0 to $n-1$ must be valid, in the sense that each $\mathcal{D}'_k$ must be well-defined. For example, in the case that each interlink is complete composition of functions, we must have $\mathcal{D}'_k = f_k(\mathcal{D}_k)\subset\mathcal{D}_{k+1}$ for each k to satisfy the domain nesting property. This constraints was alluded to in Thm. C.2, where the successive domains satisfy this constraint. Satisfaction of this constraint will often have non-trivial effects on the cost of composing gadgets, and care must be exercised to ensure that this condition is always met within a gadget network. □

D On QSP variants, constraints, protocols, and QSP-derived gadgets

This section summarizes the main results of quantum signal processing (QSP) [43–45] and multivariate quantum signal processing (M-QSP) [59], and clarifies the associated notation. As a supplement to the definition of M-QSP given in the main body (Def. 2.1), we depict the associated circuits again in Fig. 6. Recall the original result which characterizes the standard QSP unitary.

Theorem D.1 (Quantum signal processing; following [26, 47])**.** Let $k\in\mathbb{N}$; there exists $\Phi=\{\phi_0,\ldots,\phi_k\}\in\mathbb{R}^{k+1}$ such that $\forall x\in[-1,1]$,

$$U_{\mathrm{QSP}}(x;\Phi) \coloneqq \Phi[W(x)] = e^{i\phi_0\sigma_z}\prod_{j=1}^{k}\left(W(x)e^{i\phi_j\sigma_z}\right) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x)\end{pmatrix} \tag{67}$$

where σ_z is the Pauli-Z operation, and

$$W(x) \coloneqq \begin{pmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x\end{pmatrix} = e^{i\arccos(x)\sigma_x}, \tag{68}$$

where $\arccos(x)\in[0,\pi]$, if and only if complex polynomials $P(x),Q(x)\in\mathbb{C}[x]$ satisfy the following conditions:

1. $\deg(P)\leq k$ and $\deg(Q)\leq k-1$
2. P has parity k mod 2 and Q has parity $(k-1)$ mod 2
3. For all $x\in[-1,1]$, $|P(x)|^2+(1-x^2)|Q(x)^2|=1$

□

Also recall the main result characterizing the unitaries generated by M-QSP protocols.

Theorem D.2 (Laurent-picture M-QSP, following [59])**.** Let $\Phi = \{\phi_0, \ldots, \phi_k\} \in \mathbb{R}^{k+1}$ be a sequence of phase angles and let $s = \{s_1, \ldots, s_k\} \in \{0,1\}^k$. Then

$$U_{\text{M-QSP}}(x_1, x_2; \Phi, s) \coloneqq \Phi[W(x_1), W(x_2)] = e^{i\phi_0\sigma_z} \prod_{j=1}^{k} W(x_1)^{s_j} W(x_2)^{1-s_j} e^{i\phi_j\sigma_z} = \begin{pmatrix} P & Q \\ -Q^* & P^* \end{pmatrix} \tag{69}$$

where

$$W(x) \coloneqq \frac{x + x^{-1}}{2}\mathbb{I} + \frac{x - x^{-1}}{2}\sigma_x \tag{70}$$

for all $(x_1, x_2) \in \mathbb{T}^2$ if and only if $P, Q \in \mathbb{C}[x_1, x_2]$ are Laurent polynomials in x_1 and x_2, satisfying the following conditions:

1. $\deg(P) \preccurlyeq (m, n-m)$ and $\deg(Q) \preccurlyeq (m, n-m)$ where $n = |s|$ is the Hamming weight of s.
2. P has parity n mod 2 under $(x_1, x_2) \mapsto (x_1^{-1}, x_2^{-1})$ and Q has parity $(n-1)$ mod 2 under $(x_1, x_2) \mapsto (x_1^{-1}, x_2^{-1})$.
3. P has parity m mod 2 under $x_1 \mapsto -x_1$ and parity $m-n$ mod 2 under $x_2 \mapsto -x_2$. Q has parity $m-1$ mod 2 under $x_1 \mapsto -x_1$ and parity $m-n-1$ mod 2 under $x_2 \mapsto -x_2$.
4. For all $(x_1, x_2) \in \mathbb{T}^2$, we have $|P|^2 + |Q|^2 = 1$.
5. The FRT = QSP condition (that satisfaction of the multivariable Fejér-Riesz lemma, and thus a unitary completion, implies the existence of QSP phases), formulated in [59] as a conjecture, holds. □

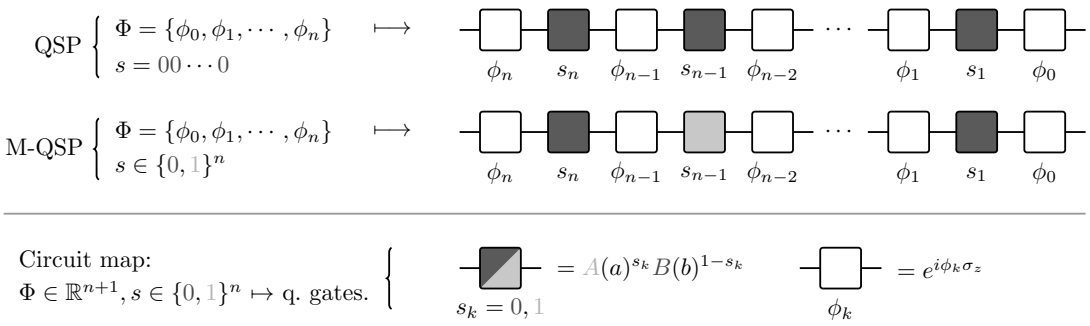


Figure 6: Explicit depiction of the circuit forms of QSP and M-QSP, following Def. 2.1, simplified to the case of two variables. Protocols are specified by an ordered list of real phases Φ and an ordered list of oracle labels s, where s prescribes the order in which oracles (from a set of possible oracles) are applied, and Φ the σ_z-rotations interspersing them. Standard QSP is the special case where s consists of a single repeated label. Such circuits are also given by Def. 2.1, and depicted in (a) of Fig. 1 using the same style. Note that this notation is distinct from that of linked gadgets, as discussed in Rem. I.2, and shown in (c) of Fig. 1.

When it comes to embedding particular polynomials P and Q in QSP and M-QSP protocols, it becomes necessary to formulate existence results which allow one, given some desired polynomial P, to find a polynomial Q such that P and Q satisfy the constraints of QSP or M-QSP. [26] proves a range of existence results for the case of standard, single-variable QSP. We restate some of these results below.

Theorem D.3. Let $Q \in \mathbb{C}[x]$. There exists a P such that P and Q satisfy the conditions set forth in Thm. D.1 if and only if

1. For all $x \in [-1, 1]$, $\sqrt{1-x^2}|Q(x)| \leq 1$.
2. In the case that k is odd (so Q has even parity), for all $x \in \mathbb{R}$, $(1+x^2)Q^*(ix)Q(ix) \geq 1$.

Similarly, given $P \in \mathbb{C}[x]$, there exists a Q such that P and Q satisfy the conditions of Thm. D.1 if and only if

1. For all $x \in [-1, 1]$, $|P(x)| \leq 1$.
2. For all $x \in (-\infty, -1] \cup [1, \infty)$, $|P(x)| \geq 1$.

3. In the case that k is even (so P has even parity), for all $x \in \mathbb{R}$, $P^*(ix)P(ix) \geq 1$.

□

Theorem D.4. Take $k \in \mathbb{N}$, let $\tilde{P}, \tilde{Q} \in \mathbb{R}[x]$. There exist $P, Q \in \mathbb{C}[x]$ such that P and Q with $\tilde{P} = \mathrm{Re}[P]$ and $\tilde{Q} = \mathrm{Re}[Q]$ satisfying conditions 1-3 of Thm. D.1, if and only if P and Q satisfy conditions 1-2 of Thm. D.1 and $|\tilde{P}(x)|^2 + (1-x^2)|\tilde{Q}(x)|^2 \leq 1$. □

Generally speaking, proving existence results for pairs of QSP polynomials is much harder in the multivariate case, as is discussed in Sec. 1. It is this very issue, among others, that this paper attempts to address with the techniques introduced.

D.1 QSP on σ_z-rotations

We proceed by describing a handful of QSP-related results which are "folkloric" within the existing QSP literature, and will be of use in proofs throughout the appendices, in particular those relating to the root protocols of Appx. F.3. In a standard QSP protocol, the "signal operator" is taken to be a fixed σ_x-rotation. Via conjugation by Hadamards, a σ_x-rotation is clearly dual to a σ_z-rotation, and vice versa. Thus, given access to a σ_z-rotation oracle, one can perform a simple rotation where the oracle becomes a σ_x-rotation, perform a QSP protocol with this signal, and rotate back into the σ_z-picture. It is protocols of this form that are considered in this section.

As was stated, $He^{i\sigma_z t}H = e^{iH\sigma_z Ht} = e^{i\sigma_x t}$. It follows from this fact that when $t \in [0, \pi]$,

$$\Phi[He^{i\sigma_z t}H] = \Phi[e^{i\sigma_x t}] = U_{\mathrm{QSP}}(\cos(t); \Phi), \tag{71}$$

which is implied by Thm. D.1. Now, consider the case when $t \in [-\pi, 0]$. It is true that $e^{i\sigma_x t} = \sigma_z e^{-i\sigma_x t}\sigma_z$, where $-t \in [0, \pi]$, so it follows that in this case,

$$\Phi[He^{i\sigma_z t}H] = \sigma_z \Phi[e^{-i\sigma_x t}]\sigma_z = \sigma_z U_{\mathrm{QSP}}(\cos(-t); \Phi)\sigma_z = \sigma_z U_{\mathrm{QSP}}(\cos(t); \Phi)\sigma_z. \tag{72}$$

This implies the following result.

Lemma D.1. Given $t \in [-\pi, \pi]$,

$$\Phi[He^{i\sigma_z t}H] = \begin{pmatrix} P(\cos(t)) & i\sin(t)Q(\cos(t)) \\ i\sin(t)Q^*(\cos(t)) & P^*(\cos(t)) \end{pmatrix} \tag{73}$$

where P and Q are the QSP-polynomials generated by Φ. □

Going forward, for some Φ, let $H\Phi[He^{i\sigma_z t}H]H = \Phi'[e^{i\sigma_z t}]$. We call the operator Φ' a σ_z-QSP protocol. It is straightforward to verify that

$$\langle 0|\Phi'[e^{i\sigma_z t}]|0\rangle = \mathrm{Re}[P](\cos(t)) + i\sin(t)\mathrm{Re}[Q](\cos(t)) \tag{74}$$

$$\langle 0|\Phi'[e^{i\sigma_z t}]|1\rangle = \mathrm{Im}[P](\cos(t)) + i\sin(t)\mathrm{Im}[Q](\cos(t)) \tag{75}$$

as well as $\langle 1|\Phi'[e^{i\sigma_z t}]|1\rangle = \langle 0|\Phi'[e^{i\sigma_z t}]|0\rangle^*$ and $\langle 1|\Phi'[e^{i\sigma_z t}]|0\rangle = -\langle 0|\Phi'[e^{i\sigma_z t}]|1\rangle^*$. Now, suppose p is a degree-n Laurent polynomial $p(x) = \sum_{j=-n}^{n} p_j x^j$ in $\mathbb{R}[x, x^{-1}]$. Clearly,

$$p(e^{it}) = \sum_{j=-n}^{n} p_j e^{ijt} = \sum_{j=-n}^{n} p_j \left[\cos(jt) + i\sin(jt)\right] = p_0 + \sum_{j=1}^{n} (p_j + p_{-j})\cos(jt) + i(p_j - p_{-j})\sin(jt). \tag{76}$$

Note that $p(e^{-it}) = p(e^{it})^*$, for a real polynomial p. For any positive integer j, note that $\cos(jt)$ can be written as a degree-j polynomial T_j acting on $\cos(t)$: this is simply the j-th Chebyshev polynomial of the first-kind. The identity $\sin(jt) = \sin(t)U_{j-1}(\cos(t))$ also holds, where U_j is the j-th Chebyshev polynomial of the second-kind. It follows that

$$p(e^{it}) = \sum_{j=0}^{m} a_j T_j(\cos(t)) + i\sum_{j=1}^{m} b_j \sin(t)U_{j-1}(\cos(t)) = p_A(\cos(t)) + i\sin(t)p_B(\cos(t)), \tag{77}$$

where we have let $a_j = p_j + p_{-j}$, $b_j = p_j - p_{-j}$ for $j \geq 1$, $a_0 = p_0$, and we have defined p_A and p_B, yielded from p, as the degree m and $m-1$ polynomials

$$p_A(x) \coloneqq \sum_{j=0}^{m} a_j T_j(x) \quad \text{and} \quad p_B(x) \coloneqq \sum_{j=1}^{m} b_j U_{j-1}(x). \tag{78}$$

Since the Chebyshev polynomials of both kinds, up to degree-n, form bases for the space of all degree-n polynomials, it follows that there exists a well-defined, natural map between real Laurent polynomials of degree-n, and pairs of real polynomials of degree m and $m-1$ respectively. We characterize this map in the following result.

Definition D.1. For $\mathbb{F}$ a field, let $\mathbb{F}[x_1,\ldots,x_k]_n$ be the vector space of all polynomials in $x_1,\ldots,x_k$ of degree less than or equal to n with coefficients in $\mathbb{F}$. □

Theorem D.5. The map $\Gamma : \mathbb{R}[z,z^{-1}]_m \to \mathbb{R}[x]_m \times \mathbb{R}[x]_{m-1}$ such that $\Gamma(p) = (p_A, p_B)$, where p_A and p_B are defined in Eq. (78), is well-defined and bijective. □

Proof. Clearly, this map is well-defined, by construction. Moreover, given a pair $(P,Q) \in \mathbb{R}[x]_m \times \mathbb{R}[x]_{m-1}$, we have $P(x) = \sum_{j=0}^m a_j T_j(x)$ and $Q(x) = \sum_{j=1}^m b_j U_{j-1}(x)$. The Chebyshev polynomials are bases, so the induced p_j and q_j are unique. We then let $r_j = \frac{a_j+b_j}{2}$ and $r_{-j} = \frac{a_j-b_j}{2}$, for $j \geq 1$. We also let $r_0 = a_0$, and let $r(z) = \sum_{j=-m}^m r_j z^j$. Define $\Gamma^{-1}(P,Q) = r$. It is trivial to check that Γ^{-1} is a well-defined inverse of Γ, so the map is bijective. □

Remark D.1 (Bijection of complex polynomials)**.** We know that $\mathbb{C}[z,z^{-1}]_m \simeq \mathbb{R}[z,z^{-1}]_m \times \mathbb{R}[z,z^{-1}]_m$, and $\mathbb{C}[x]_m \simeq \mathbb{R}[x]_m \times \mathbb{R}[x]_m$. Thus, $\Lambda \simeq \Gamma \times \Gamma$, where the identification of maps is done in the obvious way of

$$p + iq \simeq (p,q) \mapsto \Gamma(p) \times \Gamma(q) = (p_A, p_B) \times (q_A, q_B) \simeq (p_A + iq_A, p_B + iq_B), \tag{79}$$

so $p + iq \mapsto (p_A + iq_A, p_B + iq_B)$ acts as a bijective map from $\mathbb{C}[z,z^{-1}]_m$ to $\mathbb{C}[x]_m \times \mathbb{C}[x]_{m-1}$, where the real and imaginary parts of the standard and Laurent polynomials are in bijective correspondence. □

Remark D.2 (Correspondence of QSP and σ_z-QSP)**.** Using this map, it is easy to see the correspondence between standard QSP protocols and σ_z-QSP protocols induced by this map. In particular, if Φ is a QSP protocol inducing (P,Q), then it follows immediately from Eq. (77), as well as Eq. (74) and Eq. (75) and the definition of Γ that $\Phi'[e^{i\sigma_z t}]$ is the complex Laurent polynomial $\Lambda^{-1}(P,Q)$. □

Now, let us consider relevant subsets of $\mathbb{C}[x]_m \times \mathbb{C}[x]_{m-1}$, which will correspond to all admissible pairs of QSP polynomials of particular degree.

Lemma D.2. Let $S_1 \subset \mathbb{C}[x]_m \times \mathbb{C}[x]_{m-1}$ be the subset of polynomials (P,Q) such that $|P(x)|^2 + (1-x^2)|Q(x)|^2 \leq 1$ for all $x \in [-1,1]$. Then $\Lambda^{-1}(S_1) \subset \mathbb{C}[z,z^{-1}]_m$ is precisely the subset of $\mathbb{C}[z,z^{-1}]_m$ with $|p(z)| \leq 1$ for all $z \in \mathbb{T}$. □

Proof. Using Eq. 77, and the definition of Λ, as well as the fact that $p = \Re[p] + i\Im[p]$, we note that

$$\begin{aligned}
|p(e^{it})|^2 &= |\Re[p](e^{it})|^2 + |\Im[p](e^{it})|^2 && (80)\\
&= |\Re[p]_A(\cos(t))|^2 + |\sin(t)|^2|\Re[p]_B(\cos(t))|^2 + |\Im[p]_A(\cos(t))|^2 + |\sin(t)|^2|\Im[p]_B(\cos(t))|^2 && (81)\\
&= |(\Re[p]_A + i\Im[p]_A)(\cos(t))|^2 + |\sin(t)|^2|(\Re[p]_B + i\Im[p]_B)(\cos(t))|^2 && (82)\\
&= |\Lambda(p)_1(\cos(t))|^2 + |\sin(t)|^2|\Lambda(p)_2(\cos(t))|^2 && (83)
\end{aligned}$$

for all t. Given $|p(e^{it})| \leq 1$, for some $x \in [-1,1]$, we can pick t such that $\cos(t) = x$ and $\sin(t)^2 = 1-x^2$, so we will have $|\Lambda(p)_1(x)|^2 + (1-x^2)|\Lambda(p)_2(x)|^2 \leq 1$. Thus, $\Lambda(p) \in S_1$, so $p \in \Lambda^{-1}(S_1)$. Given $p \in \Lambda^{-1}(S_1)$, note that $\Lambda(p) \in S_1$, so again using the above formula, $|p(e^{it})| \leq 1$. □

Lemma D.3. Let $S_2 \subset \mathbb{C}[x]_m \times \mathbb{C}[x]_{m-1}$ be the subset of polynomials (P,Q) such that $P(x)$ has parity m mod 2 and $Q(x)$ has parity $(m-1)$ mod 2. Then $\Lambda^{-1}(S_2) \subset \mathbb{C}[z,z^{-1}]_m$ is precisely the subset of $\mathbb{C}[z,z^{-1}]_m$ with p having parity m mod 2. □

Proof. Since the parities of $T_j(x)$ and $U_j(x)$ are j mod 2, when polynomials P and Q with $(P,Q) \in S_2$ are expanded in this basis according to Eq. (78), only coefficients of index j such that $j \bmod 2 = m \bmod 2$ can be non-zero. It follows immediately that only coefficients of $\Lambda^{-1}(P,Q)$ of power j with $j \bmod 2 = m \bmod 2$ can be non-zero, impyling that $\Lambda^{-1}(P,Q)$ has parity m mod 2.

Conversely, suppose $p \in \mathbb{C}[z,z^{-1}]$ has parity m mod 2, so only powers j with $j \bmod 2 = m \bmod 2$ can be non-zero in p. Then it follows from the definition of Λ and the parities of T_j and U_j that the polynomials of the pair $(P,Q) = \Lambda(p)$ will have parity j mod 2 and $(m-1)$ mod 2, respectively. □

Clearly, $S = S_1 \cap S_2$ is the set of all admissible QSP polynomial pairs (P,Q). Since Λ is a bijection, $\Lambda^{-1}(S) = \Lambda^{-1}(S_1 \cap S_2) = \Lambda^{-1}(S_1) \cap \Lambda^{-1}(S_2)$. In the dual σ_z-picture, it follows from this fact, along with Rem. D.2, that we have the next result.

Theorem D.6. There exists a length-n σ_z-QSP protocol Φ' which yields a unitary $\Phi'[e^{i\sigma_z t}]$ with $\langle 0|\Phi'[e^{i\sigma_z t}]|0\rangle = \Re[p] \in \mathbb{R}[e^{it}, e^{-it}]$ and $\langle 0|\Phi'[e^{i\sigma_z t}]|1\rangle = \Im[p] \in \mathbb{R}[e^{it}, e^{-it}]$ if and only if $\deg(p) \leq n$, p has parity n mod 2, and $|p(e^{it})| \leq 1$ for all $e^{it} \in \mathbb{T}$. Moreover, for some $p \in \mathbb{C}[e^{it}, e^{-it}]$, the protocol Φ yields the pair $(P, Q) = \Lambda(p)$ in the standard σ_x-picture. □

Proof. If p satisfies the given properties, then $p \in \Lambda^{-1}(S)$ from the above results, and $\Lambda(p) = (P, Q) \in S$ will be a valid pair of QSP polynomials, so there exists a protocol Φ yielding them. It is then easy to see that Φ' yields p in the σ_z-picture. Conversely, if Φ' is a σ_z-QSP protocol yielding p, then Φ yields polynomials $(P, Q) = \Lambda(p) \in S$, so $p \in \Lambda^{-1}(S)$, and therefore satisfies the outlined properties. □

D.2 Embeddability properties of M-QSP protocols

It is important to make note of certain useful facts about the embeddability properties of different types of M-QSP protocols.

Theorem D.7. Given a length-n antisymmetric M-QSP protocol (Φ, s), the unitary $\Phi[U_0, \ldots, U_{t-1}]$ is twisted embeddable for any set of twisted embeddable unitaries $U_0, \ldots, U_{t-1}$. □

Proof. When using the superoperator $(N \circ R)(\Phi)$, we also assume implicitly that we have sent $s \mapsto R(s)$ as well. In the case that (Φ, s) is antisymmetric, $(N \circ R)(\Phi) = \Phi$ and $R(s) = s$. Thus, for some set of embeddable unitaries $U_0, \ldots, U_{t-1}$,

$$\Phi[U_1, \ldots, U_{t-1}]^\dagger = e^{-i\phi_n \sigma_z} \prod_{k=1}^{n} U_{s_{n-k+1}}^\dagger e^{-i\phi_{n-k}\sigma_z}. \tag{84}$$

Since each U_k is twisted embeddable, we have $U_k = e^{i\varphi_k\sigma_z/2} e^{i\theta_k\sigma_x} e^{-i\varphi_k\sigma_z/2}$, for some θ_k and φ_k, immediately implying that $U_k^\dagger = \sigma_z U_k \sigma_z$ for each k. Moreover, because $(N \circ R)(\Phi) = \Phi$, we have $-\phi_n = \phi_0$ and $-\phi_{n-k} = \phi_k$ for each k. Since $R(s) = s$, we have $s_{n-k+1} = s_k$ for each k. Thus,

$$e^{-i\phi_n\sigma_z} \prod_{k=1}^{n} U_{s_{n-k+1}}^\dagger e^{-i\phi_{n-k}\sigma_z} = e^{-i\phi_n\sigma_z} \prod_{k=1}^{n} \sigma_z U_{s_{n-k+1}} \sigma_z e^{-i\phi_{n-k}\sigma_z} = \sigma_z e^{i\phi_0\sigma_z} \prod_{k=1}^{n} U_{s_k} e^{i\phi_k\sigma_z} \sigma_z \tag{85}$$

$$= \sigma_z \Phi[U_1, \ldots, U_{t-1}] \sigma_z, \tag{86}$$

and immediately, $\langle 0|\Phi[U_1, \ldots, U_{t-1}]|0\rangle = \langle 0|\Phi[U_1, \ldots, U_{t-1}]^\dagger|0\rangle$, so the the top-left entry is real. Since $\Phi[U_1, \ldots, U_{t-1}]$ is SU(2), this implies immediately that the resulting unitary has a real diagonal, and thus the complex phase of the off-diagonal can be factored out as a σ_z-conjugation, implying $\Phi[U_1, \ldots, U_{t-1}]$ is twisted embeddable. □

Lemma D.4 (Linear error accumulation in M-QSP protocols)**.** Let (Φ, s) be a length-$(n+1)$ M-QSP protocol. Let $U_0, \ldots, U_{t-1}$ be a set of unitaries. Let $U_0', \ldots, U_{t-1}'$ be a set of unitaries such that $||U_k - U_k'|| \leq \varepsilon$ for all k. Then

$$||\Phi[U_0, \ldots, U_{t-1}] - \Phi[U_0', \ldots, U_{t-1}']|| \leq n\varepsilon. \tag{87}$$

□

Proof. This follows trivially from Thm. C.1. □

Corollary D.7.1. The set of length-$(n+1)$, antisymmetric M-QSP protocols map ε-twisted embeddable unitaries to $n\varepsilon$-twisted embeddable unitaries. □

Proof. Let U be ε-twisted embeddable, so $||U - U'|| \leq \varepsilon$ for twisted embeddable U'. From Thm. D.7, $\Phi[U']$ is twisted embeddable. From Lem. D.4, $||\Phi[U] - \Phi[U']|| \leq n\varepsilon$, so $\Phi[U]$ is $n\varepsilon$-twisted embeddable. □

The correction protocols outlined in Thm 2.1, while effective, are sometimes costly, and become much more costly in contexts where they must be called many times for a recursive nesting of gadgets. Thus, it is a problem of practical importance to consider instances in which the correction protocols are either not required, or can be implemented with fewer resources. Many gadget networks, such as those presented in the examples of Sec. 4, make use of both single-variable and multi-variable M-QSP protocols (atomic gadgets). In fact, it is often the case that the bulk of the gadgets implemented within a network are $(1, 1)$ atomic (single-variable QSP protocols). As is remarked upon through this work, the single-variable QSP protocol is much easier to analyze than its multivariate generalization. In fact, as we will demonstrate in this section, it is possible to characterize classes of QSP protocols which will be *generically* ε-embeddable, and thus will not require correction prior to composition. To begin, consider the following example.

Example D.1 (Embeddability of trivial QSP protocols)**.** Let $\Phi^{(k)} = \{0, \dots, 0\}$ be the length-k trivial QSP protocol, which is known the achieve the function $T_k(x)$: the k-th Chebyshev polynomial. Given an embeddable unitary of the form $U = e^{i\theta\sigma_x}$, $U' = \Phi^{(k)}[U] = e^{ik\theta\sigma_x}$ is *not* generally embeddable, due to the fact that $k\theta$ could be in the region $[-\pi, 0]$, in which case $e^{ik\theta\sigma_x} = \sigma_z e^{-ik\theta\sigma_x}\sigma_z$, where $e^{-ik\theta\sigma_x}$ is embeddable. Thus, in general, U' is twisted embeddable, but it is always the case that either U' *or* $\sigma_z U'\sigma_z$ is embeddable. Moreover, U' can be easily made embeddable (via σ_z-conjugation) when $\cos(k\theta)$ lies between two, *known* neighbouring roots of $T_k(x)$. □

We will demonstrate that the Chebyshev polynomials are the *only* family of polynomials which satisfy this condition: all other single-variable QSP protocols are twisted embeddable (Thm. D.7). We can provide a complete description of all single-variable QSP protocols which are ε-embeddable.

Remark D.3 (ε-embeddability of QSP protocols)**.** Φ, a QSP protocol, will yield an ε-embeddable unitary if and only if $\Phi[e^{i\theta\sigma_x}]$ is an ε-approximation of some phase function $e^{if(\theta)\sigma_x}$ for $f(\theta) \in [0, \pi]$, for each θ. This is somewhat easier to interpret in the σ_z-QSP picture, where ε-embeddable QSP protocols Φ will correspond to σ_z-QSP protocols Φ' such that $\Phi'[e^{i\sigma_z t}]$ is an ε-approximation of $e^{i\sigma_z f(t)}$ for $f(t) \in [0, \pi]$. In this picture, since the functional object being considered is a Laurent polynomial of e^{it}, it is easy to see that the only case where $\Phi'[e^{i\sigma_z t}]$ can yield *precisely* an operator of the form $e^{i\sigma_z f(t)}$ is when the Laurent polynomial of Φ', p, satisfies $\Re[p](e^{it}) = e^{if(t)}$ and $\Im[p] = 0$. Thus,

$$\sum_{j=-m}^{m} p_j e^{itj} = e^{if(t)} \implies |\Re[p](e^{it})|^2 = \sum_{k>j} 2p_k p_j \cos((k-j)t) = 1 - \sum_k p_k^2 \tag{88}$$

for all t. One can verify inductively, using the linear independence of the family of functions $\cos(kt)$, that this implies all but a single coefficient in the polynomial is non-zero. In the standard QSP picture, these cases will correspond precisely to the trivial, Chebyshev polynomial protocols. □

Despite the fact that the unitaries which are precisely embeddable are rather limited, there does exist a rich class of approximately embeddable QSP protocols (up to possible σ_z-conjugation): namely those which approximate phase functions $e^{it} \mapsto e^{if(t)}$ in the Laurent picture. These functions include arbitrary real roots $e^{it} \mapsto e^{irt}$, exponentiated polynomials more generally $e^{it} \mapsto e^{ip(t)}$, or even exponentiated trigonometric functions of the form $e^{it} \mapsto e^{i\cos(t)}$, which can be approximated via the Jacobi-Anger expansion.

Remark D.4 (Half-twisted embeddability)**.** The condition of half-twisted embeddability is required for implementation of the ancillla-free root protocol, and is in general harder to characterize than embeddability. Consider the output of an antisymmetric single-variable QSP protocol, which will take the form

$$e^{i\varphi(x)\sigma_z/2} e^{i\sigma_x \arccos(P(x))} e^{-i\varphi(x)\sigma_z/2} = \begin{pmatrix} P(x) & i\sqrt{1-P(x)^2}e^{i\varphi(x)} \\ i\sqrt{1-P(x)^2}e^{-i\varphi(x)} & P(x) \end{pmatrix} \tag{89}$$

where $Q(x) = |Q(x)|e^{i\varphi(x)} = \sqrt{\frac{1-P(x)^2}{1-x^2}}e^{i\varphi(x)}$. Here, $Q \in \mathbb{C}[x]$ is separated into its magnitude and phase, which are both functions of x. The condition that $\varphi(x) \in [-\pi/2, \pi/2]$ for all x in a given range (in other words, half-twisted embeddability) is precisely equivalent to the real component of $Q(x)$ being non-negative. Thus, we are particularly interested in pairs of QSP polynomials (P, Q) for which P is real and Q has non-negative real part. This is an opaque constraint, and therefore, we will often be forced to check whether a given QSP or M-QSP protocol achieves this condition on a case-by-case basis. □

D.3 On QSP-derived and general gadgets

We conclude this section by taking some time to discuss and disambiguate the various forms of *gadgets* define in this work; some of these definitions relate closely to QSP and M-QSP circuits, and the explicit circuits analyzed in this work are all derived from such circuits. The abstract classes of gadgets we define do not all constitute strict subsets or supersets of one-another, although again the relevant sub-classes we work with most often do obey such simple containment relations.

In this work we define *atomic gadgets* (Def. 2.3), *gadgets* (Def. 2.4), as well as *circuit gadgets* (Def. C.2). The first of these, *atomic gadgets*, (Def. 2.3) is a much more general definition of collections of M-QSP circuits, and only become true *gadgets* under special conditions, namely when they are antisymmetric (Def. 2.1), or atypical (Def. D.2) and properly pinned. Consequently *atomic gadgets* are *not* a subset of *gadgets*, though the sub-class of atomic gadgets we consider in this work are all also gadgets. Finally,

circuit gadgets (Def. C.2) are indeed all gadgets, and merely restrict the abstract definition of gadgets to those realized from discrete circuits over a finite collection of qubits. Summarizing these relations diagrammatically, we refer to Fig. 7.

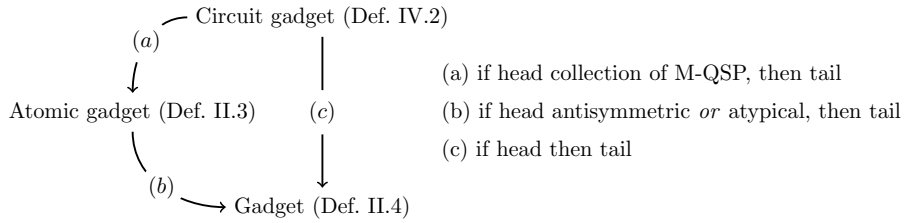


Figure 7: A diagrammatic summary of the relation between *atomic gadgets* (Def. 2.3), *gadgets* (Def. 2.4), and *circuit gadgets* (Def. C.2). The labels on arrows indicate that the relation should be read as follows: if the object is the *head* of the arrow, and satisfies the given condition(s), then the object is also the *tail* of the arrow.

In addition to antisymmetric atomic gadgets, which are the primary focus of this work, the second subclass of atomic gadgets which are also standard gadgets are *atypical gadgets* (Def. D.2), which we define below. We discuss these only in the $(1,1)$ case, but in general such atypical gadgets can exist for any number of input legs; the general condition follows from finding roots of the imaginary part of the P achieved by a general atomic gadget, and pinning (Def. A.1) to one of those roots. As we will not need such gadgets for our examples, and their theory is considerably less well-understood given its connection to M-QSP [59], we leave discussion of them to future work. Nevertheless, their existence leave a variety of interesting questions open in the construction of highly expressive atomic gadgets.

Definition D.2 (Atypical atomic gadget)**.** An atomic (a,b) gadget $\mathfrak{G}$ is *atypical* if there exists a pinning $\mathfrak{G}'$ of $\mathfrak{G}$ such that all output legs $\mathfrak{G}'_k, k \in [b]$ are twisted embeddable, but this *does not hold for all pinnings* $\mathfrak{G}'$. Note that $\mathfrak{G}$ need *not* be antisymmetric. For brevity, we call *both* $\mathfrak{G}$ and $\mathfrak{G}'$ atypical gadgets, and almost always work directly with $\mathfrak{G}'$. □

Example D.2 (Atypical atomic $(2,1)$ gadget)**.** Consider the prescription given by [26] (reproduced in Thm. D.4) for finding a QSP protocol with phases Φ for achieving the pair of functions P, Q for arbitrary choice of $\tilde{P}, \tilde{Q}$ such that $|\tilde{P}|^2 + (1-x^2)|\tilde{Q}|^2 \leq 1$, with $\Re[P] = \tilde{P}$ and $\Re[Q] = \tilde{Q}$. Moreover, choose $\tilde{Q} = 0$. Then consider the (non-antisymmetric) atomic $(2,1)$ gadget (Def. D.2) defined by the following (Ξ, S):

$$\Xi = \{\{0, \pi/4, 0, -\pi/4\} \cup \Phi \cup \{\pi/4, 0, \pi/4, 0\}\}, \tag{90}$$

$$S = \{\{1, 1, 0, 0, \ldots, 0, 0, 1, 1\}\}, \tag{91}$$

where Φ are the (symmetric [74]) phases achieving $\tilde{P}$ and $\tilde{Q} = 0$, and where S contains 0 where not otherwise indicated. It is easy to determine that this gadget does *not* output a twisted-embeddable unitary for general choice of U_1; in fact, this gadget achieves the following function, where $U_1 = e^{i\theta\sigma_z}$:

$$\langle 0|\mathfrak{G}[U_0, U_1]|0\rangle = i(\tilde{P} + \tilde{P}')\cos(2\theta) - \tilde{P}\sin^2(2\theta), \tag{92}$$

where $\tilde{P}' = \Im[P]$ as achieved by the QSP protocol with phases Φ. For $\theta = \pi/4 + k\pi/2, k \in \mathbb{Z}$, however, this gadget achieves $-\tilde{P}$, and in fact these are the only θ at which a real function is achieved for general $\tilde{P}$. Thus pinning (Def. A.1) $U_1 = e^{i(\pi/4)\sigma_z}$ produces an atypical atomic $(1,1)$ gadget achieving an arbitrary real, bounded, definite parity $\tilde{P}$. □

E Extraction of an unknown σ_z-conjugation

This section is dedicated to the problem of mapping a twisted embeddable oracle,

$$U = e^{i\varphi\sigma_z/2}e^{i\theta\sigma_x}e^{-i\varphi\sigma_z/2} = e^{i\varphi\sigma_z/2}e^{i\arccos(x)\sigma_x}e^{-i\varphi\sigma_z/2} = e^{i\varphi\sigma_z/2}W(x)e^{-i\varphi\sigma_z/2}, \tag{93}$$

to a power of the corresponding, unknown σ_z-rotation (in particular, $e^{-i\varphi\sigma_z}$). Recall that QSP protocols are invariant (in a sense) under σ_z-conjugation. That is to say, given a QSP protocol Φ and some arbitrary σ_z-rotation $e^{i\varphi\sigma_z/2}$,

$$\Phi[e^{i\varphi\sigma_z/2}Ue^{-i\varphi\sigma_z/2}] = e^{i\varphi\sigma_z/2}\Phi[U]e^{-i\varphi\sigma_z/2} \tag{94}$$

for any unitary U. Our general strategy for extracting the desired σ_z-rotation is to utilize this property of QSP to map $W(x)$ to a fixed, known unitary over all x, so that it can be cancelled, leaving behind the desired σ_z-rotation.

E.1 Constructing the extraction QSP polynomial

We wish to construct a polynomial $Q(x)$ which is close to 1 for $x \in \mathcal{D} = [-1+\delta, 1-\delta]$. The associated QSP protocol will drive an arbitrary gate of the form $W(x)$ towards $i\sigma_x$ when $x \in \mathcal{D}$. Let $A : (-1,1) \to \mathbb{R}$ be the function defined by $A(x) = \frac{1}{\sqrt{1-x^2}}$. Let $A_n(x)$ denote the $(n-1)$-th order Taylor approximation of $A(x)$. From the generalized binomial formula,

$$A_n(x) = \sum_{k=0}^{n-1} \binom{-\frac{1}{2}}{k} (-1)^k x^{2k}. \tag{95}$$

From Lem. G.4 of Appx. G, it follows immediately that the coefficients of $A_n(x)$ are weakly decreasing in magnitude, with positive sign. Thus, the Taylor series has a radius of convergence that at least contains the interval $(-1,1)$. Cor. G.1.1 now holds with $\lambda = 2$. Note that

$$M = |f(1-\delta)| = \frac{1}{\sqrt{1-(1-\delta)^2}} = \frac{1}{\sqrt{2\delta-\delta^2}} \leq \frac{1}{\sqrt{\delta}}. \tag{96}$$

Thus, setting

$$n = \frac{1}{2\delta} \log\left(\frac{1}{\varepsilon\sqrt{\delta}}\right) \geq \frac{1}{2\delta} \log\left(\frac{M}{\varepsilon}\right), \tag{97}$$

it immediately follows that $A_n(x)$ is an ε-approximation of $A(x)$ for all $x \in \mathcal{D}$. All that remains is to show the existence of a QSP protocol yielding $A_n(x)$ as the off-diagonal polynomial Q. Clearly, $A_n(x)$ is even, so both criterion outlined in Thm. D.3 must be demonstrated to hold. To begin, note (again from Lem. G.4) that because each term in the sum of $A_n(x)$ is positive for any $x \in [-1,1]$, we have $A_{n+1}(x) \geq A_n(x)$ for all n and all $x \in [-1,1]$. Since $\lim_{n\to\infty} A_n(x) = A(x)$, it follows that $A_n(x) \leq A(x)$ for all n, which means that for $x \in [-1,1]$,

$$\sqrt{1-x^2}|A_n(x)| = \sqrt{1-x^2}A_n(x) \leq \sqrt{1-x^2}\frac{1}{\sqrt{1-x^2}} = 1, \tag{98}$$

so the first condition of Thm. D.3 holds. As for the second condition, note that

$$A_n(ix) = \sum_{k=0}^{n-1} \binom{-\frac{1}{2}}{k} (-1)^k (ix)^{2k} = \sum_{k=0}^{n-1} \binom{-\frac{1}{2}}{k} x^{2k}. \tag{99}$$

Let $a_n(x) = \sum_{k=0}^{n-1} \binom{-\frac{1}{2}}{k} x^k$: the $(n-1)$-th order Taylor series of $a(x) = \frac{1}{\sqrt{1+x}}$. Since $a(x) = 2\frac{d}{dx}\sqrt{1+x}$, it follows immediately from Lem. G.5 that $a^{(n)}(x) \neq 0$ for all $x \in (0,\infty)$. Moreover, from Lem. G.4, the n-th coefficient in the Taylor series for $a(x)$ has sign $(-1)^n$. Thus, from Thm. G.2, when n is odd, we have $a_n(x) > a(x)$ for all $x \in (0,\infty)$. It immediately follows that $A_n(ix) = a_n(x^2) > a(x^2) = \frac{1}{\sqrt{1+x^2}}$ for all $x \in \mathbb{R} - \{0\}$, with equality at $x = 0$. Therefore,

$$(1+x^2)A_n^*(ix)A_n(x) = (1+x^2)A_n(ix)^2 \geq (1+x^2)\frac{1}{1+x^2} = 1 \tag{100}$$

for all $x \in \mathbb{R}$, when n is odd. The second condition of Thm. D.3 is satisfied. This leads to the following result.

Lemma E.1 (Existence of an extraction QSP polynomial)**.** Given $0 < \varepsilon, \delta \leq 1$, there exists a QSP protocol Φ of length $\delta^{-1}\log(\varepsilon^{-1}\delta^{-1/2})$ such that Q, the off-diagonal QSP polynomial, satisfies $|Q(x) - A(x)| \leq \varepsilon$ for all $x \in [-1+\delta, 1-\delta]$. Moreover, the coefficients of $Q(x)$ can be specified analytically and are highly efficient to compute. □

E.2 The extraction protocol

Using a QSP protocol associated with the polynomial defined in the previous section, it is now possible extract the desired σ_z-rotation. First, a brief lemma.

Lemma E.2. Let U be an element of SU(2). Let $a = \langle 0|U|0\rangle$, $b = \langle 0|U|1\rangle$. If $1 - |a| \leq \varepsilon$, then

$$||U - \mathrm{diag}(U)|| = \left|\left|\begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} - \begin{pmatrix} a & 0 \\ 0 & a^* \end{pmatrix}\right|\right| \leq \sqrt{2\varepsilon} \tag{101}$$

where $||\cdot||$ is the matrix spectral norm. Similarly, if $1-|b| \leq \varepsilon$, then

$$\left|\left|\begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} - \begin{pmatrix} 0 & b \\ -b^* & 0 \end{pmatrix}\right|\right| \leq \sqrt{2\varepsilon} \tag{102}$$

□

Proof. Note that for $U \in \mathrm{SU}(2)$, we have $\det(U) = |a|^2+|b|^2 = 1$. Thus, $|b|^2 = 1-|a|^2 = (1+|a|)(1-|a|) \leq 2(1-|a|)$, implying that $|b| \leq \sqrt{2(1-|a|)} \leq \sqrt{2\varepsilon}$. Thus,

$$\left|\left|\begin{pmatrix} a & b \\ -b^* & a^* \end{pmatrix} - \begin{pmatrix} a & 0 \\ 0 & a^* \end{pmatrix}\right|\right| = \left|\left|\begin{pmatrix} 0 & b \\ -b^* & 0 \end{pmatrix}\right|\right| = |b| \leq \sqrt{2\varepsilon} \tag{103}$$

The proof for the case of $|b|$ being close to 1 is essentially identical. □

From here, we can prove the main theorem of this section.

Theorem E.1 (Existence of an extraction superoperator)**.** Given a twisted embeddable unitary oracle of the form $U = e^{i\varphi\sigma_z/2}e^{i\arccos(x)\sigma_x}e^{-i\varphi\sigma_z/2} = e^{i\varphi\sigma_z/2}W(x)e^{-i\varphi\sigma_z/2}$ and $0 < \varepsilon, \delta \leq 1$, there exists a single-qubit circuit superoperator $\mathcal{E}$ calling U and single-qubit σ_z-rotations $\delta^{-1}\log(8\varepsilon^{-2}\delta^{-1/2}) = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/2}))$ times each, such that when $x \in [-1+\delta, 1-\delta]$, $||\mathcal{E}[U] - e^{-i\varphi\sigma_z}|| \leq \varepsilon$. □

Proof. Define the superoperator $\mathcal{E}$ to be

$$\mathcal{E}[U] = -i\sigma_x\Phi[U] \tag{104}$$

where Φ is a set of $\delta^{-1}\log(8\varepsilon^{-2}\delta^{-1/2})$ QSP phase angles which induces the real polynomial $Q(x)$ such that $\left|Q(x) - \frac{1}{\sqrt{1-x^2}}\right| \leq \frac{\varepsilon^2}{8}$ for all $x \in [-1+\delta, 1-\delta]$ (existence of Φ is guaranteed from Lem. E.1). It then follows that

$$|\sqrt{1-x^2}Q(x) - 1| = \sqrt{1-x^2}\left|Q(x) - \frac{1}{\sqrt{1-x^2}}\right| \leq \frac{\varepsilon^2}{8} \tag{105}$$

for all $x \in [-1+\delta, 1-\delta]$. We then note that

$$\begin{aligned}
\left|\left|\mathcal{E}[U] - e^{-i\varphi\sigma_z}\right|\right| &= \left|\left|-i\sigma_x\Phi[U] - e^{-i\varphi\sigma_z}\right|\right| \\
&= \left|\left|-i\sigma_x e^{i\varphi\sigma_z/2}\Phi[W(x)]e^{-i\varphi\sigma_z/2} - e^{-i\varphi\sigma_z}\right|\right| \\
&= \left|\left|-ie^{-i\varphi\sigma_z/2}\sigma_x\Phi[W(x)]e^{-i\varphi\sigma_z/2} - e^{-i\varphi\sigma_z}\right|\right| \\
&= \left|\left|e^{-i\varphi\sigma_z/2}\left(-i\sigma_x\Phi[W(x)] - \mathbb{I}\right)e^{-i\varphi\sigma_z/2}\right|\right| \\
&= ||-i\sigma_x\Phi[W(x)] - \mathbb{I}|| \\
&= \left|\left|\begin{pmatrix} \sqrt{1-x^2}Q(x) & -iP^*(x) \\ -iP(x) & \sqrt{1-x^2}Q(x) \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right|\right|
\end{aligned} \tag{106}$$

Then, from Lem. E.2 and the triangle inequality,

$$\begin{aligned}
\left|\left|\begin{pmatrix} \sqrt{1-x^2}Q(x) & -iP^*(x) \\ -iP(x) & \sqrt{1-x^2}Q(x) \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right|\right| &\leq \left|\left|\begin{pmatrix} \sqrt{1-x^2}Q(x) & 0 \\ 0 & \sqrt{1-x^2}Q(x) \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right|\right| \\
&\quad + \left|\left|\begin{pmatrix} \sqrt{1-x^2}Q(x) & -iP^*(x) \\ -iP(x) & \sqrt{1-x^2}Q(x) \end{pmatrix} - \begin{pmatrix} \sqrt{1-x^2}Q(x) & 0 \\ 0 & \sqrt{1-x^2}Q(x) \end{pmatrix}\right|\right| \\
&\leq |\sqrt{1-x^2}Q(x) - 1| + \frac{\varepsilon}{2} \leq \frac{\varepsilon^2}{8} + \frac{\varepsilon}{2} < \varepsilon
\end{aligned} \tag{107}$$

for all $x \in [-1+\delta, 1-\delta]$. This completes the proof. □

F Roots of unknown σ_z-rotations

In the following sections, we discuss various techniques for performing oblivious roots of σ_z-rotations, with assumptions of various different constraints and access models.

F.1 Constructing a controlled σ_z-rotation

We begin by describing a technique which allows for the construction of a controlled-σ_z rotation, given oracle access to a σ_z-rotation. To begin, we provide a similar construction to Thm. E.1, which can take twisted embeddable unitaries to the identity.

Lemma F.1 (Existence of a nullification superoperator)**.** Given a twisted embeddable unitary oracle of the form $U = e^{i\varphi\sigma_z/2}e^{i\arccos(x)\sigma_x}e^{-i\varphi\sigma_z/2} = e^{i\varphi\sigma_z/2}W(x)e^{-i\varphi\sigma_z/2}$ and $0 < \varepsilon \leq 1/2$, $0 < \delta \leq 1$, there exists a single-qubit circuit superoperator $\mathcal{E}'$ calling U and single-qubit σ_z-rotations $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}))$ times, such that when $x \in [\delta, 1-\delta]$, $||\mathcal{E}[U] - \mathbb{I}|| \leq \varepsilon$. □

Proof. Define the superoperator $\mathcal{E}'$ to be $\mathcal{E}'[U] = \Phi'[U]$, where Φ' is the set of $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}))$ phase angles achieving a $(\varepsilon^2/16)$-approximating polynomial of the step function $\text{sign}(x)$ on the domain $[-1,-\delta]\cup[\delta,1]$ as the real part of the QSP polynomial $P(x)$ (Lem. G.2). Since such a polynomial approximation is an odd parity function, the sequence Φ' will be of even length. For $x \in [\delta, 1]$, we have

$$||\mathcal{E}'[U] - \mathbb{I}|| = ||\Phi'[U] - \mathbb{I}|| = ||e^{i\varphi\sigma_z/2}\Phi'[W(x)]e^{-i\varphi\sigma_z/2} - \mathbb{I}|| = ||\Phi'[W(x)] - \mathbb{I}|| \tag{108}$$

$$= \left|\left|\begin{pmatrix} P(x) & i\sqrt{1-x^2}Q(x) \\ i\sqrt{1-x^2}Q^*(x) & P^*(x)\end{pmatrix} - \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right|\right| \tag{109}$$

Note that $|\text{Re}[P](x) - \text{sign}(x)| = |\text{Re}[P](x) - 1| \leq \varepsilon^2/16$. This implies that

$$1 - |P(x)| \leq 1 - |\text{Re}[P](x)| = |1 - \text{Re}[P](x)| \leq \frac{\varepsilon^2}{16}. \tag{110}$$

Then, from Lem. E.2 and the triangle inequality,

$$\begin{aligned}\left|\left|\begin{pmatrix} P(x) & i\sqrt{1-x^2}Q(x) \\ i\sqrt{1-x^2}Q^*(x) & P^*(x)\end{pmatrix} - \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right|\right| &\leq \left|\left|\begin{pmatrix} P(x) & 0 \\ 0 & P^*(x)\end{pmatrix} - \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right|\right| \\ &+ \left|\left|\begin{pmatrix} P(x) & i\sqrt{1-x^2}Q(x) \\ i\sqrt{1-x^2}Q^*(x) & P^*(x)\end{pmatrix} - \begin{pmatrix} P(x) & 0 \\ 0 & P^*(x)\end{pmatrix}\right|\right| \\ &\leq |P(x) - 1| + \frac{\varepsilon}{2} = \frac{\varepsilon}{2} + \sqrt{|1 - \text{Re}[P](x)|^2 + |\text{Im}[P](x)|^2}. \end{aligned} \tag{111}$$

Note that $|\text{Im}[P](x)|^2 \leq 1 - |\text{Re}[P](x)|^2 \leq 2(1 - |\text{Re}[P](x)|)$. Therefore,

$$\frac{\varepsilon}{2} + \frac{\varepsilon}{2} + \sqrt{|1 - \text{Re}[P]|^2 + |\text{Im}[P](x)|^2} \leq \sqrt{3(1 - |\text{Re}[P](x)|)} \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \tag{112}$$

This completes the proof. □

From here, we construct a *controlled* variant of the desired σ_z-rotation oracle.

Theorem F.1 (Construction of a controlled-σ_z)**.** Let U be a twisted embeddable unitary such that $U = e^{i\varphi\sigma_z/2}e^{i\arccos(x)\sigma_x}e^{-i\varphi\sigma_z/2}$ with $x \in [\delta, 1-\delta]$. Then there exists a protocol using two ancilla qubits and $\zeta \in \widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$ queries to U, as well as CSWAP gates, which achieves a ε-approximation of a controlled-$e^{-i\varphi\sigma_z}$ gate. □

Proof. Define

$$V = \text{CSWAP}(\mathbb{I}_c \otimes \mathbb{I}_{t_1} \otimes [U]_{t_2})\text{CSWAP}, \tag{113}$$

where we have labeled the control qubit with c and the target qubits with t_1 and t_2. We then define the unitary V' as

$$V' = \left([-i\sigma_x e^{i\phi_0\sigma_z}]_{t_1}\left[e^{i\phi_0'\sigma_z}\right]_{t_2}\prod_{k=1}^{\zeta-1} V\left[e^{i\phi_k\sigma_z}\right]_{t_1}\left[e^{i\phi_k'\sigma_z}\right]_{t_2}\right) \tag{114}$$

where $\Phi = (\phi_0, \ldots, \phi_{\zeta-1})$ and $\Phi' = (\phi_0', \ldots, \phi_{\zeta-1}')$ are the length-ζ QSP sequences of protocols $\mathcal{E}$ and $\mathcal{E}'$ respectively. Note that in both cases, the QSP sequences are of even length. In the case that the control qubit is in state $|0\rangle$, we have $V|0\rangle_c|\psi\rangle_{t_1}|\xi\rangle_{t_2} = |0\rangle_c|\psi\rangle_{t_1}U|\xi\rangle_{t_2}$. In addition, $V|1\rangle_c|\psi\rangle_{t_1}|\xi\rangle_{t_2} = |1\rangle_c U|\psi\rangle_{t_1}|\xi\rangle_{t_2}$. It follows that

$$V'|0\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} = -i|0\rangle\sigma_x e^{i\phi\sigma_z}|\psi\rangle_{t_1}\mathcal{E}'[U]|\xi\rangle_{t_2}, \tag{115}$$

$$V'|1\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} = |1\rangle\mathcal{E}[U]|\psi\rangle_{t_1}e^{i\phi'\sigma_z}|\xi\rangle_{t_2}. \tag{116}$$

where $\phi = \phi_0 + \cdots + \phi_{\zeta-1}$ and $\phi' = \phi'_0 + \cdots + \phi'_{\zeta-1}$. Finally, define $V'' = WV'$, where

$$W = |0\rangle\langle 0|_c \otimes [ie^{-i\phi\sigma_z}\sigma_x]_{t_1} + |1\rangle\langle 1|_c \otimes [e^{-i\phi'\sigma_z}]_{t_2}. \tag{117}$$

Therefore,

$$V''|0\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} = |0\rangle|\psi\rangle_{t_1}\mathcal{E}'[U]|\xi\rangle_{t_2}, \tag{118}$$
$$V''|1\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} = |1\rangle\mathcal{E}[U]|\psi\rangle_{t_1}|\xi\rangle_{t_2}. \tag{119}$$

Our claim is that the gate V'' will be the desired approximate controlled-σ_z rotation. Let $\mathcal{C}$ be the unitary superoperator sending some unitary W to its controlled counterpart, $\mathcal{C}[W]$. We take $\mathcal{C}[e^{-i\varphi\sigma_z}]$ as having c as its control qubit, and t_1 as its target. Note that

$$||\mathcal{C}[e^{-i\varphi\sigma_z}]|0\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} - V''|0\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2}|| = |||\xi\rangle_{t_2} - \mathcal{E}'[U]|\xi\rangle_{t_2}|| \leq ||\mathcal{E}'[U] - \mathbb{I}|| \leq \varepsilon. \tag{120}$$

In addition,

$$||\mathcal{C}[e^{-i\varphi\sigma_z}]|1\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2} - V''|1\rangle|\psi\rangle_{t_1}|\xi\rangle_{t_2}|| = ||e^{-i\varphi\sigma_z}|\psi\rangle_{t_1} - \mathcal{E}[U]|\psi\rangle_{t_1}|| \leq ||\mathcal{E}[U] - e^{-i\varphi\sigma_z}|| \leq \varepsilon. \tag{121}$$

since we have bounded the norms on a basis, it follows that the matrix spectral norm $||\mathcal{C}[e^{-i\varphi\sigma_z}] - V''||$ is too upper-bounded by ε, and we have the desired controlled approximate σ_z-rotation. □

F.2 Taking roots with ancilla qubits

Now, that we have outlined techniques for realizing a controlled-σ_z rotation, let U be an N-dimensional unitary, which can be written in its eigenbasis as

$$U = \sum_{j=1}^{N} e^{i\lambda_i}|\lambda_i\rangle\langle\lambda_i|. \tag{122}$$

Let $|\lambda_j\rangle$ be a particular eigenstate of U, and let V_j be a unitary such that $V_j|0\rangle = |\lambda_j\rangle$. In this case,

$$(X \otimes V_j^\dagger)\mathcal{C}[U](X \otimes V_j) = |1\rangle\langle 1| \otimes \mathbb{I} + |0\rangle\langle 0| \otimes V_j^\dagger U V_j \tag{123}$$
$$= \sum_{j=1}^{N} |1\rangle\langle 1| \otimes V_j^\dagger|\lambda_i\rangle\langle\lambda_i|V_j + e^{i\lambda_i}|0\rangle\langle 0| \otimes V_j^\dagger|\lambda_i\rangle\langle\lambda_i|V_j \tag{124}$$
$$= \sum_{j=1}^{N} e^{i\lambda_i/2}\begin{pmatrix} e^{i\lambda_i/2} & 0 \\ 0 & e^{-i\lambda_i/2}\end{pmatrix} \otimes V_j^\dagger|\lambda_i\rangle\langle\lambda_i|V_j. \tag{125}$$

It follows immediately that

$$(X \otimes V_j^\dagger)\mathcal{C}[U](X \otimes V_j)|\psi\rangle|0\rangle = \left[\sum_{j=1}^{N} e^{i\lambda_i/2}\begin{pmatrix} e^{i\lambda_i/2} & 0 \\ 0 & e^{-i\lambda_i/2}\end{pmatrix} \otimes V_j^\dagger|\lambda_i\rangle\langle\lambda_i|V_j\right]|\psi\rangle|0\rangle \tag{126}$$
$$= e^{i\lambda_j/2}e^{i\sigma_z\lambda_j/2}|\psi\rangle V_j^\dagger|\lambda_j\rangle = e^{i\lambda_j/2}e^{i\sigma_z\lambda_j/2}|\psi\rangle|0\rangle. \tag{127}$$

Therefore, up to an overall phase, the protocol of conjugating controlled-U with $X \otimes V_j$ yields a circuit which applies the principal root of $e^{i\sigma_z\lambda_i}$ to any $|\psi\rangle$. Repetition of this protocol, with the newly-constructed σ_z-rotation in the place of U, can yield any 2^n-th root.

Remark F.1 (The power of controlled oracle access)**.** Having access to an ancilla qubit and a controlled oracle yields a strictly more powerful access model than in the single-qubit setting. Via the mechanism presented above, it is sensible to argue that in the case when the eigenstates of a unitary are known, this power is closely related to the ability to easily take roots of the unitary eigenvalues. It is precisely this access model which allows results like those of [52, 75, 77] to drop the parity constraints associated with QSP protocols. Given access to half-powers of the unitary eigenvalues, $e^{it/2}$, polynomials with respect to $e^{it/2}$ of even parity are precisely those in the variable e^{it} with no definite parity. As will be demonstrated in App. F.3, it is possible to access the roots of unitary eigenvalues up to a desired precision without controlled access, *but this is only possible when the eigenvalues lie in a known half of the complex unit circle.* Therefore, at an even finer-grained level, the power of the controlled access model in this context can be solely related to determination of which half of the unit circle a particular phase lies within. □

Theorem F.2. Given a unitary U of dimension N, with eigenvalues $e^{i\lambda_i}$ and eigenvectors $|\lambda_i\rangle$, where $|\lambda_j\rangle$ for some j is known and preparable on a quantum computer via unitary V_j, and $n \in \mathbb{Z}^+$, there exists a quantum circuit using $n(N+1) = \mathcal{O}(nN)$ ancilla qubits, a single application of U, two applications of V_j and $V_j^\dagger$, and $2n = \mathcal{O}(n)$ controlled-SWAP gates which deterministically sends the state $|0\rangle|\psi\rangle$ to $|0\rangle e^{i\sigma_z \lambda_j/2^n}|\psi\rangle$, for any $|\psi\rangle$, up to a global phase. □

This theorem is an immediate consequence of the procedure described above. The next result follows as a corollary.

Corollary F.2.1. Given access to a single-qubit unitary U, such that $||U - e^{i\sigma_z\varphi}|| \leq \varepsilon$ for $\varphi \in [-\pi, \pi]$, there exists a quantum circuit utilizing two ancilla qubits and a single call to U which yields a gate $\widetilde{U}$ such that $||\widetilde{U}|\psi\rangle - e^{i\varphi/2}e^{i\sigma_z\varphi/2}|\psi\rangle|| \leq \varepsilon$ for every $|\psi\rangle$, with success probability at least $1 - 2\varepsilon$ □

Proof. We use the protocol of Thm. F.2 in the case of $n = 1$. In this case, eigenvalue $e^{i\varphi}$ of $e^{i\varphi\sigma_z}$ corresponds to eigenvector $|0\rangle$, so $V_j = \mathbb{I}$. Note that $\mathcal{C}[e^{i\sigma_z\varphi}]|0\rangle|0\rangle|\psi\rangle = |0\rangle|0\rangle e^{i\varphi/2}e^{i\sigma_z\varphi/2}|\psi\rangle$. We then have

$$\begin{aligned}\Big|\Big|\mathcal{C}[U]|0\rangle|0\rangle|\psi\rangle - |0\rangle|0\rangle e^{i\varphi/2}e^{i\sigma_z\varphi/2}|\psi\rangle\Big|\Big| &= \big|\mathcal{C}[U]|0\rangle|0\rangle|\psi\rangle - \mathcal{C}[e^{i\varphi\sigma_z}]|0\rangle|0\rangle|\psi\rangle\big| \\ &\leq \big|\big|\mathcal{C}[U] - \mathcal{C}[e^{i\varphi\sigma_z}]\big|\big| \\ &= \big|\big|\mathrm{CSWAP}(\mathbb{I}\otimes\mathbb{I}\otimes U - \mathbb{I}\otimes\mathbb{I}\otimes e^{i\varphi\sigma_z})\mathrm{CSWAP}\big|\big| \\ &= ||U - e^{i\sigma_z\varphi}|| \leq \varepsilon, \end{aligned} \tag{128}$$

This implies that $\mathcal{C}[U]|0\rangle|0\rangle|\psi\rangle = |0\rangle|0\rangle e^{i\varphi/2}e^{i\sigma_x\varphi/2}|\psi\rangle + |0\rangle|0\rangle|\psi'\rangle + |\psi''\rangle$, where $\Pi_0|\psi''\rangle = 0$ (the projection onto $|0\rangle|0\rangle$) and $|0\rangle|0\rangle|\psi'\rangle + |\psi''\rangle$ has norm bounded by ε. Measuring the ancilla qubits in the computational basis yields (up to normalization) the state $e^{i\varphi/2}e^{i\sigma_x\varphi/2}|\psi\rangle + |\psi'\rangle$ where $|||\psi'\rangle|| \leq \varepsilon$, with probability $||e^{i\varphi/2}e^{i\sigma_x\varphi/2}|\psi\rangle + |\psi'\rangle||^2 \geq 1 + \varepsilon^2 - 2\Re[\langle\psi|e^{i\varphi/2}e^{i\sigma_x\varphi/2}|\psi'\rangle] \geq 1 - 2\varepsilon$ (the last inequality follows from Cauchy-Schwarz). Upon this measurement outcome, the ancilla qubits are left in the state $|0\rangle|0\rangle$, so they may be utilized for the same subroutine at subsequent steps of a circuit. □

There are multiple subtleties associated with this technique. We conclude this section by remarking on each of them, demonstrating that none hinder the utility or effectiveness of the protocol.

Remark F.2 (Success probability and gadget composition)**.** The fact that the success probability of the above protocol is of the same order as the approximation error of the overall sequence implies that the non-unit success probability is not really a significant constraint in protocols which involve iterative nesting of corrected gadgets.

Suppose we compose m gadgets, making use of m individual correction protocols. Suppose each gadget achieves approximation error of $\varepsilon_1, \ldots, \varepsilon_m$, so that the overall error is on the order $\varepsilon_1 + \cdots + \varepsilon_m = \varepsilon$. Then the success probability p satisfies

$$p \geq (1 - 2\varepsilon_1)\cdots(1 - 2\varepsilon_m) \geq 1 - 2(\varepsilon_1 + \cdots + \varepsilon_m) = 1 - 2\varepsilon \tag{129}$$

(this is easy to show via induction). Thus, the success probability is automatically made high be requiring approximation error to be low, which will be necessary *a priori* to make proper use of gadgets for accurate function approximation. □

Remark F.3 (Ancilla reuse)**.** Since successful application of an approximation of the desired half-rotation corresponds to measurement of $|0\rangle|0\rangle$ in the ancilla, which is also the required input into the above protocol, it follows that the ancilla register can be reused throughout successive applications of "corrected" gadgets, at the same recursion level/depth in a gadget network. Thus, the number of ancilla qubits utilized in a gadget network scales not with the number of individual correction operators utilized, but rather the *depth* of the network, which is usually short, for practical purposes. □

Remark F.4 (Overall phase)**.** This protocol induces an overall phase. This can, generally speaking, be an issue if this protocol is utilized in a "lifted" (QSVT) context, where it is effectuated in individual singular subspaces of a larger block-encoded operator, as it will yield different relative phases between the subspaces. However, in the case of the corrective protocol of Thm. 2.1, we only *conjugate* unitaries by this root protocol, and the induced phases cancel. Thus, this subroutine can be utilized for the correction procedure, in a lifted/QSVT context, without incurring relative phases between subspaces. □

Remark F.5 (Gadget nesting and multi-controlled oracles)**.** In order to recursively perform interlinks of gadgets beyond depth two, it will be necessary to implement controlled-σ_z operations while utilizing black-box calls to circuits which themselves make use of controlled-σ_z operations. This composition implies that depth d interlinks will, in the most general case, require the implementation of $(d-1)$-controlled-σ_z. This requirement is in reality simple to fulfill once given access to single-controlled-σ_z operations: it is possible to implement an arbitrarily controlled gate (say, with d controls) via the conjugation of the single-controlled gate with Toffoli gates acting on $\mathcal{O}(d)$ ancilla qubits (see Fig. 4.10 of Nielsen and Chuang, Ref. [53]; for the complexity of such operations, we refer the reader to foundational [4] and recent [15] works on compiling and controllizing special subsets of quantum gates with and without the use of auxiliary space). □

F.3 Taking roots without ancilla qubits

As it turns out, in certain regimes, it is possible to take the square root of an unknown σ_z-rotation, $e^{i\varphi\sigma_z}$, using *no extra qubits*. The downside of this protocol is that it is only valid when the range of φ is restricted to the subset $[-\pi/2, \pi/2]$ of $[-\pi, \pi]$. In general, there are multiple strategies for performing such a transformation, all of which involve a σ_z-QSP protocol (introduced in Appx. D.1) of the unknown σ_z-rotation. Recall the half-angle formulas of elementary trigonometry, namely

$$\cos\left(\frac{t}{2}\right) = \sqrt{\frac{1+\cos(t)}{2}} \quad \text{and} \quad \sin\left(\frac{t}{2}\right) = \frac{\sin(t)}{\sqrt{2}} \frac{1}{\sqrt{1+\cos(t)}} \tag{130}$$

for all $t \in [-\pi, \pi]$. This implies immediately that the functions $f_{2^n}(e^{it})$ (i.e., the function which when applied to e^{it} produces [an approximation] to $e^{it/2^n}$) can be written in terms of $\cos(t)$ and $\sin(t)$ in a systematic way, via repeated application of these identities.

The strategy developed in this section will be the computation of polynomial approximations of these functions, then utilizing the tools developed in Sec. D.1 to map these polynomials to the σ_z-picture, where they will form a Laurent polynomial approximation of $F_n = f_{2^n}$. $G_n(z) = F_n(z^2)$ will then be an even-parity approximation of the square root function F_{n-1}, when z is restricted to half of the unit circle.

Lemma F.2. Let $f(x) = \sqrt{(1+x)/2}$, and let $B_n(x) = f^{\circ n}(x)$ (the n-fold self composition). Then, for $e^{it} \in \mathbb{T}$,

$$f_{2^n}(e^{it}) = B_n(\cos(t)) + i\sin(t) 2^n B_n^{(1)}(\cos(t)). \tag{131}$$

□

Proof. It is clear from Eq. (130) that $B_n(\cos(t)) = \cos(t/2^n)$. Thus, computing the first derivative (notated here by $B^{(1)}$, to match the following lemma) yields

$$B_n^{(1)}(\cos(t))\sin(t) = \frac{1}{2^n}\sin\left(\frac{t}{2^n}\right), \tag{132}$$

which implies that

$$B_n(\cos(t)) + i\sin(t)2^n B_n^{(1)}(\cos(t)) = \cos\left(\frac{t}{2^n}\right) + i\sin\left(\frac{t}{2^n}\right) = f(e^{it}). \tag{133}$$

This completes the proof. □

Going forward, let $C_n(x) = 2^n B_n^{(1)}(x)$.

Lemma F.3. The derivatives of $B_n(x)$ and $C_n(x)$ are non-zero for $x \in (-1, \infty)$, with $\operatorname{sign}(B_n^{(k)}(x)) = (-1)^{k+1}$ for all $k \geq 1$ and all $x \in (-1, 1]$. In addition, $\operatorname{sign}(C_n^{(k)}(x)) = (-1)^k$ for all $k \geq 1$. Here the superscript $B^{(k)}, C^{(k)}$ indicate k-th derivatives with respect to x. □

Proof. We proceed via induction. Recall that we have $B_1(x) = f(x) = \sqrt{(1+x)/2}$. Lem. G.5 immediately shows that $\operatorname{sign}(B_1^{(k)}(x)) = (-1)^{k+1}$ for $x \in (-1, \infty)$. We assume the case of $B_n(x)$ and consider $B_{n+1}(x)$.

Recall Faà di Bruno's formula, which generalizes the chain rule to arbitrary derivatives:

$$\frac{d^k}{dx^k} f(g(x)) = \sum_M C_M f^{(m_1+\cdots+m_k)}(g(x)) \prod_{j=1}^{k} \left(g^{(j)}(x)\right)^{m_j}, \tag{134}$$

where $C_M > 0$ and we sum over all k-tuples $M = (m_1, \ldots, m_k)$ such that $\sum_{i=1}^{k} i m_i = k$. It follows that for some $k \geq 1$,

$$B_{n+1}^{(k)}(x) = \frac{d^k}{dx^k} f(B_n(x)) = \sum_M C_M f^{(m_1+\cdots+m_k)}(B_n(x)) \prod_{j=1}^{k} \left(B_n^{(j)}(x)\right)^{m_j} \coloneqq \sum_M C_M s_M(x). \tag{135}$$

Consider a single term in the above sum, of the form

$$s_M(x) = f^{(m_1+\cdots+m_k)}(B_n(x)) \prod_{j=1}^{k} \left(B_n^{(j)}(x)\right)^{m_j}. \tag{136}$$

Clearly, it holds that

$$\begin{aligned}
\operatorname{sign}(s_M(x)) &= \operatorname{sign}\left(f^{(m_1+\cdots+m_k)}(b(x))\right) \prod_{j=1}^{k} \left(\operatorname{sign}\left(B_n^{(j)}(x)\right)\right)^{m_j} && (137)\\
&= (-1)^{1+\sum_{i=1}^{k} m_i} \prod_{j=1}^{m} (-1)^{(j+1)m_j} && (138)\\
&= (-1)(-1)^{2\sum_{i=1}^{k} m_i} (-1)^{\sum_{i=1}^{k} i m_i} && (139)\\
&= (-1)^{k+1}, && (140)
\end{aligned}$$

where we make use of the inductive hypothesis, and our knowledge of the signs of the derivatives of $f(x)$. This in fact holds for all terms $s_M(x)$. Thus, $\operatorname{sign}(B_{n+1}^{(k)}(x)) = (-1)^{k+1}$ for all $x \in (-1, \infty)$ and $k \geq 1$. Since $C_n(x) = 2^n B_n^{(1)}(x)$, it immediately follows that $\operatorname{sign}(C_n^{(k)}(x)) = (-1)^k$ for $k \geq 1$, and x in the same domain. □

Let $B_{(n,m)}(x)$ and $C_{(n,m)}(x)$ denote the $(m-1)$-th order Taylor approximations of $B_n(x)$ and $C_n(x)$, respectively, centred at $x = 1$.

Lemma F.4 (Existence of approximating polynomials)**.** Let $0 < \varepsilon, \delta \leq 1$. There exist polynomials P_{B_n} and P_{C_n} of degree (either even or odd) $d_{B_n}, d_{C_n} \leq r$ with $r = \mathcal{O}\left(\delta^{-1} \log(\varepsilon^{-1}\delta^{-1/2})\right)$ such that for all $x \in [-1+\delta, 3-\delta]$, $|P_{B_n}(x) - B(x)| \leq \varepsilon$ and $|P_{C_n}(x) - C(x)| \leq \varepsilon$. In addition,

$$||P_B(x)||_{[-1,1]} \leq 2 + \varepsilon \quad \text{and} \quad ||P_C(x)||_{[-1,1]} \leq \frac{2\sqrt{2}}{3\sqrt{\delta}} + \varepsilon \tag{141}$$

where $||\cdot||_S$ denotes the maximum magnitude over the set S. □

Proof. We begin by bounding the absolute sums of the Taylor series of both $B_n(x)$ and $C_n(x)$. From Lem. F.3, the signs of the coefficients in each of the Taylor series will be alternating. In particular,

$$B_{(n,m)}(x+1) = \sum_{k=0}^{m-1} b_{(n,k)} x^k = b_{(n,0)} - \sum_{k=1}^{m-1} (-1)^k |b_{(n,k)}| x^k = b_{(n,0)} - \sum_{k=1}^{m-1} |b_{(n,k)}| (-x)^k, \tag{142}$$

implying that $\sum_{k=0}^{m-1} |b_{(n,k)}|(-x)^k = 2b_{(n,0)} - B_{(n,m)}(x+1)$ for any x. Following similar logic, one can conclude that $\sum_{k=0}^{m-1} |c_{(n,k)}|(-x)^k = C_{(n,m)}(x+1)$. Let $0 < \delta \leq 1$. It follows that

$$\begin{aligned}
\sum_{k=0}^{\infty} |b_{(n,k)}| \left(2 - \frac{\delta}{2}\right)^k = \sum_{k=0}^{\infty} |b_{(n,k)}| \left(-\left(-2 + \frac{\delta}{2}\right)\right)^k &= \lim_{k\to\infty} 2b_{(n,0)} - B_{(n,k)}\left(-1 + \frac{\delta}{2}\right) && (143)\\
&= 2b_{(n,0)} - B_n\left(-1 + \frac{\delta}{2}\right) && (144)
\end{aligned}$$

Clearly, $0 \leq B_n(x)$, so the above sum is upper-bounded by 2. Since the above equality holds for arbitrary $\delta \in [0, 2]$ (and thus $|x| \in [0, 1]$), this also implies that $||B_n(x)|| \leq 2$ over the entire interval $[-1, 1]$. In addition,

$$\sum_{k=0}^{\infty} |c_{(n,k)}| \left(2 - \frac{\delta}{2}\right)^k = \lim_{k\to\infty} C_{(n,k)}\left(-1 + \frac{\delta}{2}\right) = C_n\left(-1 + \frac{\delta}{2}\right) \tag{145}$$

Because $C_n(\cos(t)) = \sin(t)^{-1}\sin(t/2^n)$, it follows that for $x \in [-1,1]$, $|C_n(x)| \leq (1-x^2)^{-1/2}$. Thus,

$$\left|C_n\left(-1+\frac{\delta}{2}\right)\right| \leq \frac{1}{\sqrt{\delta-\delta^2/4}} \leq \frac{2}{\sqrt{3\delta}} \tag{146}$$

This also implies that $||C_n(x)||_{[-1+\delta,1-\delta]} \leq 2/\sqrt{6\delta}$, as $\delta/2$ was an arbitrary choice. Thus, via Cor. G.2.1, there exists a polynomial $P_{B_n}(x)$ of degree $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}))$ which ε-approximates $B_n(x)$ on $[-1+\delta, 3-\delta]$ and satisfies the first condition of Eq. (141), as well as a polynomial $P_{C_n}(x)$ of degree $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}\delta^{-1/2}))$ which ε-approximates $C_n(x)$ on on $[-1+\delta, 3-\delta]$ and satisfies the second condition of Eq. (141). The upper bound r on both of the polynomial degrees will be in $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}\delta^{-1/2}))$ as well. □

Remark F.6. It is possible that r in the above lemma scales with n as well, in a way that is independent of ε and δ: this behaviour is not captured in the theorem utilized to prove the result. □

Lemma F.5 (Existence of normalized approximating polynomials)**.** Let $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$. There exist polynomials $\widetilde{P}_{B_n}(x)$ and $\widetilde{P}_{C_n}(x)$ satisfying all of the conditions of Lem. F.4, along with the normalization condition that

$$|\widetilde{P}_{B_n}(x)|^2 + (1-x^2)|\widetilde{P}_{C_n}(x)|^2 \leq 1 \tag{147}$$

for all $x \in [-1,1]$. □

Proof. Let $P_{B_n}(x)$ and $P_{C_n}(x)$ be $(\varepsilon/4)$-approximating polynomials of $B_n(x)$ and $C_n(x)$ on the interval $[-1+\delta/2, 3-\delta/2]$, satisfying the conditions of Lem. F.4. It follows that $P_{B_n}(x) - \varepsilon/4$ and $P_{C_n}(x) - \varepsilon/4$ will be $\varepsilon/2$-approximations of B_n and C_n such that on $[-1+\delta/2, 3-\delta/2]$ which are less than or equal to B_n and C_n, respectively, on this interval as well. Moreover, since $\varepsilon \leq 1/2$, both $\widetilde{P}_{B_n}(x)$ and $\widetilde{P}_{C_n}(x)$ are positive on $[-1,1]$. Therefore,

$$||P_{B_n}(x) - \varepsilon/4||_{[-1,1]} = ||P_{B_n}(x)||_{[-1,1]} - \frac{\varepsilon}{4} \leq 2, \tag{148}$$

$$||P_{C_n}(x) - \varepsilon/4||_{[-1,1]} = ||P_{C_n}(x)||_{[-1,1]} - \frac{\varepsilon}{4} \leq \frac{4}{3\sqrt{\delta}}. \tag{149}$$

from Eq. 141. From Lem. G.2, there exists a real polynomial $\widetilde{S}(x) = \frac{1+S\left(x+1-\frac{3\delta}{4}\right)}{2}$ of degree $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}))$ which is bounded in magnitude by 1 on $[-2,2]$, and satisfies

$$\widetilde{S}(x) \in \left[0, \frac{3\sqrt{\delta}\varepsilon}{8}\right] \text{ for } x \in \left[-3+\frac{3\delta}{4}, -1+\frac{\delta}{2}\right], \widetilde{S}(x) \in \left[1-\frac{3\sqrt{\delta}\varepsilon}{8}, 1\right] \text{ for } x \in \left[-1+\delta, 1+\frac{3\delta}{4}\right] \tag{150}$$

(this requires replacing $\delta \mapsto \delta/4$ in the lemma). Let $\widetilde{P}_{B_n}(x) = \widetilde{S}(x)(P_{B_n}(x) - \varepsilon/4)$ and $\widetilde{P}_{C_n}(x) = \widetilde{S}(x)(P_{C_n}(x) - \varepsilon/4)$. From here, note that for $x \in [-1+\delta, 1]$, we have

$$\left|B_n(x) - \widetilde{P}_{B_n}(x)\right| = B_n(x) - \widetilde{S}(x)(P_{B_n}(x) - \varepsilon/4) \tag{151}$$

$$= (B_n(x) - (P_{B_n}(x) - \varepsilon/4)) + (P_{B_n}(x) - \varepsilon/4)\left(1 - \widetilde{S}(x)\right) \tag{152}$$

$$\leq \frac{\varepsilon}{2} + |P_{B_n}(x) - \varepsilon/4|\frac{3\sqrt{\delta}\varepsilon}{8} \leq \frac{\varepsilon}{2} + \frac{\sqrt{\delta}\varepsilon}{2} \leq \varepsilon. \tag{153}$$

In addition,

$$\left|C_n(x) - \widetilde{P}_{C_n}(x)\right| = C_n(x) - \widetilde{S}(x)(P_{C_n}(x) - \varepsilon/4) \tag{154}$$

$$= (C_n(x) - (P_{C_n}(x) - \varepsilon/4)) + (P_{C_n}(x) - \varepsilon/4)\left(1 - \widetilde{S}(x)\right) \tag{155}$$

$$\leq \frac{\varepsilon}{2} + |P_{C_n}(x) - \varepsilon/4|\frac{3\sqrt{\delta}\varepsilon}{8} \leq \varepsilon. \tag{156}$$

Now, on the interval $[-1+\delta/2, -1+\delta]$, note that $|\widetilde{P}_{B_n}(x)| \leq |B_n(x)|$ and $|\widetilde{P}_{C_n}(x)| \leq |C_n(x)|$, as $|\widetilde{S}(x)| \leq 1$. Thus, on this interval

$$|\widetilde{P}_{B_n}(x)|^2 + (1-x^2)|\widetilde{P}_{C_n}(x)|^2 \leq |B_n(x)| + (1-x^2)|C_n(x)|^2 = 1, \tag{157}$$

which is easy to verify from the definitions of $B_n(x)$ and $C_n(x)$. Finally, on $[-1, -1+\delta/2] \cup [1-\delta/2, 1]$, note that

$$|\widetilde{P}_{B_n}(x)| \leq |\widetilde{S}(x)||P_{B_n}(x) - \varepsilon/4| \leq \frac{\sqrt{\delta}\varepsilon}{2} \leq \frac{1}{4}, \tag{158}$$

$$|\widetilde{P}_{C_n}(x)| \leq |\widetilde{S}(x)||P_{C_n}(x) - \varepsilon/4| \leq \frac{\varepsilon}{2} \leq \frac{1}{4}. \tag{159}$$

Thus, it is straightforward to check that the normalization condition is satisfied on this domain as well. We conclude that $\widetilde{P}_{B_n}(x)$ and $\widetilde{P}_{C_n}(x)$ are polynomials with degree in $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-1}\delta^{-1/2}))$ which are ε-approximations of $B_n(x)$ and $C_n(x)$ respectively, such that $|\widetilde{P}_{B_n}(x)|^2 + (1-x^2)|\widetilde{P}_{C_n}(x)|^2 \leq 1$ on the entire interval $[-1, 1]$. This completes the proof. □

Equipped with the polynomials of Lem. F.5, we can complete our construction.

Theorem F.3 (Existence of an approximating QSP polynomial)**.** Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists a σ_z-QSP protocol Φ'_n of length $\mathcal{O}(\delta^{-2}\log(\varepsilon^{-2}\delta^{-1/2}))$ such that for all $\varphi \in [-\pi, \pi]$ satisfying $\cos(\varphi) \in [\delta, 1]$, $\Phi'_n[e^{i\varphi\sigma_z}]$ is ε-close to $e^{i\varphi\sigma_z/2^n}$. □

Proof. Recall the bijective map $\Lambda : \mathbb{C}[z, z^{-1}]_m \to \mathbb{C}[x]_m \times \mathbb{C}[x]_{m-1}$ defined in Thm. D.5. Let $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/4$. Let $(\widetilde{P}_{B_n}(x), \widetilde{P}_{C_n}(x))$ be constructed from Lem. F.5, such that they are real and $\varepsilon^2/8$-approximate real polynomials $B_n(x)$ and $C_n(x)$ respectively on the interval $[-1+2\delta^2, 1]$, and satisfy the normalization constraint of Eq. (147). Clearly, $(\widetilde{P}_{B_n}(x), \widetilde{P}_{C_n}(x)) \in S_1 \cap \mathbb{R}[x]_m \times \mathbb{R}[x]_{m-1}$ for some even $m \in \mathcal{O}(\delta^{-2}\log(\varepsilon^{-2}\delta^{-1/2}))$: the subset of real polynomial pairs satisfying the normalization condition of Eq. (147). It follows from Lem. D.2 that the real Laurent polynomial $p = \Lambda^{-1}(\widetilde{P}_{B_n}, \widetilde{P}_{C_n})$ satisfies $|p(z)| \leq 1$ for all $z \in \mathbb{T}$. Let $g(z) = p(z^2)$. Clearly, p will be even, and thus has parity m mod 2. It follows from Thm. D.6 that there exists a length $2m \in \mathcal{O}\left(\delta^{-2}\log(\varepsilon^{-2}\delta^{-1/2})\right)$ σ_z-QSP protocol Φ'_n such that

$$\langle 0|\Phi'_n[e^{i\varphi\sigma_z}]|0\rangle = g(e^{i\varphi}) = p(e^{2i\varphi}) = \widetilde{P}_{B_n}(\cos(2\varphi)) + i\sin(2\varphi)\widetilde{P}_{C_n}(\cos(2\varphi)) \tag{160}$$

where, for $\cos(2\varphi) \in [-1+2\delta^2, 1]$, $|\widetilde{P}_{B_n}(\cos(2\varphi)) - B_n(\cos(2\varphi))| \leq \varepsilon^2/8$ and $|\widetilde{P}_{C_n}(\cos(2\varphi)) - C_n(\cos(2\varphi))| \leq \varepsilon^2/8$. Thus, via triangle inequality,

$$\left|\langle 0|\Phi'_n[e^{i\varphi\sigma_z}]H|0\rangle - B(\cos(2\varphi)) + i\sin(2\varphi)C(\cos(2\varphi))\right| \leq \frac{\varepsilon^2}{4}. \tag{161}$$

For all $2\varphi \in [-\pi, \pi]$, so for all $\varphi \in [-\pi/2, \pi/2]$ ($\cos(\varphi) \in [0, 1]$), we know that $B_n(\cos(2\varphi)) + i\sin(2\varphi)C_n(\cos(2\varphi)) = e^{i\varphi/2^n}$. Note that

$$\cos(2\varphi) = 2\cos^2(\varphi) - 1 \in [-1+2\delta^2, 1] \quad \text{and} \quad \cos(\varphi) \geq 0 \iff \cos(\varphi) \in [\delta, 1]. \tag{162}$$

Therefore, for $\cos(\varphi) \in [\delta, 1]$,

$$\left|\langle 0|\Phi'_n[e^{i\varphi\sigma_z}]|0\rangle - e^{i\varphi/2^n}\right| \leq \frac{\varepsilon^2}{4} \quad \text{and} \quad \left|\langle 1|\Phi'_n[e^{i\varphi\sigma_z}]|1\rangle - e^{-i\varphi/2^n}\right| \leq \frac{\varepsilon^2}{4}. \tag{163}$$

It is easy to verify that $\Phi'_n[e^{i\varphi\sigma_z}]$ is SU(2). Letting $a = \langle 0|\Phi'_n[e^{i\varphi\sigma_z}]|0\rangle$, it is clear that

$$1 - |a| \leq 1 - |a|^2 = e^{-i\varphi/2^n}e^{i\varphi/2^n} - a^*a = e^{-i\varphi/2^n}(e^{i\varphi/2^n} - a) + a(e^{-i\varphi/2^n} - a^*) \tag{164}$$

$$|e^{i\varphi/2^n} - a| + |e^{-i\varphi/2^n} - a^*| \leq \frac{\varepsilon^2}{2} \tag{165}$$

so from Lem. E.2, $||\Phi'_n[e^{i\varphi\sigma_z}] - e^{i\varphi\sigma_z/2^n}|| \leq \varepsilon$, and the proof is complete. □

As a corollary, it is easy to see that the procedure provided above can be utilized to take roots of σ_x-rotations as well (since $e^{i\varphi\sigma_x} = He^{i\varphi\sigma_z}H$). This result will be useful in Sec. 4.

Corollary F.3.1. Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists a QSP protocol Φ_n of length $\zeta = \mathcal{O}(\delta^{-2}\log(\varepsilon^{-2}\delta^{-1/2}))$ such that for all $\varphi \in [-\pi, \pi]$ satisfying $\cos(\varphi) \in [\delta, 1]$, $\Phi_n[e^{i\varphi\sigma_x}]$ is ε-close to $e^{i\varphi\sigma_x/2^n}$. □

G Techniques for polynomial approximation

In this section, we present a toolbox of useful techniques for constructing polynomial approximations, in the context of QSP. The first results showcased are constructive, and are utilized for when defining and proving properties of the extraction and root polynomials of Appx. E and Appx. F.

Theorem G.1 (Polynomial approximation with bounding series)**.** Let $f(x)$ be a function, let $P_n(x) = \sum_{k=0}^{n-1} p_k x^{\lambda k}$ be a family of $(n-1)$-th order approximating polynomials, where $\lambda \in \mathbb{Z}^+$, such that $|\lim_{n\to\infty} P_n(x) - f(x)| = 0$ for each $x \in (-1,1)$ (uniform convergence is not required). Let $0 < \varepsilon, \delta \leq 1$. Suppose $|p_k| \leq q_k$ for a weakly decreasing sequence q_k. Let $g(x) = \sum_{k=0}^{\infty} q_k x^{\lambda k}$. Let $M = |g(1-\delta)|$. Then, if $n \geq (\lambda\delta)^{-1} \log(M\varepsilon^{-1})$, $|P_n(x) - f(x)| \leq \varepsilon$ for all $x \in (-1+\delta, 1-\delta)$. □

Proof. Clearly, for each $x \in (-1+\delta, 1-\delta)$, n can be chosen such that $|P_n(x) - f(x)| \leq \varepsilon$, as $|\lim_{n\to\infty} P_n(x) - f(x)| = 0$ for all $x \in (-1,1)$. For some n, note that

$$|P_n(x) - f(x)| \leq |\lim_{n\to\infty} P_n(x) - P_n(x)| + |\lim_{n\to\infty} P_n(x) - f(x)| = |\lim_{n\to\infty} P_n(x) - P_n(x)| \tag{166}$$

$$= \left|\sum_{k=n}^{\infty} p_k x^{\lambda k}\right| = |x|^{\lambda n} \left|\sum_{k=n}^{\infty} p_k x^{\lambda(k-n)}\right| \leq |x|^{\lambda n} \sum_{k=n}^{\infty} |p_k||x|^{\lambda(k-n)} \leq |x|^{\lambda n} \sum_{k=n}^{\infty} g_k |x|^{\lambda(k-n)} \tag{167}$$

$$\leq |x|^{\lambda n} \sum_{k=n}^{\infty} g_{n-k} |x|^{\lambda(k-n)} = |x|^{\lambda n} \sum_{k=0}^{\infty} g_k |x|^{\lambda x} \leq |x|^{\lambda n} |g(1-\delta)|, \tag{168}$$

where we use the fact that $|p_k| \leq q_k \leq q_{k-n}$. Thus,

$$|P_n(1-\delta) - f(1-\delta)| \leq M|x|^{\lambda n} \leq M(1-\delta)^{\lambda n}. \tag{169}$$

Note that

$$M(1-\delta)^{\lambda n} \leq \varepsilon \Leftrightarrow n \geq \frac{1}{\lambda} \log\left(\frac{M}{\varepsilon}\right) \log\left(\frac{1}{1-\delta}\right)^{-1}. \tag{170}$$

Finally, note that for $\delta \in (0,1]$, we have $\log(1-\delta) \leq -\delta$, so $\log(1/(1-\delta)) = -\log(1-\delta) \geq \delta$. This implies that $\log(1/(1-\delta))^{-1} \leq 1/\delta$. It follows that if we set

$$n \geq \frac{1}{\lambda\delta} \log\left(\frac{M}{\varepsilon}\right) = \mathcal{O}\left(\frac{1}{\delta} \log\left(\frac{M}{\varepsilon}\right)\right), \tag{171}$$

then by Eq. (170) and Eq. (169), the bound will hold. Note that the resulting polynomial will have degree λn. □

This result is fairly general, and yields the following corollaries immediately.

Corollary G.1.1 (Polynomial approximation with weakly decreasing coefficients)**.** Let $f(x)$ be a function, let $P_n(x) = \sum_{k=0}^{n-1} p_k x^{\lambda k}$ be a family of $(n-1)$-th order approximating polynomials, where $\lambda \in \mathbb{Z}^+$, such that $|\lim_{n\to\infty} P_n(x) - f(x)| = 0$ for all $x \in (-1,1)$. Let $0 < \varepsilon, \delta \leq 1$. Suppose $|p_k|$ is weakly decreasing. Moreover, suppose the sign of each p_k is constant, or alternating between positive and negative (and, in this latter case, λ is odd). In the former case, let $N = |f(1-\delta)|$. In the latter, let $N = |f(-1+\delta)|$. Then, if $n \geq (\lambda\delta)^{-1} \log(N\varepsilon^{-1})$, then $|P_n(x) - f(x)| \leq \varepsilon$ for all $x \in (-1+\delta, 1-\delta)$. □

Proof. Let $g_k = |p_k|$. In the case that each p_k has the same sign, $g(x) = \pm\sum_{k=0}^{\infty} p_k x^k = \pm f(x)$, so $M = |f(1-\delta)|$, and we can apply Thm. G.1. In the latter case, $g(x) = \pm\sum_{k=0}^{\infty} (-1)^k p_k x^{\lambda k} = \pm\sum_{k=0}^{\infty} p_k (-x)^{\lambda k}$ (the final equality follows from the fact that λ is odd), so $M = |g(1-\delta)| = |f(-1+\delta)|$, and we once again can apply Thm. G.1. □

Corollary G.1.2 (Polynomial approximation with constant bound)**.** Let $f(x)$ be a function, let $P_n(x) = \sum_{k=0}^{n-1} p_k x^{\lambda k}$ be a family of $(n-1)$-th order approximating polynomials, where $\lambda \in \mathbb{Z}^+$, such that $|\lim_{n\to\infty} P_n(x) - f(x)| = 0$ for all $x \in (-1,1)$. Let $0 < \varepsilon, \delta \leq 1$. Suppose $|p_k| \leq C$. Then, if $n \geq (\lambda\delta)^{-1} \log(C\delta^{-1}\varepsilon^{-1})$, $|P_n(x) - f(x)| \leq \varepsilon$ for all $x \in (-1+\delta, 1-\delta)$. □

Proof. Let $q_k = C$, so $g(x) = C\sum_{i=0}^{\infty} x^i = \frac{C}{1-x}$ for $x \in (-1,1)$. In this case, $M = |g(1-\delta)| = \frac{C}{\delta}$. We can then apply Thm. G.1 and the result follows. □

Theorem G.2. Let $C(x)$ be an analytic function, let $C_n(x) = \sum_{i=0}^{n-1} c_i x^i$ be the $(n-1)$-th order Taylor series of f centered at 0. Suppose that the n-th derivative of $C(x)$ is non-zero for all $x \in [0, d)$ where $d > 0$. Moreover, suppose $\text{sign}(c_n) = (-1)^n$. If n is even, then $C_n(x) < C(x)$ for all $x \in \mathcal{D} = (0, d)$. If n is odd, then $C_n(x) > C(x)$ for all $x \in \mathcal{D}$. □

Proof. Clearly, $C^{(k)}(0) = C_n^{(k)}(0)$ for all k from 0 to $n-1$. Therefore, we have

$$C(x) - C_n(x) = \sum_{k=0}^{n-1} c_i x^i - \sum_{k=0}^{\infty} c_i x^i = \sum_{k=n}^{\infty} c_i x^i = (-1)^n |c_n| x^n + \sum_{k=n+1}^{\infty} c_k x^i, \tag{172}$$

where we know that $|c_n| > 0$. Therefore the term with lowest degree in x of the above sum has a coefficient of non-zero magnitude: positive for even n and negative for odd n. Thus, for an open neighbourhood $U = (0, \gamma)$ with $\gamma > 0$, $\text{sign}(C(x) - C_n(x)) = (-1)^n$, so when n is even, $C(x) > C_n(x)$ and when n is odd, $C_n(x) < C(x)$.

Now, suppose there exists some $r_0 \in \mathcal{D}$ such that $C(r_0) = C_n(r_0)$. We know that $r_0 > 0$ as $\mathcal{D} \subset (0, \infty)$. Recall the single-variable mean value theorem (MVT), which states that for $a, b \in \mathbb{R}$ with $a < b$ and differentiable functions f, g, if $f(a) = g(a)$ and $f(b) = g(b)$, there exists some $c \in (a, b)$ such that $f^{(1)}(c) = g^{(1)}(c)$. Since $C(0) = C_n(0)$, there must exist some $r_1 \in (0, r_0)$ such that $C^{(1)}(r_1) = C_n^{(1)}(r_1)$. Recall that $C^{(k)}(0) = C_n^{(k)}(0)$ for all k from 0 to $n-1$. Thus, we can inductively apply MVT until we arrive at a point $r_n \in \mathcal{D}$ such that the n-th derivatives satisfy $C^{(n)}(r_n) = C_n^{(n)}(r_n) = 0$. But this is a contradiction as $C^{(n)}(x) \neq 0$ for $x \in \mathcal{D}$. It follows that we cannot have $C(x) = C_n(x)$ for $x \in \mathcal{D}$. Since $C_n(x) < C(x)$ in $(0, \gamma)$ for n even, we must then have that $C_n(x) < C(x)$ on the whole interval $\mathcal{D}$ in this case. Similarly, for n odd, we must have $C_n(x) > C(x)$ for $x \in \mathcal{D}$. □

In addition, we restate some of the theorems of [26], which give existence results for Fourier-based approximations of polynomials, with particular properties.

Lemma G.1 (Corollary 66, Ref. [26])**.** Let $x_0 \in [-1, 1]$, $r \in (0, 2]$, $\delta \in (0, r]$, and let $f : [x_0 - r - \delta, x_0 + r + \delta] \to \mathbb{C}$ be such that $f(x_0 + x) = \sum_{\ell=0}^{\infty} a_\ell x^\ell$ for all $x \in [-r - \delta, r + \delta]$. Suppose $B > 0$ is chosen such that $\sum_{\ell=0}^{\infty} (r + \delta)^\ell |a_\ell| \leq B$. Then given $\varepsilon \in (0, \frac{1}{2B}]$, there exists an efficiently-computable polynomial $P \in \mathbb{C}[x]$ of degree $\mathcal{O}\left(\frac{1}{\delta} \log\left(\frac{B}{\varepsilon}\right)\right)$ such that $|f(x) - P(x)| \leq \varepsilon$ for $x \in [x_0 - r, x_0 + r]$, and we have

$$||P(x)||_{[-1,1]} \leq \varepsilon + ||f(x)||_{[x_0 - r - \delta/2, x_0 + r + \delta/2]}. \tag{173}$$

where $||\cdot||_I$ denotes the infinite norm over the interval I. □

This allows us to prove the following corollary.

Corollary G.2.1. Suppose $f : [-1 + \delta/2, 3 - \delta/2] \to \mathbb{C}$ is such that $f(x + x_0) = \sum_{\ell=0}^{\infty} a_\ell x^\ell$ for all $x \in [-2 + \delta/2, 2 - \delta/2]$. Suppose $B > 0$ is chosen such that $\sum_{\ell=0}^{\infty} (2 - \delta/2)^\ell |a_\ell| \leq B$. Then given $\varepsilon \in (0, \frac{1}{2B}]$, there exists an efficiently-computable polynomial $P \in \mathbb{C}[x]$ of degree $\mathcal{O}\left(\frac{1}{\delta} \log\left(\frac{B}{\varepsilon}\right)\right)$ such that $|f(x) - P(x)| \leq \varepsilon$ for $x \in [-1 + \delta, 3 - \delta]$, and we have

$$||P(x)||_{[-1,1]} \leq \varepsilon + ||f(x)||_{[-1 + 3\delta/4, 3 - 3\delta/4]}. \tag{174}$$

□

Proof. We set $x_0 = 1$, $r = 2 - \delta$. We then apply Lem. G.1 (where we send $\delta \mapsto \delta/2$ in the statement of the lemma). □

Remark G.1. In the case that f has a real codomain, then this lemma can be amended to require that $P \in \mathbb{R}[x]$: this simply follows from the fact that if $f(x)$ is real, then

$$|f(x) - P(x)| = \sqrt{(\text{Re}[P](x) - f(x))^2 + \text{Im}[P](x)^2} \geq |\text{Re}[f](x) - P(x)| \tag{175}$$

and $||P(x)||_{\mathcal{D}} \geq ||\text{Re}[P(x)]||_{\mathcal{D}}$ for some subset $\mathcal{D} \subset [-1, 1]$. Thus, the bounds of Lem. G.1 still hold when P is replaced with $\text{Re}[P] \in \mathbb{R}[x]$. □

Lemma G.2 (Polynomial approximation of the sign function, Ref. [26])**.** For each $\delta > 0$, $0 < \varepsilon < 1/2$, there exists an efficiently computable odd polynomial $S(x) \in \mathbb{R}[x]$ of degree in $\mathcal{O}\left(\frac{1}{\delta} \log\left(\frac{1}{\varepsilon}\right)\right)$ such that $|S(x)| \leq 1$ for all $x \in [-2, 2]$ and for all $x \in [-2, -\delta] \cup [\delta, 2]$, $|S(x) - \text{sign}(x)| \leq \varepsilon$. □

Lemma G.3 (Polynomial approximation of rectangle functions, Ref. [26])**.** Let $0 < \varepsilon, \delta < \frac{1}{2}$. Let $t \in [-1, 1]$. There exists an even polynomial $R(x) \in \mathbb{R}[x]$ of degree in $\mathcal{O}\left(\frac{1}{\delta} \log\left(\frac{1}{\varepsilon}\right)\right)$ such that $|R(x)| \leq 1$ for all $x \in [-1, 1]$, and

$$R(x) \in [0, \varepsilon] \quad \text{for } x \in [-1, -t - \delta] \cup [t + \delta, 1], \tag{176}$$

$$R(x) \in [1 - \varepsilon, 1] \quad \text{for } x \in [-t + \delta, t - \delta]. \tag{177}$$

□

Finally, we state and prove miscellaneous lemmas, which are used in repeatedly throughout this work.

Lemma G.4. Given $j \in \mathbb{Z}^+$,

$$\binom{-\frac{1}{2}}{j} = (-1)^j \prod_{k=1}^{j} \left(1 - \frac{1}{2k}\right) \quad \text{and} \quad \binom{\frac{1}{2}}{j} = \frac{(-1)^j}{2j-1} \prod_{k=1}^{j} \left(1 - \frac{1}{2k}\right). \tag{178}$$

Thus, both binomial functions are decreasing in magnitude as a function of j, and have sign $(-1)^j$. □

Proof. For $a, b \in \mathbb{Z}$, note that

$$\binom{\frac{a}{2}}{b} = \frac{\left(\frac{a}{2}\right)!}{b!\left(\frac{a}{2} - b\right)!} = \frac{1}{b!} \prod_{k=1}^{b} \left(\frac{a}{2} + 1 - k\right) = \frac{1}{b!} \left(-\frac{1}{2}\right)^b \prod_{k=1}^{b} (2k - (a+2)) = (-1)^b \prod_{k=1}^{b} \left(1 - \frac{a+2}{2k}\right). \tag{179}$$

Thus, when $a = -1$, $b = j$, we have

$$\binom{-\frac{1}{2}}{j} = (-1)^j \prod_{k=1}^{j} \left(1 - \frac{1}{2k}\right), \tag{180}$$

and when $a = 1$, $b = j$, we have

$$\binom{\frac{1}{2}}{j} = \frac{1}{j!} \left(-\frac{1}{2}\right)^j \prod_{k=1}^{j} (2k - 3) = \frac{1}{2j-1} (-1)^j \prod_{k=1}^{j} \frac{2k-1}{2k} = \frac{(-1)^j}{2j-1} \prod_{k=1}^{j} \left(1 - \frac{1}{2k}\right). \tag{181}$$

□

Lemma G.5. Let $f(x) = \sqrt{1+x}$. Then for $k \geq 1$, $f^{(k)}(x) = \frac{(-1)^{k+1}(2k-3)!!}{2^k} (1+x)^{\frac{1}{2}-k}$ where the double factorial takes the product of all positive integers less than or equal to, and of equal parity to $2k-3$ and $(-1)!! = 1$. Here the superscript $f^{(k)}$ indicates the k-th derivative of f with respect to its argument. □

Proof. This is simple to check via induction. The base case is immediate. We assume the case of k, and note that

$$\frac{d}{dx} \frac{(-1)^{k+1}(2k-3)!!}{2^k} (1+x)^{\frac{1}{2}-k} = \frac{(-1)^{k+2}(2k-1)!!}{2^{k+1}} (1+x)^{-\frac{1}{2}-(k+1)}, \tag{182}$$

which proves the claim. □

H Examples of composite gadgets

In this section, we provide proofs and further explanation of several of the constructions introduced in Sec. 4.

H.1 Proofs of results

We begin with a proof demonstrating the existence of the simple bandpass family.

Example 4.9 (Simple bandpass family)**.** Given $a \in [\delta, 1/\sqrt{2}]$, there exists a family of $(1, 1)$ gadgets $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$, such that $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$ ε-approximately achieves a function f satisfying $|f| \leq 1$ as well as

$$f(x) \in \begin{cases} [1 - \varepsilon, 1] & \text{if } x \in [-a + \delta, a - \delta] \\ [-1, -1 + \varepsilon] & \text{if } x \in \left[-\frac{1}{\sqrt{2}}, -a - \delta\right] \cup \left[a + \delta, \frac{1}{\sqrt{2}}\right], \end{cases} \tag{20}$$

where the total depth of the gadget, ζ', satisfies

$$\zeta' = \mathcal{O}\left((2a\delta)^{-4(\nu_\ell+1)}\log^{1+\nu_\ell}\left(\varepsilon^{-1}\right)\right). \tag{21}$$

Moreover, given the phases of $\Phi^{(\ell)}_{\text{step}}$, a sequence of Ex. 4.8, the gadget $\mathfrak{G}^{(\ell)}_{\text{bandpass}}(a)$ can be described analytically, in terms of these phases. □

Proof. As is stated, we perform a composition of $(1,1)$ gadgets: the constant shift gadget $\mathfrak{G}_{\text{shift}}(T_4^{-1}(-T_4(a)))$ of Ex. 4.7 and the step function gadget $\mathfrak{G}^{(\ell)}_{\text{step}}$ of Ex. 4.8. Since $\mathfrak{G}^{(\ell)}_{\text{step}}$ is atomic, it is not necessary to perform any kind of corrective protocol: the composition $\mathfrak{G}^{(\ell)} \circ \mathfrak{G}_{\text{shift}}(2T_4^{-1}(a^2))$ immediately achieves the function $f\left(\frac{1}{2}T_4(x) - \frac{1}{2}T_4(a)\right)$, where $f(x)$ is an ε-approximation of $\text{sign}(x)$ on the interval $x \in [-1,-\delta] \cup [\delta,1]$. Suppose $x \in [-a+\delta, a-\delta]$. Then, since $x = \pm 1/\sqrt{2}$ are the local minima of $T_4(x)$ and $a \leq 1/\sqrt{2}$,

$$\begin{aligned}
\frac{1}{2}T_4(x) - \frac{1}{2}T_4(a) \geq \frac{1}{2}(T_4(a-\delta) - T_4(a)) &= 4\left((a-\delta)^4 - (a-\delta)^2 - a^4 + a^2\right) && (183)\\
&= 4\left(a^2 - (a-\delta)^2 - \left((a-\delta)^2 + a^2\right)\left(a^2 - (a-\delta)^2\right)\right) && (184)\\
&= 4(a^2 - (a-\delta)^2)(1 - (a-\delta)^2 - a^2) && (185)\\
&= 4(2a\delta - \delta^2)((1-2a^2) + (2a\delta - \delta^2)) && (186)\\
&\geq 4(2a\delta - \delta^2)^2 \geq 4a^2\delta^2. && (187)
\end{aligned}$$

Thus, we require that f is an ε-approximation of the sign function on $[-1,-4a^2\delta^2] \cup [4a^2\delta^2, 1]$ in order for the ε-condition to hold. The resulting depth ζ' of this gadget is then given by substituting $\delta \mapsto 4a^2\delta^2$ in Eq. (21), so

$$\zeta' = \mathcal{O}\left((2a\delta)^{-4(\nu_\ell+1)}\log^{1+\nu_\ell}\left(\varepsilon^{-1}\right)\right) \tag{188}$$

and the proof is complete. □

We now provide a proof of Thm. 4.1, which we restate for convenience.

Theorem 4.1 (Inverses of Chebyshev polynomials)**.** Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists $(1,1)$ gadgets of depth and cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieve functions $T_{2^n}^{-1}(x)$ (right-inverses) for $x \in [-1+\delta, 1-\delta]$, using a single qubit (no ancillae). □

Proof. Let Φ_n be the length $\zeta = \mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ protocol of Cor. F.3.1 which approximately achieves $e^{i\theta\sigma_z} \mapsto e^{i\theta\sigma_z/2^n}$ for $\cos(\theta) \in [\sqrt{\delta}, 1]$. Let $\mathfrak{G}^{(n)}$ be the corresponding $(1,1)$ atomic gadget. Now, consider the $(3,1)$ atomic gadget with $\Xi \equiv \Phi = \{0,0,\phi_0,0,\phi_1,0,\ldots,\phi_{N-1}\}$, where $\Xi' \equiv \Phi_n = (\phi_0,\ldots,\phi_{N-1})$ is the protocol obtained from $\mathfrak{G}^{(n)}$. Let $S = \{2,1,0,1,0,\ldots,1,0\}$. The $(1,1)$ gadget obtained from $\mathfrak{G}^{(n)}$ by pinning (Def. A.1) the index-1 input as $U_1 = e^{-i\pi\sigma_x/2}$ and the index-2 input as $U_2 = e^{i\pi\sigma_x/2^{n+1}}$ then achieves, given an embeddable unitary $U_0 = e^{i\arccos(x)\sigma_x}$, the unitary

$$U = \Phi[U_0, U_1, U_2] = e^{i\pi\sigma_x/2^{n+1}}\Phi_n[e^{i(\arccos(x)-\pi/2)\sigma_x}]. \tag{189}$$

Since $x \in [-1+\delta, 1-\delta]$, it follows that $\cos(\arccos(x) - \pi/2) = \sqrt{1-x^2} \in [\sqrt{2\delta-\delta^2}, 1] \subset [\sqrt{\delta}, 1]$. Thus, by Cor. F.3.1,

$$||U - e^{i\arccos(x)\sigma_x/2^n}|| = ||\Phi_n[e^{i(\arccos(x)-\pi/2)\sigma_x}] - e^{i(\arccos(x)-\pi/2)\sigma_x/2^n}|| \leq \varepsilon. \tag{190}$$

and the proof is complete. □

Let us now discuss the gadgets which enact the transformations of Eq. (17) and Eq. (18). Indeed, achieving these transformations is simply a matter of pre-composing protocols yielding these formulas with the Chebyshev inversion protocol Thm. 4.1. What is particularly convenient about these protocols is that they *require no correction*. The inverse Chebyshev polynomials fall under the umbrella of approximate phase functions, discussed in Sec. D.2, so their outputs will automatically be ε-embeddable. We then have the following results.

Theorem 4.2 (Arbitrary approximate multiplication)**.** Given $0 < \delta \leq 1$ an $0 < \varepsilon \leq 1/2$, there exists a $(2,1)$ gadget of depth/cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieves the function $f(x_0,x_1) = x_0x_1$ for all $x_0 \in [-1+\delta, 1-\delta]$ and $x_1 \in [-1,1]$ using no ancilla qubits. □

Proof. Denote the $(1,1)$ gadget achieving and ε-approximation of $T_2^{-1}(x)$ for $x \in [-1+\delta, 1-\delta]$ (Thm. 4.1) by $\mathfrak{G}$: this gadget will have a cost/depth of $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$. Moreover, the output will be ε-embeddable (Rem. D.3). Let $\mathfrak{G}'$ denote the $(2,1)$ gadget which partially composes $\mathfrak{G}$ with the first leg of the multiplication gadget of Ex. 4.3. The resulting gadget will have the same asymptotic cost, and will 2ε-achieve $T_2(f(x_0))x_1$ for x_0 and x_1 in the desired ranges, where f is an ε-approximation of $T_2^{-1}(x)$ (the 2ε-error comes from the fact that one of the oracles, being used twice, is ε-embeddable, so we apply Lem. D.4). Then, note that $T_2'(x) = 4x$, so $T_2(x)$ has a Lipschitz constant of 4 on $[-1,1]$, so

$$|T_2(f(x_0))x_1 - x_0x_1| \leq 4|f(x_0) - T_2^{-1}(x_0)| \leq 4\varepsilon. \tag{191}$$

Thus, the total error is of order $\mathcal{O}(\varepsilon)$, and the proof is complete. □

Theorem 4.3 (Arbitrary approximate addition)**.** Given $0 < \delta \leq 1$ and $0 < \varepsilon \leq 1/2$, there exists a $(2,1)$ gadget of depth/cost $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/4}))$ which ε-approximately achieves the function $f(x_0,x_1) = (x_0+x_1)/2$ for all $x_0, x_1 \in [-1+\delta, 1-\delta]$ using no ancilla qubits. □

Proof. We proceed similarly to the previous theorem. Denote the $(1,1)$ gadget achieving an ε-approximation of the function $T_4^{-1}(x)$ for $x \in [-1+\delta, 1-\delta]$, at a depth of $\mathcal{O}(\delta^{-1}\log(\varepsilon^{-2}\delta^{-1/2}))$. As before, the output of this protocol of ε-embeddable. We compose two instances of $\mathfrak{G}$ with the input legs of the addition gadget of Ex. 4.6. This will yield a $\mathcal{O}(\varepsilon)$-approximation of $(T_4(f(x_0)) + T_4(f(x_1)))/2$ for $f(x)$ and ε-approximation of $T_4^{-1}(x)$. Note that $T_4'(x) = 32x^3 - 16x$, which has 16 as its maximum, so the Lipschitz constant on $[-1,1]$ is 16, and

$$\left|\frac{T_4(f(x_0)) + T_4(f(x_1))}{2} - \frac{x_0+x_1}{2}\right| \leq 8\left(|f(x_0) - T_4^{-1}(x_0)| + |f(x_1) - T_4^{-1}(x_1)|\right) \leq 16\varepsilon. \tag{192}$$

Therefore, the total approximation error is on the order $\mathcal{O}(\varepsilon)$, and the proof is complete. □

To conclude our discussion of addition and multiplication gadgets, we will consider the embeddability properties of the gadgets produced from these protocols.

Definition H.1 (Domain-modified gadget)**.** Given an (a,b) gadget $\mathfrak{G}$, define the parameterized family of gadgets $\mathfrak{G}(\phi)$ to be those which achieve the unitaries $e^{i\phi\sigma_z/2}U_k'e^{-i\phi\sigma_z/2}$ for $k \in [b]$. Generally speaking, particular choice of ϕ will allow us to vary the domain on which the unitaries produced by the gadget are half-twisted embeddable, and therefore can be corrected with the ancilla-free corrective procedure. We let $\mathfrak{G}_{\text{mult}}(\phi)$ and $\mathfrak{G}_{\text{add}}(\phi)$ denote the domain-modified multiplication and addition gadgets of Ex. 4.3 and Ex. 4.6. □

Remark H.1 (Half-twisted embeddability of multiplication and addition)**.** For the case of multiplication, it is easy to compute the real component of the off-diagonal element of the achieved unitary. In particular, if we let $U'(x_0,x_1)$ be unitary achieved by $\mathfrak{G}_{\text{mult}}(0)$, then

$$\Im\left[\langle 0|U'(x_0,x_1)|1\rangle\right] = \sqrt{1-x_1^2}, \tag{193}$$

which will always be greater than or equal to 0. In fact, given the restriction of $x_1 \in [-\sqrt{1-\delta^2}, \sqrt{1-\delta^2}]$, it follows that $\Im\left[\langle 0|U'(x_0,x_1)|1\rangle\right] \geq \delta$. For the general product gadget of Thm. 4.2, because the Chebyshev inversion only occurs on the leg encoding variable x_0, the same holds true for this case. In summary, *both multiplication gadgets always yield half-twisted embeddable output* (Rem. D.4).

Now, we turn our attention to the addition gadgets. Let $U_\phi'(x_0,x_1)$ denote the unitary achieved by $\mathfrak{G}_{\text{add}}(\phi)$. As is easy to verify by hand or via symbolic mathematical software, we have, for example

$$\Im\left[\langle 0|U_\pi'(x_0,x_1)|1\rangle\right] = (2x_0 - 4x_0^3)\sqrt{1-x_0^2} + (2x_1 - 4x_1^3)\sqrt{1-x_1^2}, \tag{194}$$

$$\Im[\langle 0|U_{\pi/2}'(x_0,x_1)|1\rangle] = (2-4x_1^2)x_0\sqrt{1-x_0^2} + (4x_0^2-2)x_1\sqrt{1-x_1^2}. \tag{195}$$

The conditions under which these outputs have fixed sign are more complicated than the case of multiplication, but it is apparent that they *can* achieve fixed sign for significant domains of interest in each of the variables x_0 and x_1. For example, for $\phi = \pi$, if we are guaranteed that $x_0, x_1 \in [0, 1/\sqrt{2}]$, then Eq. (194) will always be non-negative, and the output will be half-twisted embeddable by Rem. D.4. There exist many such regions in the domain of x_0, x_1, which can be utilized to guarantee half-twisted embeddable output on a case-by-case basis. However, half-twisted embeddability *will not hold generically* for the addition gadget. □

Now, we revisit some of the examples discussed in Sec. 4. We begin with the example of the 2^n-mean.

H.2 Contrasting gadgets with LCU: the 2^n-mean

Our goal is to explicitly characterize the fundamental differences between using the gadget method vs. LCU for achieving the 2^n-mean of variables x_k which are encoded in σ_x-rotations $e^{i\arccos(x_k)\sigma_x}$. We assume, for the sake of performing corrections, that we have access to controlled variants of the oracles we are summing.

Example 4.10 (2^n mean)**.** Let $\mathfrak{G}$ be the $(2,1)$ gadget achieving $(x_0+x_1)/2$. Then one can build a composite $(2^n,1)$ gadget $\mathfrak{G}'$, as well as the correction protocol, achieving $2^{-n}(x_0+x_1+\cdots+x_{2^n-1})$ using (2^n-1) instances of $\mathfrak{G}$ and successively pooling pairs of variables. E.g., $(x_0+x_1)/2$ and $(x_2+x_3)/2$ are taken to their average $(x_0+x_1+x_2+x_3)/4$. □

Remark H.2 (Incomparability for LCU and gadgets)**.** Even in the natural context of taking a sum, the gadget formalism and LCU are still *highly incomparable*, as the circuit-level operations they achieve are fundamentally different. Despite the fact that LCU achieves better scaling in error/domain parameters over the gadget technique (as will be demonstrated), the gadget access model, when defined correctly, is much more difficult to achieve. In particular, the gadgets presented take a collection of oracles $e^{i\arccos(x_k)\sigma_k}$ and output an approximation of the *unitary* $V=e^{i\arccos((x_0+\cdots+x_{2^n-1})/2^n)\sigma_x}$. On the other hand, LCU takes the same set of oracles and outputs a $\widetilde{\mathcal{O}}(2^n)$-dimensional block-encoding of the operator $V'=\frac{1}{2^n}\left(e^{i\arccos(x_0)\sigma_x}+\cdots+e^{i\arccos(x_{2^n-1})}\right)$, which is generally non-unitary, and therefore can only be applied to a particular, *known* state $|\psi\rangle$ via amplitude amplification.

We refer to the former access model as *strong* and the latter as *weak*, to emphasize the clear separation between the two operations realized, despite the fact that their functional form in the particular block being considered is the same. □

To achieve the desired output with precision ε over $x_0,\ldots,x_{2^n-1}$ all within $[-1+\delta,1-\delta]$ with gadgets, we require repeated use of the addition and corrective protocols. In particular, to generate an embeddable sum of input leg variables, we require usage of both the addition and correction protocols, which when composed will result in a gadget of circuit depth $\widetilde{\mathcal{O}}(\zeta n\delta^{-1}\log(\varepsilon^{-1}))$ in order to achieve an $\varepsilon/2^n$-approximation of the desired sum, for ζ one of the choices of Theorem 2.1 (this choice also dictates number of required ancilla qubits and success probability). Performing the n-fold successive pooling of 2^n individual oracles will therefore result in a circuit of depth $\widetilde{\mathcal{O}}((Cn)^n\zeta^n\delta^{-n}\log^n(\varepsilon^{-1}))$, for some constant C.

We can now outline the individual methods:

- **2^n-mean gadget with ancillae (and controlled access)**: In the case that we make use of the correction protocol which utilizes ancillae and controlled access, it is straightforward to see that the nesting depth of the gadget network is n, where at each layer, two legs are paired with each other. Thus, the total number of ancillae used throughout the gadget is $\mathcal{O}(n)$: the same complexity as used by LCU to add 2^n individual block-encodings. In this case, performing an n-fold nesting of these gadgets will result in a circuit of depth $\widetilde{\mathcal{O}}((C_1n)^n\delta^{-2n}\log^{2n}(\varepsilon^{-1}))$ for some constant C_1 as $\zeta=\widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$. Such a gadget will achieve the desired sum as an ε-approximation with probability at least $(1-\varepsilon/2^n)^{2^n}\geq 1-\varepsilon$.

- **2^n-mean gadget without ancillae**: In the case that we are able to make use of the ancilla-free correction protocol, $\zeta=\widetilde{\mathcal{O}}(\delta^{-1}\gamma^{-2}\log^2(\varepsilon^{-1}))$. Thus, with unit success probability and no ancilla qubits, the resulting gadget has circuit depth $\widetilde{\mathcal{O}}((C_2n)^n\gamma^{-2n}\delta^{-2n}\log^{3n}(\varepsilon^{-1}))$ to achieve an ε-approximation of the desired sum.

- **2^n-mean with LCU**: Finally, when using LCU, we are able to achieve a block-encoding of 2^n individual oracles with circuit depth $\mathcal{O}(2^n)$, and $\mathcal{O}(n)$ individual ancilla qubits. However, in general, this will create a block-encoding in a much higher-dimension space than the gadget-based protocol. Thus, if we wish to *access* this block-encoding, and apply it to a particular state, we will need to perform rounds of amplitude amplification.

  In particular, suppose we wish to construct a quantum circuit which applies the sum V' of Rem. H.2 to a particular single-qubit state $|\psi\rangle$ with success probability $1-\varepsilon$ (the same as the gadget case). The initial success probability will be $|\langle\psi|(V')^\dagger V'|\psi\rangle|^2$. Generally speaking, the overlap can be very small, if the sum of the operators being computed is close to 0. Since we are assuming that $\arccos(x_k)\in[0,\pi]$ for each k, this sum can only be close to 0 if the x_k lie too close to -1 and 1. Indeed, consider the case where of the 2^n values of x_k being summed, 2^{n-1} of them are $1-\delta$

and 2^{n-1} are $-1+\delta$: the resulting average of σ_x-operations will be δ-close to 0, and the success probability will be δ-small. More formally, we restrict $x_k \in [-1+\delta, 1-\delta]$, and in the worst case (the case of the previous sentence), we have

$$V' = \begin{pmatrix} 0 & i\sqrt{1-(1-\delta)^2} \\ i\sqrt{1-(1-\delta)^2} & 0 \end{pmatrix} = i\sqrt{2\delta-\delta^2}\sigma_x \tag{196}$$

and the resulting success probability will be lower-bounded by δ. By the variant of fixed-point oblivious amplitude amplification of [26], to boost the success probability to $1-\varepsilon$, we require $\widetilde{\mathcal{O}}(\delta^{-1}\log(\varepsilon^{-1}))$ executions of the LCU circuit, for a total circuit depth of $\widetilde{\mathcal{O}}(2^n\delta^{-1}\log(\varepsilon^{-1}))$.

Method	Depth	Anc.	Domain	Succ.	Access model
Gadget w/ anc.	$\widetilde{\mathcal{O}}((C_1 n)^n \delta^{-2n}\log^{2n}(\varepsilon^{-1}))$	$\widetilde{\mathcal{O}}(n)$	$\mathcal{D}$	$1-\varepsilon$	Strong
Gadget w/o anc.	$\widetilde{\mathcal{O}}((C_2 n)^n \gamma^{-2n}\delta^{-2n}\log^{3n}(\varepsilon^{-1}))$	None	$\mathcal{D}'$	1	Strong
LCU	$\widetilde{\mathcal{O}}(2^n\delta^{-1}\log(\varepsilon^{-1}))$	$\widetilde{\mathcal{O}}(n)$	$\mathcal{D}$	$1-\varepsilon$	Weak

Table 3: A table summarizing the differences between utilizing gadgets vs. LCU to construct the 2^n-mean block-encoding, as presented above. Note that $\mathcal{D} = [-1+\delta, 1-\delta]$. In addition, $\mathcal{D}' \subset \mathcal{D}$, determined on a case-by-case basis, as the domain is determined by the half-twisted embeddability of the constituent variables being summed. For more information, see Rem. H.1. As discussed in the main text, the apparent improvement by LCU made here is because the gadget-achieved block-encoding in the gadget setting can uniquely be used coherently as a *unitary* (where the mean has been computed *inside the phase*) by another quantum protocol.

I Functional programming, QSP/QSVT, and gadgets

This paper does not construct a quantum programming language, but it does denote a series of operations within quantum computing in a succinct, abstracted style. It will turn out that these operations are most cleanly interpreted *functionally*; by *functional* we mean that each statement in our notation (either atomic or composite) operates by transforming a set of inputs to a set of outputs [62, 63]; commonly this is contrasted with *imperative* operations, by which one sequentially updates a collection of global variables. In a functional setting, one builds programs through the application and composition of functions; this is a so-called declarative paradigm, by which one describes the logic of a program rather than its control flow. In our setting, function definitions are trees of expressions mapping values to other values, rather than a sequence of statements updating some program state. In practice, there is some debate and disagreement over the best characterization of functional programming conceits, as well as programming languages which mix and match between paradigms; however, for our purposes, we will truly consider only the construction of composite functions.

A defining feature of most functional programming languages is the treatment of functions as first class objects; they can be passed as arguments, bound to names, and returned by other functions, just as any other data type. In this, small functions can be combined in a modular style to yield composite processes of greater complexity. The claim of functional programming (and especially its strongly typed and pure subclasses) is that such constraints mitigate *side-effects* (modifications of the program state outside the local environment) and thus aid program correctness and programmer reasoning. Functional programming has longstanding ties to theoretical computer science through the lambda calculus, and underlies a variety of general purpose and academic programming languages, realizing many of its purported benefits.

We have claimed that the constructions in this work allow us to discuss QSP and QSVT in the context of *functional programming*. Interestingly, in contrast to the classical case, where many aspects of programming language design can appear syntactical (or at worst superficial), we show that unique aspects of coherent quantum computation promote a constructive role for techniques of functional programming; the protocols we provide, in satisfying required properties of common objects in functional programming, rooted in methods in category and type theory, can immediately be reasoned about (and optimized over) using a wealth of preëxisting (classical) literature.

Before discussing the connection between functional programming and our setting, we provide a remark on the differences between QSP and QSVT, which are largely cosmetic in the context of our results. In this way, the rest of this section (and the paper as a whole) can concern itself with the simpler, single-qubit setting, and results can be lifted to QSVT if desired, up to the conditions discussed below.

Remark I.1 (On differences between QSP and QSVT)**.** In much of this paper we refer to QSP and QSVT jointly; this is no coincidence, as the latter is often viewed as a lifted version of the former [26], in which QSP occurs within privileged small subspaces spanned by the singular vectors of carefully engineered linear operators (this can be understood as a consequence of Jordan's Lemma [33, 57], or more simply the cosine-sine decomposition [67]). For our purposes, most results valid for QSP are also valid for QSVT within these preserved subspaces, and indeed it is this simultaneous manipulation of (possibly exponentially many) eigenvalues or singular values that often underlies exponential advantage for QSVT-based algorithms, e.g., those for Hamiltonian simulation or matrix inversion [26, 47]. There are caveats to this statement, many addressed in recent work on dequantizing QSVT under low-rank assumptions [10], but in the general case we assume that the difficulty of computing a linear operator's SVD underlies the classical hardness.

Despite technical analogousness, presentations of QSP and QSVT often switch between pictures based on interleaved rotations versus interleaved reflections. Following [26, 48], we repeat this distinction here, to be used for anyone wishing to convert this work (which uses the rotation convention throughout) to the reflection picture (or vice versa).

$$\Phi[U] \equiv e^{i\phi_0\sigma_z} \prod_{k=1}^{n} U e^{i\phi_k\sigma_z} \quad \text{(Rotation)}, \tag{197}$$

$$\Phi[V] \equiv e^{i\phi_0\sigma_z} \prod_{k=1}^{n} V e^{i\phi_k\sigma_z} \quad \text{(Reflection)}, \tag{198}$$

where the rotation U and reflection V are parameterized by $x \in [-1, 1]$ in the usual way:

$$U \equiv \begin{bmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{bmatrix}, \qquad V \equiv \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}. \tag{199}$$

Converting between these two pictures following from the simple identity $V = ie^{-i(3\pi/4)\sigma_z} U e^{I(\pi/4)\sigma_z}$, from which we see that the phases Φ for an equivalent rotation-based protocol can be derived, given phases Ψ for a reflection-based protocol, by

$$\phi_k = \psi_k - \pi/2, \quad k \in \{1, \cdots, n-1\}, \tag{200}$$

$$\phi_0 = \psi_0 - 3\pi/4, \tag{201}$$

$$\phi_n = \psi_n + \pi/4. \tag{202}$$

where the two unitaries will agree up to an unimportant overall phase depending only on n. For all discussions of antisymmetric protocols, we will refer to the reflection picture, though analogous conditions can be given for the reflection picture using the above map. Up to this distinction, and within the rank of a given non-square operator transformed by QSVT, we can treat our results on semantic embedding identically to the QSP case. □

The naturalness of applying functional programming to our setting stems from the character of QSP and QSVT. In these algorithms a succinct, classical data structure (a list of real phases) corresponds to a succinct, classical mathematical object (a polynomial transformation). However, the method by which this transform is enacted (namely, the aspect of QSP and QSVT which requires a quantum computer), means that the QSP phases and the realized transform are only of practical use when wrapped in a unitary operation (i.e., the circuit realizing the QSP or QSVT protocol). The input and output of a QSP and QSVT protocol are thus in practice, depending on the choice of the algorithmist, very different. While we might like to reason purely about the application, manipulation, and composition of polynomial functions, it is not obvious how the pre-image of the quantum circuit realizing such transforms, while respecting the contiguity of subroutines, behaves, or whether such a pre-image can even exist.

It turns out that it is precisely the business of functional programming to specify patches for the combination of subroutines such that disparate inputs and outputs can be automatically wrapped, handled, and verified. In turn this restores high-level algorithmic reasoning in an evidently safe and intuitive way.

The composition of QSP- and QSVT-based subroutines at the level of their achieved functional transforms can be seen in the single-variable setting in [48, 60], where it is noted that the obvious method for composing circuits *does not* induce composition of the achieved transform save under special global conditions on the QSP phases. The subtlety of this condition is made more evident in [48], where the requirement of real functional transforms is not explicitly mentioned, perhaps under the assumption

that the non-real case merely analytically continues the achieved composition, which as shown in [60] is generally untrue.

Before discussing our case more specifically, we remark that not only does this work not introduce a full quantum programming language, but that there has been extensive research on the design of (especially functional) full quantum programming languages [27, 55, 62, 63], for which the reader is directed to consult an excellent high-level survey in [23]. For the most part these programming languages rely on sophisticated mathematical techniques to address the unique features in quantum computation (e.g., entanglement, superposition, measurement collapse) as they pertain to program compilation, verification, control flow, and general design considerations. Nevertheless, the general difficulty of algorithm design in such expansive languages remains, albeit in a modified, organized form. In our work, we consider only a limited class of (surprisingly expressive and historically useful) quantum computing algorithms, and borrow results from functional programming theory to characterize their combination in a high-level and interpretable way.

I.1 Natural transformations and the QSP gadget

The usefulness of QSP and QSVT as quantum algorithms is founded in reasons so clear that they become difficult to see. Mainly, instead of quantum circuits, these algorithms allow one to consider continuous functions—said functions, in turn, are guaranteed to be applicable to the spectra of large linear operators. This action of moving from a circuit picture to a functional picture hides a series of homomorphisms. In this subsection we give these homomorphisms (of which there are many) names, and discuss their role in constituting (to borrow a category-theoretic term [42]) *natural transformations*. As discussed in previous work [60], the difficulty arising from these homomorphisms is that various manipulations in their images may not correspond to sensible manipulations in their pre-images. Understanding when such analogies can be formally restored (to borrow category-theoretic terms again, when certain diagrams *commute*) is a benefit of this work, and one motivation for applying techniques from functional programming, type theory, and category theory to QSP- and QSVT-based quantum algorithms.

Remark I.2 (Pictures for QSP and QSVT)**.** This remark aims to clarify a misconception possible with the diagrams we present. In the depiction of quantum computational programs, one often relies on the circuit depiction [53] in which quantum gates (unitary operations) represented by boxes are applied to qubits (or some other finite-dimensional assembled quantum systems), represented by lines, in organized diagrams in which time moves from left to right. Such diagrams clearly depict the gate-level evolution of quantum *states*, taking them as input and producing them as output. One can also consider the depiction of (specifically QSP- and QSVT-based) algorithms which take as input oracular unitary processes. In this case, boxes depict gadgets (e.g., a QSP or QSVT superoperator), and lines indicate input and output *quantum processes*. This work only lightly considers the direct depiction of quantum circuits in terms of their gates, and instead primarily focuses on the combination and composition of *higher-order quantum processes* whose inputs and outputs are themselves quantum processes.

The functional programming character of this work stems from the observation that there are two natural levels from which to consider the combination of higher-order processes. The first is in gluing together the inputs and outputs of these processes in a diagrammatic way, as discussed in Fig. 9. The second concerns the underlying scalar transformations achieved by these superoperators through the language of QSP and QSVT; these polynomial functions can also be manipulated and composed, and it is the business of this work to ensure that actions in this *functional space* (i.e., functional composition, which is semantically clear) are reflected by simple actions in *gadget space* (i.e., superoperator composition). In this way, algebraic manipulations are reduced to geometric ones, and both are guaranteed to have easy to compute circuit pre-images.

At the end of the day, we still desire that these computations are achievable at the gate level, respecting the contiguity of subroutines used in black-box ways. It is precisely the statements of the main theorems of this work that this achieveability is summarized by the valid linking of superoperators in the first picture discussed above. Consequently, if the *functional space* has a valid diagram (a pre-image) in *gadget space*, then our desired functional manipulation is guaranteed to be both realizable by a simple circuit, and have well-characterized runtime and resource requirements. □

Previous work by two of the authors of this work discussed the limited combination and composition of single-variable QSP and QSVT protocols; in this case the major category-theoretic object referenced was the *natural transformation*, the properties of which gave name and structure to the major theorems of that work [60]. While the section following this one will primarily refer to objects in functional programming

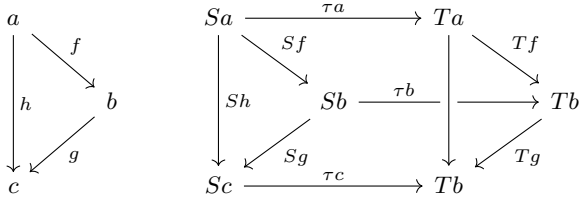


Figure 8: A diagram presenting a natural transformation between morphisms S and T. Special assignments from quantum algorithms to the objects depicted here provide a concrete definition for semantic embedding in [60], from which the assignments in Table 4 are borrowed.

proper, the connection between category theory and common functional programming objects is strong enough to be of use (and we will refer back to the diagram given in Fig. 8 when discussing definitions of monads in the next section, which can also themselves be defined in terms of diagrams, per those of Def. I.2).

Natural transformations can be thought of as a method to discuss relations between functors (they are *morphisms between functors*), where functors are themselves objects which translate category theoretic diagrams to analogous diagrams (they are *morphisms between categories*). In [60], we formalized the two pictures in Rem. I.2 in terms of a series of functors over QSP/QSVT-specific mathematical objects (e.g., QSP phase lists, QSP unitaries, polynomial transforms, etc.), by which the *components* of the natural transformation became the various homomorphisms which allow one to move between objects acted on by various functors. This culminates in definition statement (reproduced from [60] in Def. I.1) which specifies that the embedding of a functional transformation in a QSP/QSVT protocol is *semantic* if there exist special functors over a category of QSP phase-lists which permit the diagram in Fig. 8 to *commute* (i.e., that any traversal along its arrows yields the same result if its start and end points are the same).

Definition I.1 (Semantic embedding for QSP)**.** Taking the diagram in Fig. 8, we say that the morphisms f, g, h are *semantically embedding* for QSP protocols if S, T as translated by Table 4 are such that Sf, Sg, Sh correspond to circuit composition (respecting contiguity of the inner phase list), Tf, Tg, Th correspond to polynomial composition, the components of the natural transformation $\tau a, \tau b, \tau c$ are simple projection from a unitary operator to its top-left matrix element, and the diagram given commutes. Moreover, the functions f, g, h should be efficiently computable. □

This appendix is not meant to meticulously cover results in category theory (for which we refer a reader to the appendices of [60] or a common textbook [42]). Rather, we use this space to present that the conditions we care about in combining QSP/QSVT subroutines as modules are of great interest within category theory, which has already developed advanced structures to describe and analyze them.

Category picture	QSP picture	Description
a, b, c	Φ	QSP phase list
f, g, h	$\Phi_0 \to \Phi_1 \circ \Phi_0$	QSP phase nesting
Sa, Sb, Sc	$U(\Phi)$	QSP unitary
Sf, Sg, Sh	$U(\Phi_0) \to U(\Phi_1 \circ \Phi_0)$	QSP unitary embedding
Ta, Tb, Tc	$P(x)$	Polynomial transform
Tf, Tg, Th	$P_0(x) \to (P_1 \circ P_0)(x)$	Polynomial composition
$\tau a, \tau b, \tau c$	$U(\Phi) \to P(x)$	Projection to matrix element

Table 4: A table relating the definition of natural transformations, and the instantiation of this diagram with objects and arrows (functions) in QSP. We suppress multi-labeling of QSP objects in the second column for brevity. A full explanation of the notation given here can be found in the source work [60].

The upshot of this work is that, following the major theorems in the main body, and specifically Thm. 3.1 and its corollaries in Appx. A, the algorithmist is free to diagrammatically link the QSP gadgets (with possibly many oracular inputs and unitary outputs) in any way permitted by a natural notion of their geometry. In Fig. 9, we summarize the two pictures specified above, considering our gadgets as circuits, and then as superoperators, as well as a simple case illustrating the free choices one has to compile simpler gadgets into larger ones. It is the authors' hope that this depiction is simple, and serves to bolster the linear, text-based description of these operations in Appx. A.

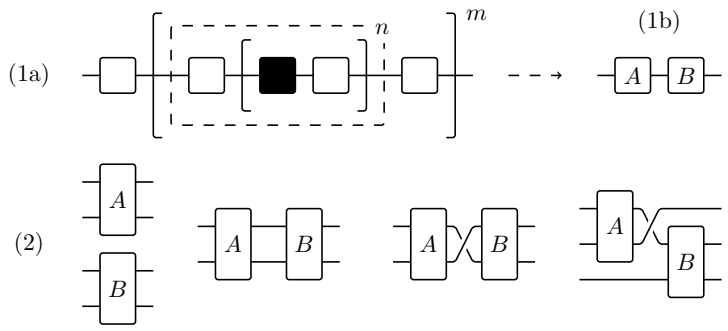


Figure 9: Simplified depiction of the circuit picture (1a) versus the gadget picture (1b) for single-variable recursive embedding of QSP [48, 60] (equivalently the composition of two $(1,1)$ gadgets). In the multi-input and multi-output setting the gadget picture of (2) offers significant simplification in enumerating partial compositions, each of whose functional actions can be interpreted following similar methods to those of Thm. 3.1, characterized by *interlinks* (Def. 3.1). In this example, $(2,2)$ gadgets A and B linked following a subset of the possible linkages, resulting in composite gadgets of type $(4,4)$, $(2,2)$, and $(3,3)$, as shown. For a full enumeration of such partial compositions, see Appx. A, and for computing their costs, consult Lem. C.1.

I.2 On the QSP and QSVT types, and their related monadic type

At a low-level, our work is concerned with the following problem: given repeatable access to single-qubit unitaries which are crucially (1) known to look as if they could have (approximately) resulted from QSP protocols with certain mild symmetries, and (2) encode in a specified matrix element a (Laurent) polynomial function of possibly many unknown variables, do there exist techniques to manipulate these matrix elements in well-understood and controllable ways while generating unitaries that respect properties (1) and (2). In other words, we are interested in superoperators that preserve (1)-(2), and more ambitiously (given that such operations are not too difficult to find) in the description of the closure under (1)-(2)-preserving operations. This closure is ideally described in terms of the achieved matrix elements, which are (Laurent) polynomial functions of some fixed set of underlying real-valued inputs.

Our access model is restrictive for an important reason: while our computation lives in SU(2), the underlying unknown parameters are real, and the achieved functions (as matrix elements) are also real and bounded. It is precisely that we project down to fixed matrix elements, and parameterize these same matrix elements by real values from an interval, that allows us to flexibly achieve non-linear transformations of said signals obliviously. However, for most intents and purposes, we are enticed to reason purely in terms of the real functions achieved, and moreover we would like to do this irrespective of the realizing dynamics (under some assurance that they are efficiently implementable, and computable). This scenario is quite common in functional programming, as mentioned, and we take the rest of this section to formalize our desire in terms of the natural object dealing with such scenarios: monads.

Commonly a monad is defined (for programmers) apart from category theoretic concepts [71–73]; while we will return to a category theoretic definition later, in part to match with the functors and natural transformations defined in [60], for the moment we consider a monad as defined in three parts (often called a Kleisli triple): these are a type constructor M, a type converter (often called `unit`), and a combinator (called `bind` or denoted `>>=`). The first of these builds up a type, and can be thought of as a specification for how `unit` wraps its given data (i.e., embeds it with other, surrounding information constituting the monad object), while `bind` specifies how to unwrap this data, apply a desired function to it, and return the result re-wrapped in the monadic type. These operations have to respect a few basic identities, namely

$$\texttt{unit}\ x\ \texttt{>>=}\ f \leftrightarrow f\ x \tag{203}$$

$$\texttt{ma}\ \texttt{>>=}\ \texttt{unit} \leftrightarrow \texttt{ma}, \tag{204}$$

$$\texttt{ma}\ \texttt{>>=}\ \lambda x \to (f(x)\ \texttt{>>=}\ g) \leftrightarrow (\texttt{ma}\ \texttt{>>=}\ f)\ \texttt{>>=}\ g, \tag{205}$$

where here λ is an anonymous function, and where these conditions formalize that `unit` is both the left and right identity for `bind`, and indicate that `bind` is associative. Ultimately, this construction gives rise to the ubiquitous (if opaque) expression that a monad is a *monoid in the category of endofunctors*, where `unit` and `bind` are the identity and binary operation respectively.

In the context of functional programming from a more purely category theoretic perspective, the structure of a monad can also be derived from a functor with two additional natural transformations

[49, 50]. We discuss this to make a connection to [60], which introduces semantic embedding in category-theoretic terms. In this case, one starts with a higher-order function (or functional/functor) named `map` with the following type, serving as our base functor:

$$\texttt{map}\ \phi : (\texttt{a} \to \texttt{b}) \to (\texttt{ma} \to \texttt{mb}). \tag{206}$$

Consequently, two objects `join` and `unit` can be defined in terms of this functor, the latter in satisfying the following identity,

$$(\texttt{unit} \circ \phi)\ x \leftrightarrow ((\texttt{map}\ \phi) \circ \texttt{unit})\ x. \tag{207}$$

The `join` object is a little more involved, and is as promised in category theoretic terms a *natural transformation* [41, 42] which allows one to flatten nested applications of the monad. It must satisfy the following three identities:

$$(\texttt{join} \circ (\texttt{map}\ \texttt{join}))\ \texttt{mmma} \leftrightarrow (\texttt{join} \circ \texttt{join})\ \texttt{mmma} \leftrightarrow \texttt{ma}, \tag{208}$$

$$(\texttt{join} \circ (\texttt{map}\ \texttt{unit}))\ \texttt{ma} \leftrightarrow (\texttt{join} \circ \texttt{unit})\ \texttt{ma} \leftrightarrow \texttt{ma}, \tag{209}$$

$$(\texttt{join} \circ (\texttt{map}\ \texttt{map}\ \phi))\ \texttt{mma} \leftrightarrow ((\texttt{map}\ \phi) \circ \texttt{join})\ \texttt{mma} \leftrightarrow \texttt{mb}, \tag{210}$$

To convert entirely to the diagrammatic methods of category theory, we can give an equivalent, more diagrammatic definition, breaking away from the identities as presented above (as well as those using `unit` and `bind` previously).

Definition I.2 (Monad, following [42])**.** A monad $T = \langle T, \eta, \mu \rangle$ (the aforementioned Kleisli triple) in a category X consists of a functor $T : X \to X$ and two natural transformations (here denoted by $\dot{\longrightarrow}$

$$\eta : I_X \dot{\longrightarrow} T, \quad \mu : T^2 \dot{\longrightarrow} T, \tag{211}$$

which assert that the following diagrams commute

$$\begin{array}{ccc} T & \xrightarrow{T_\mu} & T^2 \\ \downarrow{\scriptstyle \mu T} & & \downarrow{\scriptstyle \mu} \\ T^2 & \xrightarrow{\mu} & T, \end{array} \qquad \begin{array}{ccccc} IT & \xrightarrow{\eta T} & T^2 & \xleftarrow{T\eta} & TI \\ \| & & \downarrow{\scriptstyle \mu} & & \| \\ T & = & T & = & T. \end{array} \tag{212}$$

Here we can identify η with the identity and μ the multiplication of the monad T built from the endofunctor $T : X \to X$. In this way, we can recognize the two diagrams as summarizing the conditions previously cited for `unit` (here η) and `join` (here μ), with `map` (here T) acting as the base functor. In short, the left diagram shows associativity for the monad, while the right diagram express the left and right *unit laws*. These diagrams appear often within category theory, unsurprisingly being precisely those of a monoid whose base set (often denoted M) is here replaced by the endofunctor T. □

It turns out that the functor T defined in previous work [60] and depicted in Fig. 8 is precisely the endofunctor over which this paper instantiates a monad, as discussed in Rem. I.3, save in the more expansive, multivariable case, in which the realizing dynamics of `bind` are less simple, and have non-trivial resource cost. In this way, we see that the benefit of a functional programming characterization is not merely syntactic, but more accurately a guide toward a constructive superoperator which, if realized, immediately permits the desired declarative methods of functional programming.

Classically, monads are defined in a number of equivalent ways with differing terms, among the more common being `unit` (or `pure`) and `bind`, as mentioned previously, which ultimately ascribe a method for a programmer to pipeline subroutines together with declarative assurance that the input and output structure of these elements are respected and handled in the expected way (the common terms here are managing *control-flow* and *side-effects*). The common analogy is that a monad allows one to wrap data (here used in the broad, functional programming sense) such that the programmer is freed to reason agnostic to operational details, working purely over types. In our setting, these types are real scalar values and real-argument, real-valued multivariable polynomial functions. We can thus summarize the work of this paper in the following remark, which ties together our constructions and their natural description in terms of monads and their defining properties.

Remark I.3 (Constructing the QSP monad)**.** Given a real-valued polynomial function of possibly multiple variables, this function can be suitably embedded in a single qubit unitary given access to a basic

unitary process which encodes the argument(s) for the intended function in the angle of rotation about a known and fixed axis. This encoding is analogous to `unit`, with the standard type signature:

$$\texttt{unit} : T \to M\ T, \tag{213}$$

where T is some type (in this case polynomial functions), and where MT is the same function, but now encoded in a unitary process in the standard way (i.e., as the top left element of an antisymmetric QSP superoperator applied to an oracle unitary). In other words, QSP constitutes a monadic type in whose context one can interpret underlying simpler, polynomial functions. The primary goal of this work, then, is to show that even in the multivariable setting, `bind` is efficiently implementable, where we can write the standard type signature:

$$\texttt{bind} : (M\ T, T \to M\ U) \to M\ U, \tag{214}$$

where U is another type, here the same as T in most cases. The primary difficulty before this work was that, although we understood of the nature of the map $T \to M\ U$ quite well within QSP, i.e., from the unwrapped type T to the wrapped type $M\ U$, the action of wrapping a type through `unit` was not so obviously reversible in the general case. It just so happens that for single-variable QSP, antisymmetric phases, as discussed in [48, 60], make `bind` nearly trivial to implement. In our setting, applying a broadband-NOT, followed by an approximate square root, permits us to build a relatively efficient subroutine (under some mild bound restrictions on the achieved functions) for recovering `bind` in such a way as to satisfy the identities for unit. Moreover, we find that our construction not only permits the repeated self-embedding of multivariable QSP protocols within themselves (while respecting the desired action within T), but that it opens a variety of manipulations in T (including manipulations on functors over T) to be suitably lifted through M and thus realized within a quantum computation. In this way, the natural, mathematically attractive endofunctors of the space of real-valued polynomials are elevated as first class objects about which the algorithmist can reason unencumbered by circuit considerations. □

We could have also chosen to rewrite this monad in terms of `unit` and `join`, following from the natural transformation shown between the two functors over the space of polynomials and unitaries as discussed in [60]. In either case, the language of monads expediently enumerates the consequences of our constructions. Moreover, given there exist many disparate quantum circuits resembling QSP, which in turn realize disparate (and often useful) transformations in the algorithmically cognizable space of polynomial functions over scalars and operators, similar techniques to those in this work, by the great generalizing power of functional programming, hold promise in almost automatically applying to larger classes of composite quantum programs. Such functional programming methods for the manipulation of higher order quantum processes have been cursorily applied in the black box setting already [54].

Remark I.4 (Discussions on the QSP and QSVT types)**.** In the standard definition of a monad, the wrapped value $M\ T$ can be seen as the original data T along with bookkeeping objects designed to keep track of side effects [71–73]. In our setting, T is a class of well-defined mathematical objects (polynomial functions over scalar variables, possibly with multiple inputs and outputs), while $M\ T$ are quantum circuits (or unitaries) which in some sense realize these transforms over the spectrum of a given oracle unitary. In our case, we are apparently only given the ability to *actually run these programs*, and are thus only ever shown the wrapped monadic type, under the assumption that certain tasks achieved by QSVT do not admit efficient classical algorithms. Therefore, understanding how to apply monadic functions to these *QSP and QSVT types* is the only way to fruitfully manipulate these algorithms as subroutines. It is in this sense that we say functional programming techniques are naturally applied to QSP and QSVT; by specifying a satisfying assignment for the monadic functions, we free the algorithmist to reason about the much simpler multivariable polynomial transforms and their endofunctors. The QSP and QSVT types, in this case specially symmetrized circuits, are effectively identical up to the relation discussed in Rem. I.1, and by merit of their restricted structure can be manipulated, applied, and composed without leaving the coherent setting in which much of quantum advantage appears to reside. □

I.3 The grammar of gadgets

We have described gadgets as specifying modular computations, themselves possibly comprising simpler, analogous gadgets. It is worthwhile to understand such composite objects in the context of formal techniques from programming language theory; more narrowly, we focus on techniques for the specification of syntax, and methods for coupling said syntax to hierarchical structures within programming languages.

While this work does not specify a language for quantum computation generally, it does specify a language for enacting multivariable polynomials on the spectra of linear operators. Even this limited scope has shown great success in quantum algorithms and, in what follows, can be shown to have a substantively complex descriptive theory. In this section we discuss specifications of formal *grammars* and *languages*, connect these to our construction of gadgets, and explore some of the consequences of making this connection.

To keep this section brief, we do not discuss formal grammars from first principles, fronting only relevant objects, and otherwise pointing toward common references for the interested reader. Generally, the study of formal grammars seeks to investigate valid *sentences* built from a specified *alphabet* according to a formal *syntax*. These linguistic terms are no mistake, given the field's roots in the analysis of natural language [14], though such study has ultimately found greater purchase in the theory of programming languages, where one wishes to investigate valid *programs* built from specified *symbols* according to a formal *syntax*. Where possible we try to use terminology from programming language theory, but give common alternative names for objects where useful. We note that such *syntactical* studies do not concern the meaning or *semantics* of programs, only their structural correctness; nevertheless, understanding such structure is vital when writing *parsers* (recognizing and breaking down programs into atomic components) and *optimizers* (simplifying a program according to rigidly permitted rules based in syntax).

Definition I.3 (Context-free grammar (CFG))**.** A *context-free grammar* G is defined by a four-tuple (V, Σ, R, S) where

(a) V is a finite set, and each element $v \in V$ is called a *non-terminal character*. Each variable represents a different type of clause (phrase) in the program (sentence).

(b) Σ is a finite set of *terminal characters*, disjoint from V; this set is the alphabet of the language defined by the grammar of G.

(c) R is a finite relation in $V \times (V \cup \Sigma)^*$, where $*$ is the Kleene star operation. The members of R are often called the *productions* of the grammar.

(d) S is the *start symbol* of the program, and is a member of V.

A production rule R in a CFG is formalized as a pair (α, β) where $\alpha \in V$ is a non-terminal character and $\beta \in (V \cup \Sigma)^*$ is a finite string of terminals and non-terminals. Most commonly this production is written as $\alpha \to \beta$. Note that α can contain *only* non-terminal characters (hence context-free), and that β may be ε, the empty string. Commonly productions with the same right-hand-side are written together, with vertical bars separating alternative left-hand-sides, e.g, $\alpha \to \beta_0 \,|\, \beta_1$. □

Definition I.4 (Context-free language (CFL))**.** A *context-free language* of a context free grammar $G = (V, \Sigma, R, S)$ is the set of all terminal symbol strings derivable from the start symbol according to the productions in R:

$$L(G) = \{w \in \Sigma^* : S \xrightarrow{*} w\}, \tag{215}$$

where $\xrightarrow{*}$ indicates finitely repeated application of production rules (also referred to as a derivation). A language L is said to be a CFL if there exists a CFG G such that $L = L(G)$. □

A problem arises in the use of CFGs in programming language design; as their name suggests, such grammars (and their derived languages) cannot capture agreement or reference, where two different parts of a program must agree with each other in some way. To be sensitive to such distinctions while parsing, additional structure is required which substantially increases the complexity of the resulting languages and their analysis. Given that our gadgets can be of arbitrary size, and their connection involves agreement, e.g., in leg number, across perhaps large regions of the resulting diagram, it will turn our that the generic description of the grammar of gadgets will require such additional structure. We give a minimal presentation of one construction below. Before this, we provide an overarching comment on some common grammars of differing strength.

Remark I.5 (RGs, CFGs, CSGs, and UGs)**.** In the discussion of grammars, reference is often made to the *Chomsky hierarchy* [14], which specifies four common varieties of grammar in a hierarchy, where the set of languages generated by those grammars higher in the hierarchy strictly contain the set of those generated by lower grammars. These are: regular grammars (RGs), context-free grammars (CFGs), context-sensitive grammars (CSGs), and unlimited grammars (UGs). The defining characteristics of these grammars are the nature of their *productions*, which have increasingly general form as one moves

up the hierarchy. Generally, RGs and CSGs achieve the most study in programming language theory, given the efficiency of verifying that a given string exists in the language generated by one of these grammars. Nevertheless, CSGs and UGs have achieved attention historically given documented cases of super-CFG structure in both programming and natural languages. Moreover, there exist a variety of augmented grammars which fall between hierarchical levels or beyond them, as discussed below, again with documented use in programming languages. □

We now introduce an augmented grammar which, while outside of the Chomsky hierarchy, possesses a particularly compact structure that illustrates the sort of bookkeeping necessary in keeping track of the assembly of arbitrary gadgets. We stress that this construction, like all formal grammars, does not capture semantic structures, only syntactic ones. For QSP-derived gadgets, semantic structure concerns the achieved polynomial transforms, while the syntactic structure concerns only how such transforms can be combined. We have already established a diagrammatic formalism for understanding the combination of atomic gadgets, and thus what follows maps directly onto this diagrammatic formalism, allowing one in principle to verify, given a string of symbols from a (in our case possibly infinite) alphabet, whether such a string could have been generated by the specified grammar (and consequently whether a given assemblage of gadgets could have resulted from a valid diagrammatic composition).

Definition I.5 (Van Wijngaarden grammar (W-grammar))**.** W-grammars are two-level grammars, i.e., defined by a pair of grammars, that operate on different levels [69, 70]. The first is a *hypergrammar* (an attribute grammar [35]), which is a CFG whose non-terminal characters may have attributes. The second is a *metagrammar*, which is a CFG whose induced language defines the possibilities for the hypergrammar's attributes (its terminal strings). The language of a W-grammar is defined by a two step process:

(a) Within each hyperrule, for each attribute that occurs, choose a value generated by the metagrammar. The result is a standard CFG rule. Do this for all possible values.

(b) Using the resulting (possibly infinite) CFG, generate strings of terminal symbols in the normal way.

W-grammars are known to be Turing complete, and thus membership of a given string in a given W-grammar, or even whether a given W-grammar is empty or not, are undecidable. □

We now specify a W-grammar for gadget assemblages; in this case the metagrammar will define tuples (attributes) limiting the size of the initial gadget, and the size of permitted sub-gadgets into which said initial gadget is allowed to be decomposed. For each attribute, we can then define a CFG whose productions cover the decomposition of one gadget into two coupled gadgets, and the decomposition of an atomic gadget into an assembly of M-QSP protocols. Once more, we capture only the formal structure (syntax) of such protocols, not their achieved functional transforms (semantics).

Example I.1 (A W-grammar for gadgets)**.** We consider a two-level grammar (equivalently a W-grammar, Def. I.5) generating the language of compositions of atomic gadgets. For the hypergrammar we will define a finite series of hyperrules according to the possible attributes of non-terminal characters; in this case, these attributes will be tuples positive integers representing (1) the number of input or output legs on a gadget, and (2) the number of involved legs in an interlink. We give the CFG of this hypergrammar first, and then discuss the CFG generating the attributes present in the hypergrammar. Specifically, consider the following productions:

$$S \to A[a,b,c], \qquad a,b \leq c, \qquad \text{(Spawn gadget)} \quad (216)$$

$$A[a,b,c] \to A[a-e,f,c]B[d,c]A[e+d,b-f+d,c], \qquad 0 \leq e \leq a, 0 \leq d \leq f \leq c, \qquad \text{(Split gadget)} \quad (217)$$

$$A[a,b,c] \to A[a,d,c] \,|\, A[a,e,c] \,|\, A[f,b,c], \qquad d \in [b], b \leq e \leq c, f \in [a], \qquad \text{(Mod. gadget)} \quad (218)$$

$$A[a,b,c] \to C[a,b,c], \qquad \text{(Atomize)} \quad (219)$$

$$C[a,b,c] \to \phi_k C[a,b,c] \,|\, \phi_k, \qquad k \in [b], \qquad \text{(Add phase)} \quad (220)$$

$$C[a,b,c] \to \zeta_{j,k} C[a,b,c] \,|\, \zeta_{j,k}, \qquad j \in [a], k \in [b], \qquad \text{(Add oracle)} \quad (221)$$

$$B[d,c] \to B[d,c]\sigma_{j,k} \,|\, \sigma_{j,k}, \qquad j,k \in [d], \qquad \text{(Permute links)} \quad (222)$$

where one can think of the non-terminal $A[a,b,c]$ as a gadget with a input legs, b output legs, and which is allowed comprise gadgets with at most c input or output legs internally. Here $B[d]$ represents an interlink involving d legs. S is the start symbol as usual. Note that for fixed positive integers a,b,c, there are only finitely many productions, and thus this is a true CFG under for each valid attribute. The terminal

symbols, $\phi_k, \zeta_{j,k}, \sigma_{j,k}$ can be thought of as M-QSP phases for the k-th output leg, uses of the j-th M-QSP oracle for the k-th output leg, and a permutation between the j-th and k-th operating qubits respectively.

We have named each of the productions of the hypergrammar, which allow one to break down gadgets into smaller gadgets (splitting), apply common gadget superoperators (modification) such as elision, augmentation, and pinning, convert a gadget non-terminal symbol to an atomic gadget non-terminal symbol (atomization), and parameterize a series of M-QSP protocols within an atomic gadget. Note that for brevity we do not include a production enforcing antisymmetrization of the phases (though this is permissible with a CFG), and moreover note that this grammar does not account for atypical gadgets. Again for simplicity, note that permutation is a subset of interlinks covered by the final production.

In sum, this grammar allows derivation of unambiguous prescriptions for *assemblages of atomic gadgets*; these are distinct from the *quantum circuit realizing this assemblage*, and exemplifies the difference between semantic and syntactic aspects of language. Note also that this prescription does not capture the semantic relation between gadget legs (for instance in augmentation, elision, or pinning). In other words, the metagrammar is merely present to capture context-sensitive aspects of the resulting grammar necessary to constrain which gadget geometries can be combined. This mix of certain semantic and syntactic information while parsing is a hallmark of attribute grammars, and their main utility [35].

The metagrammar we desire to build is one which generates finite tuples of positive integers. This is pretty simple to do, as we only need pairs and triples of positive integers, and so we consider the following productions:

$$S \to A \,|\, B, \qquad \text{(Spawn tuple)} \tag{223}$$
$$A \to 0 \,|\, 1 \,|\, 0A \,|\, 1A, \qquad \text{(Add to tuple)} \tag{224}$$
$$B \to 0 \,|\, 1 \,|\, 2 \,|\, 0B \,|\, 1B \,|\, 2B, \qquad \text{(Add to tuple)} \tag{225}$$

where we have condensed productions from A and B in single lines. The resulting tuple can be determined by counting the number of each symbol in the resulting string, with order within the tuple specified by the natural numerical order of the symbols. Together, this and the attribute-augmented CFG define a W-grammar whose induced W-language unambiguously identifies an assemblage of atomic gadgets. □

Remark I.6 (On the notation in Ex. I.1)**.** We note that the production rules provided in the hypergrammar of Ex. I.1 uses some constants relied on elsewhere in this manuscript. As this subsection is self contained, we intend for these constants to be used only here. Nevertheless, the a, b used in the productions do indeed align with the (a, b) gadget notation. The positive integer c, the maximum number of input or output legs (equivalently, *arity/co-arity* or *valence/co-valence*) of a gadget within a given CFL, is not mentioned or relevant to other sections of this work. Likewise $B[d, c]$ refers to interlinks over d legs for gadgets with maximum arity c, while $C[a, b, c]$ are antisymmetric atomic gadgets, again with maximum arity c. We assume throughout derivations that input and output legs are in a definite, fixed order when instantiated, and all operations act with respect to this order; ultimately this is unimportant given the existence of the permutation production (222). □

A neat corollary to the above example follows from recognizing that the CFG of the hypergrammar can be considered on its own, for a fixed attribute. We discuss this in the example below, noting that most of the examples discussed in the main body of this work are low-degree gadgets.

Example I.2 (A CFG for limited gadgets)**.** It is worthwhile noting that when one is allowed to consider only gadgets of fixed maximum input and output leg number, the construction of the hypergrammar in Ex. I.1 demonstrates that a true CFG results. Consequently, if we restrict to this setting, we can recover many of the desirable properties of CFGs, including algorithms for efficiently verifying the correctness of a gadget assemblage. It is not clear, however, whether there is a concise way of understanding this more limited model in comparison to the general setting. □

While general composite objects built from atomic gadgets appear to be describable only by relatively complex grammars outside the Chomsky hierarchy [14], it is worthwhile to consider whether useful subgrammars for gadgets can be specified. For instance, in the case where one only considers $(1, 1)$ and $(2, 1)$ gadgets, or in fact any finite number of gadget arities, as well as restrictions on the possible assemblages of gadgets to trees or other cycle-free directed graphs, it should be possible to completely capture wide functional expressivity while maintaining efficient algorithms for the verification of membership in the induced language. There exists a wealth of literature on augmented CFGs which nevertheless maintain nice verification properties, for instance attribute grammars [35]. This in turn may permit the applicability of highly developed classical tools for compiler design and program optimization [40] to the theory of QSP-based gadgets.

However, specification of a minimal sufficient grammar for gadgets is not the main focus of this paper, and understanding optimization over such grammars in general may also depend on substantive problems in the theory of polynomial approximation and decomposition, given the relative complexity of the map between a specification of an assemblage of gadgets, and a specification of a quantum circuit resulting from that assemblage. Nevertheless, we believe that the concrete examples above support the validity of considering higher-order abstractions in the design of quantum algorithms for block encoding manipulations.