Efficient quantum amplitude encoding of polynomial functions

Javier Gonzalez-Conde[1,2], Thomas W. Watts[3], Pablo Rodriguez-Grasa[1,2,4], and Mikel Sanz[1,2,5,6]

[1]Department of Physical Chemistry, University of the Basque Country UPV/EHU, Apartado 644, 48080 Bilbao, Spain
[2]EHU Quantum Center, University of the Basque Country UPV/EHU, Apartado 644, 48080 Bilbao, Spain
[3]School of Applied and Engineering Physics, Cornell University, Ithaca, NY 14853, USA
[4]TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain
[5]IKERBASQUE, Basque Foundation for Science, Plaza Euskadi 5, 48009, Bilbao, Spain
[6]Basque Center for Applied Mathematics (BCAM), Alameda de Mazarredo, 14, 48009 Bilbao, Spain

Loading functions into quantum computers represents an essential step in several quantum algorithms, such as quantum partial differential equation solvers. Therefore, the inefficiency of this process leads to a major bottleneck for the application of these algorithms. Here, we present and compare two efficient methods for the amplitude encoding of real polynomial functions on n qubits. This case holds special relevance, as any continuous function on a closed interval can be uniformly approximated with arbitrary precision by a polynomial function. The first approach relies on the matrix product state representation (MPS). We study and benchmark the approximations of the target state when the bond dimension is assumed to be small. The second algorithm combines two subroutines. Initially we encode the linear function into the quantum registers either via its MPS or with a shallow sequence of multi-controlled gates that loads the linear function's Hadamard-Walsh series, and we explore how truncating the Hadamard-Walsh series of the linear function affects the final fidelity. Applying the inverse discrete Hadamard-Walsh transform converts the state encoding the series coefficients into an amplitude encoding of the linear function. Thus, we use this construction as a building block to achieve an exact block encoding of the amplitudes corresponding to the linear function on k_0 qubits and apply the quantum singular value transformation that implements a polynomial transformation to the block encoding of the amplitudes. This unitary together with the Amplitude Amplification algorithm will enable us to prepare the quantum state that encodes the polynomial function on k_0 qubits. Finally we pad $n - k_0$ qubits to generate an approximated encoding of the polynomial on n qubits, analyzing the error depending on k_0. In this regard, our methodology proposes a method to improve the state-of-the-art complexity by introducing controllable errors.

Javier Gonzalez-Conde: javier.gonzalezc@ehu.eus

1 Introduction

Over the past few decades, there has been a significant interest in quantum computing due to its theoretical capacity to surpass classical information processing for certain specific application areas. Despite the fact that current quantum computers are hindered by noise and decoherence, there have been successful experimental demonstrations of quantum advantage [1–3] and even recently the first logical quantum processor was announced [4]. However, these achievements have yet to have any practical relevance, leaving the search for useful applications ongoing. Many promising quantum algorithms, such as solving systems of linear equations [5,6], performing data fitting [7], computing scattering cross sections [8,9], pricing financial derivatives [10–17], and in general solving differential equations [18–34], require the efficient loading of classical data into quantum devices. Unfortunately, this step re-

mains a challenging problem, and it is a major bottleneck for the practical application of quantum computation, especially within the emerging field of quantum machine learning in the NISQ-Era [35–37].

In this regard, the main drawback comes from the fact there is no universal loading protocol and each particular case must be carefully studied in order to design bespoke encoding protocols that cater to the specific problem to be solved [38–40]. In this sense, one of the main embedding techniques is the amplitude encoding, which loads the values of a discretized, normalized complex function into the amplitude of the quantum states [41–63]. Several approaches have already been presented in the literature for implementing the amplitude embedding, many requiring a huge (exponential) number of resources (ancillas and gates) [43–47], oracles [48–53], sparsity in the quantum state [54], training variational circuits [55–57], truncating a series expansion [58, 59], use the quantum singular value transformation [41, 42] or matrix product states [60–69].

There exists a necessity for loading polynomial functions due to the growing number of applications of quantum computing. Specifically, in finance [10–17], the ability to efficiently load first order polynomials, $f(j) = aj + b$ allows for options pricing via quantum amplitude estimation (QAE) without coherent arithmetic [70]. In this sense, applying QAE allows us to extract the amplitude $\sum_j f(j)p(j) = \mathbb{E}[f(X)]$ where X is a random variable with probability mass function $p(j)$. This approach can be generalized to the multidimensional setting where the ability to load multivariate linear functions yields efficient algorithms for pricing basket options [71]. Furthermore, quantum circuits for efficiently loading the linear function can be used to construct a block encoding of the identity function thus allowing us to apply the quantum singular value Transformation [72–77] (QSVT) in order to obtain any polynomial amplitude encoding [41,42,78,79]. This is a powerful tool capable of uniformly approximating the encoding of any continuous real function defined on a closed interval with arbitrary precision [80,81].

In this article we present two methods for implementing the amplitude encoding of real valued polynomials into quantum computers with linear complexity. The first one is based on the matrix product sate (MPS) representation of quantum states and its implementation on a quantum computer. We explore how approximating the MPS affects the achieved fidelity and the resources requirements [82–87]. On the other hand, the second method consists of two steps, first we propose a novel protocol to efficiently load the linear function on k_0 qubits based of the discrete Hadamard-Walsh transform (DHWT) [88] that we use to achieve a block encoding of the amplitudes with complexity $\mathcal{O}(k_0)$. Secondly, we use the QSVT to implement a polynomial transformation on the eigenvalues of the block encoding to achieve the desired target state [78,79]. Note that as we have implemented a block encoding of the linear function, our method avoids errors due to the polynomial approximation of the $\arcsin(x)$ as proposed in previous works [41, 42]. Furthermore, we reduce the complexity of the state-of-the-art for encoding polynomials functions via QSVT by introducing a controllable error.

The article is structured as follows, first we review the loading of polynomials via MPS. Next, we introduce our new methodology that combines DHWT to load the linear function with the QSVT to implement the polynomial transformation of the amplitudes. Finally we show numerical results and compare our method with previous results in the literature.

2 Loading of polynomials via MPS

In this section we analyze the methods based on MPSs to encode polynomials into the amplitude of a quantum state according to following definition.

Definition 1 *Let $P(x) : [a, b] \to \mathbb{C}$, be a polynomial with complex coefficients. We define the n-qubit normalized representative state of $P(x)$ as the quantum state $|\Phi_P\rangle = \frac{1}{C_P}\sum_{j=0}^{2^n-1} P(x_j)|j\rangle$, with $x_j = a + j\frac{b-a}{2^n-1}$ and C_P the normalization factor.*

In the particular case of the linear function, i.e. $P(x) = x$, defined on $[0, 2^n - 1]$, we have the normalization factor is $C_P = \sqrt{(2^{n+1}-1)2^n/(6(2^n-1))}$. For now on for simplicity, we will assume that the polynomials we work with are defined on the interval $[0, 1]$, so that might include rescaling of the domain of

the linear function.

The complete description of a quantum state of n linearly connected qubits (sites) can be represented by a tensor, A, with n physical indices. A state of this kind is referred to as a matrix product state (MPS) [82–85]. Each physical index is assigned to a qubit and has a degree of $d = 2$ i.e., the index is either 0 or 1. For a specific choice of each physical index j_i, the tensor's values give rise to a collection of n matrices whose product is equal to the amplitudes of the computational basis state $|j_{n-1} \dots j_0\rangle$. For open boundary conditions, the MPS representation of a quantum state is given by

$$|\Psi\rangle = \sum_{j_0 \cdots j_{n-1}} \sum_{\alpha_1 \cdots \alpha_{n-1}} A^{[1]}_{j_0,\alpha_1} A^{[2]}_{j_1,\alpha_1\alpha_2} \cdots A^{[n]}_{j_{n-1},\alpha_{n-1}} |j_{n-1} \dots j_0\rangle . \quad (1)$$

This representation has $n-2$ tensors of order 3, denoted as $A^{[i]}_{j_{i-1},\alpha_{i-1}\alpha_i}$, $\forall\ i \neq 0, n-1$, and 2 external tensors $A^{[1]}_{j_0,\alpha_1}$ and $A^{[n]}_{j_{n-1},\alpha_{n-1}}$ of order 2, with j_i the physical indexes that range from 0 to $d-1$ and α_i the virtual indices from 0 to $\chi_i - 1$. The virtual dimensions, χ_i, connecting each pair of tensors via the virtual indices are referred as the bond dimensions. We define the bond dimension of the entire MPS as $\chi = \max_i \chi_i$.

When it comes to representing polynomials as quantum states, Grasedyck [60] proved that for a real valued polynomial of degree d encoded in the amplitudes of a quantum state according to Def. 1, the MPS bond dimension is as much $\chi = d+1$.

2.1 Obtaining the exact MPS

The MPS representation of a quantum state $|\Psi\rangle$ is not unique, as different choices of $A^{[i]}_{j_i,\alpha_{i-1}\alpha_i}$ can yield the same quantum state. We focus on the left canonical form, which implies the following conditions

$$\sum_{j_0,\alpha_1} A^{[1]}_{j_0,\alpha_1} A^{[1]\dagger}_{j_0,\alpha_1} = 1, \quad (2)$$

$$\sum_{j_i,\alpha_i} A^{[i]}_{j_i,\alpha_{i-1}\alpha_i} A^{[i]\dagger}_{j_i,\alpha'_{i-1}\alpha_i} = \delta_{\alpha_{i-1}\alpha'_{i-1}}, \quad (3)$$

$$\sum_{j_{n-1}} A^{[n]}_{j_{n-1},\alpha_{n-1}} A^{[n]\dagger}_{j_{n-1},\alpha'_{n-1}} = \delta_{\alpha_{n-1}\alpha'_{n-1}}. \quad (4)$$

To obtain the MPS representation of a quantum state, the singular value decomposition (SVD) is employed [89]. Initially, the quantum state, represented as a tensor A of rank n and dimension 2, is reshaped into a matrix by combining all the indices except one. The SVD is then applied to this matrix, decomposing it into the matrix of left singular vectors, U, the matrix of singular values, Σ, and the matrix of right singular vectors, $V^\dagger$. As we will depict in the following section, it is possible to truncate the smallest singular values by choosing a desired bond dimension χ and keeping only the χ largest values. Then, the matrix of singular values is contracted with the matrix of right singular vectors (left canonical form), and the resulting matrix, $\Sigma V^\dagger$, is reshaped back into a tensor that now has an extra virtual index. This process, shown in Fig. 1, is iterated for each physical index. Finally, we obtain an MPS that approximates the original quantum state while providing a compact and efficient representation.

The computational cost of performing SVD on an $m \times n$ matrix is typically $\mathcal{O}(\min(mn^2, m^2n))$ and must be taken into account as a preprossessing cost in this methodology. This cost can be reduced for sparse or structured low-rank matrices. In the case of an exact matrix product state (MPS), the bond dimension doubles for each connection between core tensors. This leads to a maximum bond dimension of $2^{\lfloor n/2 \rfloor}$, occurring in the middle of the MPS. As a result, the computational cost of the entire algorithm is primarily determined by the SVD of this central square matrix, i.e. $\mathcal{O}(2^{3n/2})$, which exhibits exponential scaling with the number of qubits. Therefore, this non-negligible classical pre-processing cost must be considered in the overall complexity of the MPS algorithm.

In the particular case of the linear function, the analytical expression of the exact MPS [90], which has bond dimension $\chi = 2$, reads

$$|\Phi_L\rangle = \sum_{j_0 \cdots j_{n-1}} \begin{pmatrix} j_0/C & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2j_1/C & 1 \end{pmatrix} \cdots \begin{pmatrix} 1 \\ 2^{n-1} j_n/C \end{pmatrix} |j_{n-1} \dots j_0\rangle . \quad (5)$$

2.2 Approximation of the exact MPS

While, in the worst case scenario, it is possible to exactly represent any quantum state as an MPS by allowing the bond dimensions to grow up to

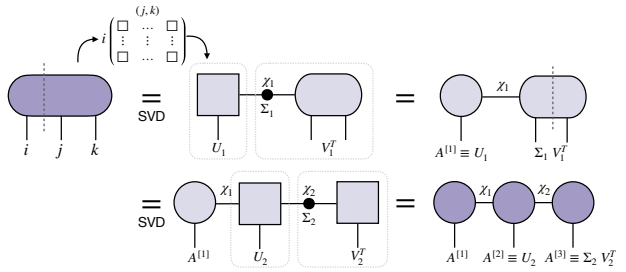

Figure 1: Iterative singular value decomposition (SVD) [89] procedure for obtaining the MPS from a tensor with $n = 3$ physical indices. The process involves $n-1$ uses of the SVD. Notably, the matrices containing the singular values are absorbed to the right, resulting in the left canonical form of the MPS.

$2^{\lfloor n/2 \rfloor}$, we can potentially achieve an exponential compression by approximating the initial tensor using $\mathcal{O}(2n\chi^2)$ elements, given a fixed bond dimension, $\chi_i = \chi \; \forall \; i$. However, as the entanglement of the state to be approximated increases, the minimum bond dimension χ required to get a good description of the state using an approximated MPS representation also grows [91, 92]. Notice that when truncating the maximum bond dimension of the MPS, the complexity of the classical preprocessing becomes $\mathcal{O}(n\chi^3)$.

In order to estimate the error incurred when truncating the bond dimension, it is necessary to consider that during each iteration of the compression protocol we perform a SVD and discard singular values. The error incurred when approximating a matrix by considering its k largest singular values is determined by the Eckart-Young theorem [93], and corresponds to the sum of the omitted singular values. During compression, this rank-k approximation is performed for each core tensor. Thus, the overall error of the MPS approximation can be upper bounded in the Frobenius norm as

$$\|A - \tilde{A}\|_F^2 \leq \sum_{i=1}^{n-1} \left(\sum_{k=\chi_i+1}^{\dim(\Sigma_i)} \sigma_k^2(\Sigma_i) \right), \qquad (6)$$

where $\tilde{A}$ denotes the approximated MPS. This equation encompasses the error contributions from the $n-1$ singular value matrices Σ_i, each characterized by a fixed bond dimension, χ_i. In order to keep the approximation error low, it is crucial that the spectrum of each Σ_i decays rapidly.

In this regard, the state-of-the-art technique for preparing smooth, differentiable, real-valued functions using matrix product states relies on singular values exhibiting exponential decay as demonstrated in Ref. [61]. This allows for good approximations of such a class of functions while maintaining low bond dimension values, thus significantly reducing the required resources to prepare the associated function-encoding quantum state. The method relies on the fact that, for such functions, the entanglement entropy, quantified by the von Neumann entropy, scales logarithmically with the number of qubits, n, and therefore, these functions can be efficiently approximated by an MPS with a low bond dimension, as argued in Ref. [91]. Although empirical results with this technique applied to polynomials show good performance with $\chi = 2$, the upper bound of the von Neumann entropy depends on the maximum derivative value of the function within the considered interval. Consequently, as we discuss later in this work, for certain polynomials the truncation to $\chi = 2$ does not yield a satisfactory approximation.

In the particular case of the linear function, the analytical expression from Eq. 2.1 reveals that achieving the exact MPS representation requires a bond dimension of $\chi = 2$. However, one might consider reducing the bond dimension to 1 to investigate how severely accuracy decays. We explore this possibility by comparing the linear function approximated by MPSs of bond dimension $\chi = 2$ (exact MPS) and $\chi = 1$ (product state) in Section 4.

2.3 From MPS to circuit

Let us now analyze the resources needed to translate an MPS in either cases, exact or approximated, into a quantum circuit [85, 86, 94, 95]. First we will assume that the bond dimensions are powers of two, padding the tensors with zeros if needed. Therefore, we can assume that these tensors are isometries of $2\chi_i \times \chi_{i+1}$ and thereby can be embedded into a unitary gate acting on $n_{\chi_i} = \max\{\log_2(2\chi_i), \log_2(\chi_{i+1})\}$ qubits. The factor of two arises from the two possible values that each of the physical indices can take. The arrangement of gates in the MPS conversion process, following a linear topology, results in the formation of a single layer of multi-qubit unitaries, whose sizes n_{χ_i} depends on the associated bond dimensions. These unitaries are organized in a staircase topology, commonly referred

to as a linear circuit layer [94]. Therefore the complexity is equivalent to implementing a cascade of $n-1$ multi-qubit unitaries. In the case $\chi = 2$, this complexity scales as $\mathcal{O}(n)$ two-qubit unitaries. On top of this, the cost of decomposing these unitaries into two-qubit gates must be taken into account. This cost is considerably larger when decomposing multi-qubit unitaries and might result in an exponential number of two-qubit gates. [96,97]. Additionally, one might consider implementing circuits that approximate the exact multi-qubit unitaries [94]. Lastly, in Ref. [87] authors proposed a method for loading translation-invariant short-correlation MPS with an error ϵ in depth $\mathcal{O}(\log(N/\epsilon))$ thereby establishing a class of MPSs that admit efficient circuit implementations.

We can conclude that even though MPSs encoding polynomials admit an analytical construction, we have to consider the cost of obtaining the SVD and the cost of the decomposition of the associated unitaries. These costs can be significantly reduced by truncating down to a bond dimension of $\chi = 2$, although this introduces an uncontrollable source of error [41,61,62,98]. Additionally, one can also consider the possibility of approximating the multi-qubit unitaries of a MPS's circuit representation [94], which could result in high fidelity states in some cases although there is not a priori guarantee of success.

3 Efficient loading of polynomials via DHWT and QSVT

In this section we present a method for loading polynomials by combining a technique to encode the linear function into the amplitudes of a quantum state via the discrete Hadamard-Walsh transform (DHWT) with the quantum Singular Value Transformation (QSVT) algorithm. The first methodology encodes the linear function with a shallow sequence of multi-controlled gates that loads its Hadamard-Walsh series expansion, followed by the inverse discrete Hadamard-Walsh transform. By truncating the series expansion, we demonstrate that this approach allows for a controllable approximation of the state representing the linear function up to a certain error. We analyze this error in terms of infidelity, ϵ with respect to the exact state and through the deviation $\delta_j^{(2)}$ in each amplitude from the exact amplitudes.

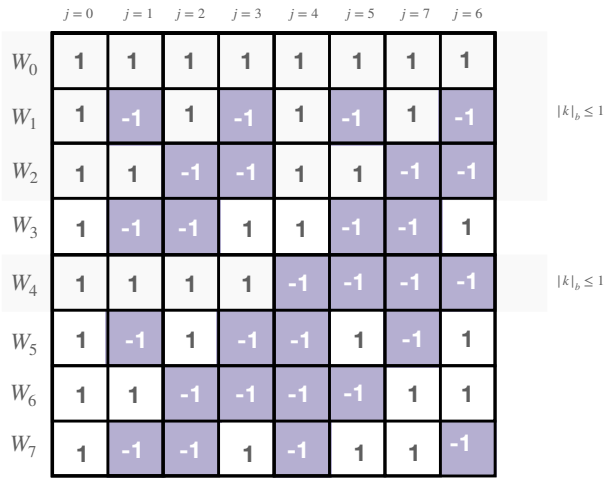

Figure 2: Discrete Hadamard-Walsh Transform for $n = 3$ qubits in the natural order representation.

The second step leverages the results of [78,79] to generate a block encoding of the linear function. This involves using the unitary to load the linear function into amplitudes of a k_0-qubit quantum state. Then, the Quantum Singular Value Transformation (QSVT) is applied to implement a polynomial transformation, $P(x)$, on the amplitudes of the quantum state corresponding to the identity function. This results into a unitary block encoding of the polynomial that applied to an adequate initial state yields an encoding of the desired polynomial. Finally, we pad the remaining $n-k_0$ qubits, obtaining an approximated encoding.

3.1 The Discrete Hadamard-Walsh transform

Definition 2 *The discrete Hadamard-Walsh transform (DHWT) is a linear, orthogonal and symmetric operation that transforms discrete signals or sequence of sorted data into a new representation given by the Hadamard-Walsh Series*

$$HWT : (z_n^{(0)} \dots z_n^{(N-1)}) \to (x_n^{(0)} \dots x_n^{(N-1)}), \quad (7)$$

$$x_n^{(k)} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} z_n^{(j)} W_k(j) \quad (8)$$

where $j = \sum_{m=0}^{n-1} j_m 2^m$, $k = \sum_{m=0}^{n-1} k_m 2^m$ *with* $N = 2^n$, $j_m, k_m \in \{0,1\}$ *and* $W_k(j) = (-1)^{\sum_{m=0}^{n-1} j_m k_m}$ *is the k-th Walsh function, where we have used the natural order.*

We also define the binary norm of an integer as $|k|_b = \sum_{m=0}^{n-1} k_m$. Note that when $|k|_b = 1$, it is

equivalent to say that k is a power of 2. Additionally, when representing a quantum state $|j\rangle$ in terms of a binary notation we will denote the state as $|j_{n-1}\ \dots\ j_0\rangle$, taking the order of the tensor product from right to left.

Lemma 1 *Let* $(0, 1 \dots, 2^n - 1)$ *be the discrete sorted input sequence. Then, the coefficients of its Hadamard-Walsh series are given by*

$$x_n^{(k)} = \begin{cases} 2^{n/2}(2^n - 1)/2 & \text{if } k = 0 \\ -2^{n/2}k/2 & \text{if } |k|_b = 1 \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Note that in general, the sparsity of the state encoding the Hadamard-Walsh series of a polynomial of degree d represented in n qubits, with $d \leq n$, is $s = \sum_{k=0}^{d} \binom{n}{k}$ [99]. Therefore, one might consider the techniques in Ref. [54] to implement the state with depth $\mathcal{O}(\log(ns))$ and $\mathcal{O}(ns\log(s))$ ancillary qubits.

3.2 Exact loading of the linear function via DHWT

In the particular case of the linear function, the target state is $|\Phi_L\rangle_n = \frac{1}{C_n}\sum_{j=0}^{2^n-1} j\,|j\rangle$, with normalization constant $C_n = \sqrt{(2^{n+1}-1)(2^n-1)2^n/6}$. Notice that Lemma 1 provides the state that encodes the discrete Hadamard-Walsh transform of the coefficients of $|\Phi_L\rangle_n$ which we can write as

$$\left|\tilde{\Phi}_L\right\rangle_n = \frac{1}{\tilde{C}_n}\sum_{|k|_b \leq 1} x_n^{(k)}\,|k\rangle\,,$$

with $\tilde{C}_n$ the corresponding normalization factor. Note that the state above is a $|1\rangle$-sparse quantum state, i.e. the bit string representations of the basis states whose superposition conforms the state have at most one $|1\rangle$, i.e. $\{|00\ \dots 00\rangle\,,\ |10\ \dots 0\rangle \dots, |00\ \dots 01\rangle\}$.

Due to its structure, the state $\left|\tilde{\Phi}_L\right\rangle_n$ can be efficiently encoded into a quantum computer with gate complexity $\mathcal{O}(n)$ according to the circuit depicted in Fig. 3, where the angle of every multi-controlled rotation is given by $\theta_k = \arcsin\left(2x_n^{(2^k)}/\tilde{C}_n G_n(k)\right)$, with $k = 0,\ \dots,\ n-1$ and $G_n(k) = \prod_{i=k+1}^{n-1}\cos(\theta_i/2)$ if $k < n-1$ and $G_n(k) = 1$ if $k = n-1$. We denote the unitary corresponding to this circuit as $U_{L,n}$.

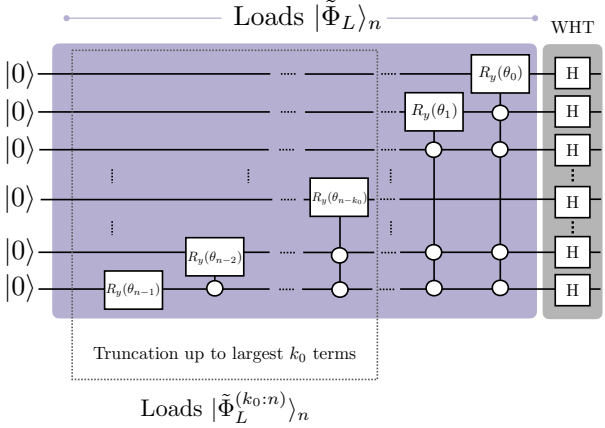

Figure 3: Circuit implementation for loading the quantum state $|\Phi_L\rangle_n$, denoted as $U_{L,n}$ with complexity $\mathcal{O}(n)$ multi-controlled gates. The angle of every multi-controlled rotation is given by $\theta_k = \arcsin\left(2x_n^{(2^k)}/\tilde{C}_n G_n(k)\right)$, with $k = 0\ \dots\ n-1$ and $G_n(k) = \prod_{i=k+1}^{n-1}\cos(\theta_i/2)$ if $k < n-1$ and $G_n(k) = 1$ if $k = n-1$. The first part of the circuit loads the state encoding its Hadamard-Walsh series $\left|\tilde{\Phi}_L\right\rangle_n$ and once it has been loaded, we apply the Hadamard-Walsh transform to achieve $|\Phi_L\rangle_n$. If we consider only the first k_0 rotations, then we get the approximated Hadamard-Walsh series $\left|\tilde{\Phi}_L^{(k_0:n)}\right\rangle_n$ and the respectively approximated state $\left|\Phi_L^{(k_0:n)}\right\rangle_n$. With this truncation the angles change its value to $\theta_k^{k_0} = \arcsin(2x_n^{(2^k)}/\tilde{C}_n^{(k_0:n)}G_n^{(k_0:n)}(k))$, with $\tilde{C}_n^{(k_0:n)}$ the new normalization factor and $G_n^{(k_0:n)}(k) = \prod_{i=k+1}^{n-1}\cos(\theta_i^{k_0}/2)$ if $n-k_0 \leq k < n-1$ and $G_n^{(k_0:n)}(k) = 1$ if $k = n-1$ and the complexity is reduced to $\mathcal{O}(k_0)$. We denote this last circuit that loads the approximated state as $U_{L,n}^{(k_0:n)}$. See the appendix in Ref. [55] for the details of the decomposition of multi-controlled gates.

Once that we have encoded the state $\left|\tilde{\Phi}_L\right\rangle_n$ that represents the discrete Hadamard-Walsh series of our target state, we can simply uncompute the Hadamard-Walsh transform to obtain $|\Phi_L\rangle_n$. In terms of gates on a quantum computer, this operation is a parallel implementation of Hadamard gates on all the qubits.

Additionally, we would like to remark that due to normalization, any linear function with an arbitrary slope and 0 offset will lead to the same encoding. However, it is possible to vary the slope by changing the offset of the linear function under the restriction of the quantum state to be normalized.

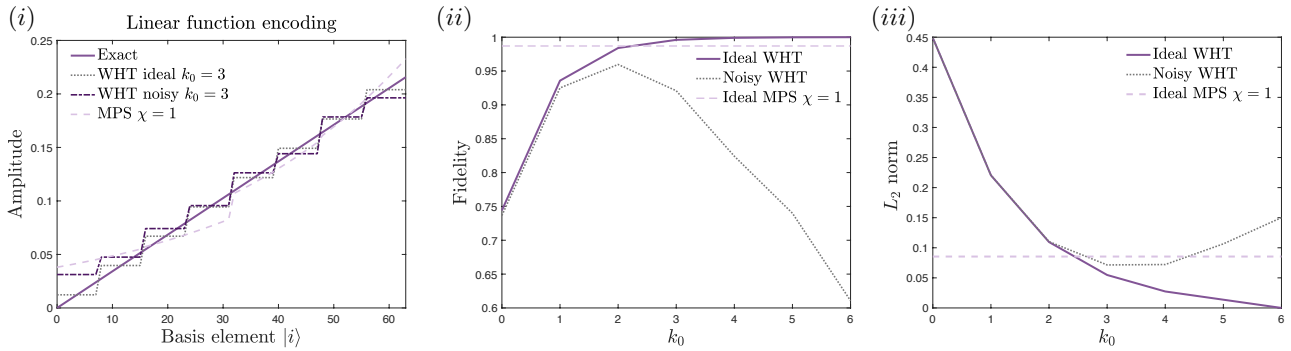

Figure 4: Comparison of different methods for loading the linear function on $n = 6$ qubits. (i) Illustrates the resulting state by using different encoding protocols, DHWT with $k_0 = 3$ and MPS $\chi = 1$, in noisy an ideal scenarios. The choice of $k_0 = 3$ is due to the fact this case has a high fidelity and the minimum error measured in the L_2 norm for DHWT method in presence of noise. This is shown in (ii) and (iii), where fidelity and error measured in the L_2 norm for both the ideal and noisy DHWT methods are plotted with respect to k_0. We have not considered noise for the case MPS $\chi = 1$ as its loading circuit is only a layer of single qubit rotations, and its impact is marginal.

3.3 Approximated loading of linear polynomials via DHWT

In the preceding section, we have conclusively demonstrated the efficient loading of linear functions. Now, the inevitable query arises: can the Hadamard-Walsh series be deliberately truncated while maintaining control over the process? In other words, is it possible to strike a balance between the number of terms truncated and the resulting error, thereby achieving a quantum state approximation that accurately encodes our intended target?

We now proceed to illustrate how the loading of the linear function can be approximated by truncating the Hadamard-Walsh series. Let us assume that we have the DHWT for n qubits, then the non zero coefficients are

$$\vec{h}_n = (x_n^{(0)}\ x_n^{(1)}\ x_n^{(2)}\ x_n^{(4)}\ \dots\ x_n^{(2^{n-1})}) \qquad (10)$$

with $|x_n^{(1)}| < |x_n^{(2)}| < \dots < |x_n^{(2^{n-1})}| < |x_n^{(0)}|$. We keep x_0 and the largest k_0 values of the coefficients with $|k|_b = 1$, i.e. $\vec{h}_n^{(k_0:n)} = (x_n^{(0)}\ 0\ 0 \dots 0\ x_n^{(2^{n-k_0})}\ \dots\ x_n^{(2^{n-1})})$. Next, we construct the circuit to generate the state encoding these renormalized coefficients of the truncated series. Then, the fidelity of the resulting state, $|\Phi_L^{(k_0:n)}\rangle_n$, with the exact state, $|\Phi_L\rangle_n$, is given by

$$F = \frac{\frac{1}{4}\left(-1+2^n\right)^2 + \frac{1}{3}2^{-2+2n}\left(1-2^{-2k_0}\right)}{\frac{1}{4}\left(-1+2^n\right)^2 + \frac{1}{3}2^{-2+2n}\left(1-2^{-2n}\right)}. \qquad (11)$$

More details about how to derive this expression are given in the Appendix A.

Note that while the structure of the circuit is the same, see Fig. 3, the angles have changed its value to $\theta_k^{k_0} = \arcsin(2x_n^{(2^k)}/\tilde{C}_n^{(k_0:n)} G_n^{(k_0:n)}(k))$, with $\tilde{C}_n^{(k_0:n)}$ the new normalization factor and $G_n^{(k_0:n)}(k) = \prod_{i=k+1}^{n-1}\cos(\theta_i^{k_0}/2)$ if $n - k_0 \leq k < n-1$ and $G_n^{(k_0:n)}(k) = 1$ if $k = n-1$.

Now, assuming an infidelity $\epsilon = 1 - F$, it is possible to obtain the expression

$$k_0 = -\frac{1}{2}\log_2\left[\frac{1}{2^{2n}} + \epsilon\left(4 + \frac{2(1-3\cdot 2^n)}{2^{2n}}\right)\right] \qquad (12)$$

which establishes the trade off between infidelity and truncation. In the asymptotic limit $n \to \infty$, we obtain $k_0 = \frac{1}{2}\log_2\left[1/(4\epsilon)\right]$.

Additionally to this analysis, we also study the deviation of the amplitudes of the approximated state, $\left|\Phi_L^{(k_0:n)}\right\rangle_n$ with respect to the ideal state, $|\Phi_L\rangle_n$. We now focus on describing the state $\left|\Phi_L^{(k_0:n)}\right\rangle_n$ that comes from the truncation of the series, denoted as $\vec{h}_n^{(k_0:n)} = (x_n^{(0)}\ 0\ 0 \dots 0\ x_n^{(2^{n-k_0})}\ \dots\ x_n^{(2^n-1)})$. We compare the k_0 qubit state obtained from $\vec{h}_{k_0}^{(k_0:n)} = (x_n^{(0)}\ x_n^{(2^{n-k_0})}\ \dots\ x_n^{(2^{n-1})})$ with the state produced by the series corresponding on doing the DHWT to the linear function encoded on k_0 qubits given by $\vec{h}_{k_0} = (x_{k_0}\ x_{k_0}^{(0)}\ x_{k_0}^{(2^0)}\ \dots\ x_{k_0}^{(2^{k_0-1})})$. From this comparison we can obtain that

$$\alpha := x_n^{(2^{n-k_0+j})}/x_{k_0}^{(2^j)} = 2^{3/2(n-k_0)}$$

$$\forall \;\; j = 0,\; ...,\; k_0 - 1.$$

Therefore we can write

$$\vec{h}_{k_0}^{(k_0:n)} = \alpha(x_{k_0}^{(0)}\; x_{k_0}^{(2^0)}\; \ldots\; x_{k_0}^{(2^{k_0-1})}) + \beta(2_{k_0/2}^{(0)}\; 0\; \ldots\; 0) \tag{13}$$

with $\beta = 2^{(n-k_0)/2-1}(2^{n-k_0} - 1)$ the offset that will induce the maximum deviation in the amplitude between both the ideal and approximated quantum state. Thus, we can conclude that when truncating the sate, we are loading the k_0-qubit state $\left|\Phi_{L,k_0}^{(k_0:n)}\right\rangle = \frac{1}{C_{k_0}^{(k_0:n)}} \sum_{j=0}^{2^{k_0}-1}(\alpha j + \beta)\,|j\rangle$ on the most significant k_0 qubits, with $C_{k_0}^{(k_0:n)}$ the corresponding normalization factor. When adding the remaining $n-k_0$ qubits, this will lead to an state with a degeneracy 2^{n-k_0} on every of this amplitudes (graphically a step wise function)

$$\left|\Phi_{L}^{(k_0:n)}\right\rangle_n = \frac{1}{C_n^{(k_0:n)}} \sum_{j=0}^{2^{k_0}-1} \sum_{l=0}^{2^{n-k_0}-1} (\alpha j + \beta)\,|j\rangle\,|l\rangle$$

with

$$C_n^{(k_0:n)} = 2^{n/2}\left[\frac{\alpha^2}{6}(2^{k_0+1}-1)(2^{k_0}-1) + \alpha\beta(2^{k_0}-1) + \beta^2\right]^{1/2}. \tag{14}$$

Finally, merging both sums we finally get

$$\left|\Phi_{L}^{(k_0:n)}\right\rangle_n = \frac{1}{C_n^{(k_0:n)}} \sum_{j=0}^{2^n-1} (\alpha \lfloor j/2^{n-k_0} \rfloor + \beta)\,|j\rangle\,. \tag{15}$$

From these expression we can define the deviation of the amplitude of $|j\rangle$ as

$$\delta_j^{(2)} := \left| \frac{1}{C_n^{(k_0:n)}}(\alpha \lfloor j/2^{n-k_0} \rfloor + \beta) - \frac{j}{C_n} \right| \leq \beta / C_n^{(k_0:n)} \tag{16}$$

In the asymptotic limit the upper bound of $\delta_j^{(2)}$ given by $\beta/C_n^{(k_0:n)}$ decays as $\mathcal{O}(\frac{1}{\sqrt{2^n}})$ with a dominant coefficient given by

$$\frac{1}{2^{3k_0/2+1}\left(\frac{(2^{k_0}-1)(2^{k_0}-1)}{6\cdot 2^{3k_0}} + \frac{2^{k_0}-1}{2^{3k_0+1}} + \frac{1}{2^{3k_0+2}}\right)^{1/2}}, \tag{17}$$

which converges to 0 when k_0 tends to infinity as $\mathcal{O}(\frac{1}{2^{k_0}})$. We depict the complexities of loading the exact and approximated versions of linear function either via MPS or DWHT in Tab. 1.

Alternative to this set up, one can also consider a MPS based methodology to load the linear function, which in the exact case scales linearly on the number of qubits, see Section 2.3. Thus, from now on, we will assume that we have access to a state preparation oracle of the exact linear function on k_0 qubits denoted as U_{L,k_0}, for $1 \leq k_0 \leq n$, along with its adjoint and their controlled variants. This allows us to load the linear function either in an exact manner with a circuit depth that scales $\mathcal{O}(k_0)$. Note that in the worst case scenario, the controlled version can be achieved by controlling every gate of the oracle, which keeps the complexity invariant.

3.4 Polynomial transformation of amplitudes via quantum singular value transformation

Once we have introduced the quantum circuit that loads the linear function into k_0 qubits via the unitary operator U_{L,k_0} with a complexity of $\mathcal{O}(k_0)$, our method uses the quantum singular value transformation (QSVT) [72–77] to achieve the polynomial transformation of the amplitudes. One also could consider the unitary to load the MPS of the linear function with $\chi = 1$ on k_0 qubits, although the final results are considerable worst as we show later.

In this work we follow the procedure detailed in Ref. [78] and in its subsequent exponential improvement in Ref. [79], which provides a generalization that presents a diagonal block encoding and introduces the importance sampling. The remarkable insight of these works lies in the authors' demonstration of how it is possible to use the QSVT polynomial transformation to explicitly construct quantum circuits that apply the polynomial transformation to the amplitudes of any quantum state of interest, provided that the

qubits \ method	**Exact**	**Aprx**
k_0	$\mathcal{O}(k_0)$	-
n	$\mathcal{O}(n)$	$\mathcal{O}(k_0(\epsilon))$

Table 1: Complexities for loading the linear function into a different number of qubits k_0 and n, with $k_0 < n$, either in an exact or approximated form. $k_0(\epsilon)$ corresponds to Eq. (12).

Parameter	Description	Value
SQG time	Single qubit gate time (ns)	35
CX time	CX gate time (ns)	540
rD	Deviation ratio for the single qubit gates	2,457E-04
Pbf	Bit-flip error during the rz gate	2,457E-04
CNOTerror	Deviation ratio for CX gate	8,328E-03
pmeas	Readout error	2,23E-01
pth	Thermal population of the ground state	0.01
T1	Decoherence time (us)	214.84
T2	Dephasing time (us)	214.84

Table 2: Noise parameters description and their value. We have estimated the numerical values from the calibration data provided for the IBM device 'ibm_jakarta'.

encoding unitary is known.

The steps needed to achieve the state amplitude encoding of the polynomial are: the block encoding of the amplitudes, the polynomial transformation of the block encoding and the quantum amplitude amplification algorithm.

3.4.1 Block encoding

The first step is the block encoding of the amplitudes of the linear function given the unitary U_{L,k_0} that prepares the state that represents the linear function.

Definition 3 *[42, 72–77, 100] Let be A a k_0 qubit operator, $\alpha, \varepsilon \in \mathbb{R}^+$ and $a \in \mathbb{N}$. We say that the $(a+k_0)$-qubit unitary U_A is an (α, a, ε)-block encoding of A if*

$$\|A - \alpha(\langle 0|^{\otimes a} \otimes I) U_A (|0\rangle^{\otimes a} \otimes I)\|_2 \leq \varepsilon. \quad (18)$$

where α is the proportionality constant, a the number of ancilla qubits required and ϵ the error achieved measured with $||A||_2 = \sigma_{\max}(A)$.

According to Theorem 2 in Ref. [79], given a k_0-qubit state $|\psi\rangle = \sum_{j=0}^{2^{k_0}-1} \psi_j |j\rangle$ with real amplitudes loaded by a unitary U, i.e. $U|0\rangle^{\otimes n} = \sum_{j=0}^{2^{k_0}-1} \psi_j |j\rangle$, $\psi_j \in \mathbb{R}$, it is possible to prepare a $(1,\ k_0+3,\ 0)$ block encoding U_A of $A = \sum_{j=0}^{2^{k_0}-1} \psi_j |j\rangle\langle j|$ with $\mathcal{O}(k_0)$ circuit depth and $\mathcal{O}(1)$ queries to a controlled version of U. Note that if the amplitudes are real the number of ancillary qubits required for the block encoding is $k_0 + 2$.

Here we use this result to create from U_{L,k_0} the unitary $U_{A_{L,k_0}}$ which is a $(1,\ k_0+2,\ 0)$ block encoding of the Hermitian operator $A_{L,k_0} = \frac{1}{C_{k_0}} \sum_{j=0}^{2^{k_0}-1} j |j\rangle\langle j|$ that encodes the amplitudes of $|\Phi_L\rangle_{k_0}$ with $\mathcal{O}(k_0)$ circuit depth, see Appendix C for further details.

An alternative $(1, 1, 0)$-block encoding U_B resulting by following the idea of Ref. [41] could be implemented by applying the unitary dilation technique [101] to $B = \frac{1}{2_0^k - 1} \sum_{j=0}^{2_0^k-1} j |j\rangle\langle j|$, given that $\|B\| \leq 1$. This operation would require an efficient simulation of the Hamiltonian $H = \arccos(B)$ [102]. Discussions of more possible block encoding for the linear function are shown in Appendix C.

3.4.2 Polynomial transformation of the block encoding

Once that we have obtained the amplitude block encoding, we show how to implement polynomial transformations of complex amplitudes via the Quantum Singular Value Transformation (QSVT) [41, 42, 75, 78, 79]. The construction and efficiency of explicit quantum circuits for applying the polynomial transformation to amplitudes of a k_0-qubit quantum state rely on the complexity of implementing the block encoding of the amplitudes, denoted as U_{L,k_0}. It is crucial that the complexity of U_{L,k_0} remains of the order $\mathcal{O}(k_0)$ to prevent it from dominating the overall algorithmic cost.

Lemma 2 *(Lemma 7 in Ref. [79]) Let $\gamma > 0$. Given a $(1,\ k_0+2,\ 0)$ block-encoding U_A of $A = \sum_{j=0}^{2^{k_0}-1} \psi_j |j\rangle\langle j|$ and $P(x)$ a polynomial with complex coefficients of degree d, such that $|P(x)| \leq 1/4\ \forall x$. Then, it is possible to obtain a $(1,\ k_0+5,\ \delta)$ block encoding $U_{P(A)}$ of $P(A)$ with $\mathcal{O}(dk_0)$ circuit depth and $\mathcal{O}(d)$ calls to the con-*

trolled version of U_A and $U_A^\dagger$. The circuit (rotation angles) can be computed with classical time complexity of $\mathcal{O}(poly(d, \log(1/\gamma))$ [77, 104, 105].

By applying Lemma 2 with a polynomial $P(x)$ to U_{L,k_0} one obtains a block encoding U_{P,k_0} of $P(A_{L,k_0})$. Additionally, note that due to the normalization, the values of the amplitudes of the state encoding the linear function have been renormalized to the interval $[0, (2^{k_0} - 1)/\sqrt{(2^{k_0+1} - 1)(2^{k_0} - 1)2^{k_0}/6}]$. Therefore, to standardize the application of polynomials to the interval [0,1], we must compose the polynomial with the rescaling transformation, which keeps the degree of the polynomial invariant. For simplicity's sake, we will ignore this transformation from now on.

3.4.3 Amplitude amplification

We now apply the unitary $U_{P(A)}$ to the initial state $|+\rangle^{\otimes k_0} |0\rangle^{\otimes k_0+5}$. This results into a quantum state $\frac{1}{4\sqrt{2^{k_0}}||P||_\infty} \sum_j P(j/C_{k_0})|j\rangle |0\rangle^{\otimes k_0+5} + \ldots$. The success probability of measuring $|\Phi_P\rangle_{k_0} = \frac{1}{4\sqrt{2^{k_0}}||P||_\infty} \sum_j P(j/C_{k_0})|j\rangle |0\rangle^{\otimes k_0+5}$ is given by $0.0625\mathcal{F}_P^2$, with $\mathcal{F}_P$ being the filling ratio defined as

$$\mathcal{F} = \frac{||P||_2}{\sqrt{2^{k_0}}||P||_\infty}, \tag{19}$$

and $||P||_\infty = \max_{x\in[0,1]}|P(x)|$. Therefore the quantum amplitude amplification algorithm would require $\mathcal{O}(4/\mathcal{F})$ rounds of the oracle that prepares the state to boost the success probability to 1 [41].

Note that the complexity of this protocol depends on the polynomial encoding that we aim to achieve. In this sense, some particular transformations will head to an efficient circuit, while others will introduce an exponential consumption of resources, depending on the filling ratio $\mathcal{F}$.

Once that we have obtained the state $|\Phi_P\rangle_{k_0}$ we can tensor it with the state $|+\rangle^{n-k_0}$ to obtain a quantum state $\left|\Phi_P^{(k_0:n)}\right\rangle_n = |\Phi_P\rangle_{k_0} \otimes |+\rangle^{n-k_0}$ that approximately encodes the polynomial P on n qubits.

Before concluding this section, we would like to remark that for the particular case the polynomial transformation satisfies $P(0) = 0$, one can use the importance sampling technique recently presented in Ref. [79] (Theorem 3) to achieve an exponential enhancement in the overall complexity do to the improvement in the step that encodes the transformed amplitudes into the final state.

3.5 Amplitude deviation in the polynomial encoding

Finally, once the polynomial transformation has been achieved, we can analyze how the error coming from the approximation of loading the exact k_0 qubit polynomial into n qubits. In order to set a proper comparison with Ref. [41, 42], we compute the error in the same way,

$$\delta = \max_x \left\| \frac{P_{\text{exact}}(x)}{||P_{\text{exact}}(x)||_\infty} - \frac{P_{\text{aprx}}(x)}{||P_{\text{aprx}}(x)||_\infty} \right\|_\infty \tag{20}$$

for our particular case we assume $||P_{\text{exact}}(x)||_\infty = M$ and therefore

$$\begin{aligned} \delta &= \max_x \frac{1}{M} \left\| P_{\text{exact}}(x) - P_{\text{exact}}(x \pm \frac{2^{n-k_0}-1}{2^n-1}) \right\|_\infty \\ &\leq \max_x \frac{|P'(x)| \frac{2^{n-k_0}-1}{2^n-1}}{M} \end{aligned} \tag{21}$$

If we now assume $P(x) = \sum_{k=0}^d c_k x^k$, then

$$\delta_\infty \leq \frac{\frac{d^2-d}{2} \frac{2^{n-k_0}-1}{2^n-1}}{M} D \tag{22}$$

with $D = \max_k\{|c_k|\}$. In the asymptotic limit this upper bound decays as $\mathcal{O}(\frac{1}{2^{k_0}})$.

Note that one might think that it can be much cheaper to prepare an appropriate product state and then try to achieve an approximation to the desired polynomial encoding by applying a polynomial function to the product state via QSVT. However, since the complexity of the block encoding is not dominated by the oracle that loads the linear function, a reduction in complexity would not be achieved. Additionally, due to the approximation of MPS,this leads to a non-uniform grid discretization would result in a higher overall error when transforming the amplitudes.

4 Numerical Results

Having established the theoretical framework, in this section we present numerical simulations

Method	$\mathcal{F}$	L_2 norm	Fidelity
DHWT ($k_0 = 1$) + QSVT	0.8164	0.1506	0.0753
DHWT ($k_0 = 2$) + QSVT	0.6864	0.1486	0.0858
DHWT ($k_0 = 3$) + QSVT	0.6331	0.0828	0.6095
Direct Pol MPS ($\chi = 1$)	-	0.0752	0.6712
Lin MPS ($\chi = 1$) + QSVT	-	0.0701	0.7099
DHWT ($k_0 = 4$) + QSVT	0.6190	0.0370	0.9144
Direct Pol MPS ($\chi = 2$)	-	0.0252	0.9597
DHWT ($k_0 = 5$) + QSVT	0.6183	0.0141	0.9874
Direct Pol MPS ($\chi = 3$)	-	0.0049	0.9985
DHWT ($k_0 = 6$) + QSVT	0.6184	0	1
Direct Pol MPS ($\chi = 4$)	-	0	1

Table 3: Error in L_2 Norm (fourth column) and Fidelities (fifth column) for Different Loading Methods of $P(x) = \frac{1}{C_p}(x - 1/(2^n - 1))(x - 20/(2^n - 1))(x - 50/(2^n - 1))(x - 60/(2^n - 1))$ defined on $[0, 1]$. Methods are arranged in ascending order based on fidelity of the polynomial encoding (fifth column). We also show the fidelity of loading the linear functions for those 2 steps methods (second column) and the filling ratio $\mathcal{F}$ (third column) for the different k_0 values.

comparing the methods outlined in this paper. The primary objective is to provide empirical support for our analytical findings.

4.1 Linear function

We begin our analysis by evaluating the performance of the exact and approximated loading of the linear function and its its robustness in terms of fidelity against noise.

When performing noisy simulations, the quantum circuit is transpiled to a native set of gates, including *CNOT, Id, Rz* (θ), *X*, and *Sx*. We consider various noise quantum channels, namely, bit-flip (Pbf), amplitude-damping (T1), dephasing (T2), gate errors (rD, CNOT error), and measurement error (pmeas). The specific noise parameters used in the simulations are summarized in Tab. 2.

The analysis of loading the linear function is depicted in Fig. 4, where we have considered $n = 6$ qubits. In Fig. 4 (i) we have depicted a comparison of the ideal case $\chi = 1$ for the MPS technique, the noisy and ideal cases with $k_0 = 3$ for the DHWT, and the exact encoding. In Fig. 4 (ii) and (iii), we observe a trade-off between the truncation error and the experimental error for different values of k_0 when using the DHWT technique. The two figures of merit considered are the fidelity and the L_2 norm, which is defined as $\|\vec{v}\|_2 = \left(\frac{1}{2^n}\sum|v_i|^2\right)^{1/2}$, where $\vec{v}$ corresponds to the difference between the approximation and the ideal quantum state. We observe that the highest fidelity and the smallest L_2 norm error are achieved at truncation levels $k_0 = 2$ and $k_0 = 3$. In the comparison, we also include the ideal MPS with $\chi = 1$.

4.2 Example of polynomial function

We present a second analysis where we consider the encoding of a polynomial function. In Fig. 5, we present the loading of the polynomial function

$$P(x) = \frac{1}{C_p}(x - 1/(2^n - 1))(x - 20/(2^n - 1))$$

$$(x - 50/(2^n - 1))(x - 60/(2^n - 1)),$$

with C_p the normalization factor, by using $n = 6$ qubits and the two methods studied in this paper. Our approach involves two distinct loading strategies: first, loading the linear function and then applying the polynomial transformation with the QSVT method, ensuring no induced errors (Figs. 5 (*i*) and (*ii*)), and second, using the matrix product state (MPS) representation of the polynomial itself (Fig. 5 (*iii*)). For additional examples see Appendix D.

In order to load the linear function, we utilize either the DHWT method introduced in this work for different truncation values k_0, or the MPS approach with $\chi = 1$. In Tab. 6 we show the L_2 norm and fidelity of the final state with respect to the exact state for each methodology. We observe that for $k_0 \geq 5$, the value of the fidelity

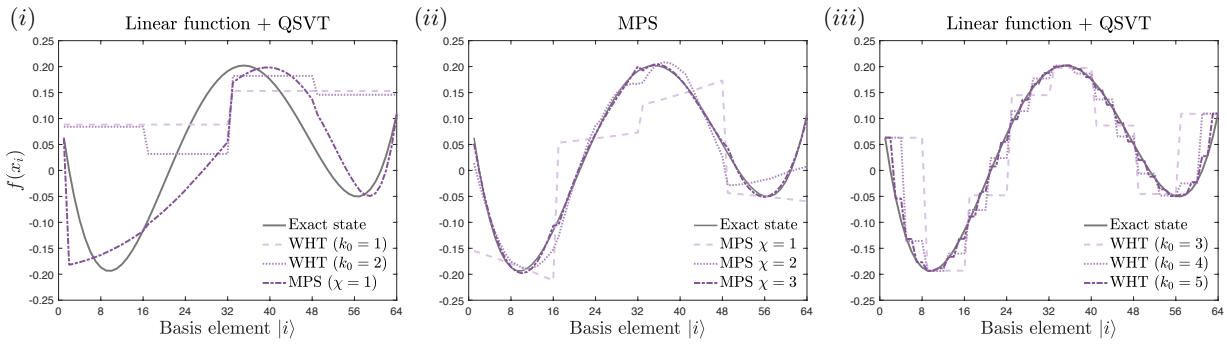

Figure 5: Different methods to load the polynomial function $P(x) = \frac{1}{C_p}(x - 1/(2^n - 1))(x - 20/(2^n - 1))(x - 50/(2^n - 1))(x - 60/(2^n - 1))$ for $n = 6$ qubits using different methods. For this particular example the filling ratio for $k_0 = 6$ is $\mathcal{F} = 0.6184$. The worst and best implementation methods in which the linear function is loaded first, followed by the application of QSVT are displayed in (i) and (iii), respectively. In (ii), results for the loading using MPS are shown, achieving perfect loading with $\chi = 4$, although theory predicts $\chi = 5$ as the upper bound. The L_2 norm and fidelities for each case are displayed in Tab. 6.

achieved by the combined protocol is better than the direct polynomial MPS technique with $\chi = 2$.

Additionally, the fidelities resulting from combining the approximate loading of the linear function with the QSVT to perform the polynomial transformation are presented. Remarkably, the loading of the linear function via a $\chi = 1$ MPS achieves a fidelity of $F = 0.9885$. This suggests that the quantum state of the linear function is nearly a product state. To gain deeper insights, we performed an analysis of the single-qubit rotations that generate this approximate state. We fitted the angles of the rotations to an analytical expression and leveraging this fitting information, we trained a variational circuit aimed at preparing a product state that optimizes the fidelity with respect to the exact linear function. It is crucial to emphasize that the effectiveness of the fitting is not guaranteed across different number of qubits, hence we undertook this approach as a validation method. For additional details, please refer to the Appendix B.

5 Comparison with other methods

The first statement we would like to highlight is that we have put large part of our efforts in achieving an approximate loading of functions as simple as the linear function, by introducing a controllable error that reduces the depth of an already efficient circuit. As far as our knowledge is, there is no previous result in the literature that can do the same task with a comparable performance. That being said, we proceed to compare our method with similar results.

The QSVT enables the application of polynomial transformations to the amplitudes of a quantum state. However, when utilizing this technique to load a polynomial function encoded in the quantum state's amplitudes, the efficient loading of the linear function is crucial. Previously in literature, authors in Ref. [41, 42] explored the possibility of applying the QSVT to block encoding of the sine function. Our method posses the same advantages of using QSVT that they mention: 'we avoid discretizing the values the function can take, providing instead a continuous approximation to the function. Our method is straightforward and versatile, as the same circuit template can be used for a wide range of functions. In contrast, our method avoids the error that propagates into the final loaded state due to the polynomial approximation of the arcsin function by efficiently loading the block encoding of linear function, and the subsequent polynomial transformations applied to load the desired polynomials into the amplitudes. In this context, our approach focuses on implementing the block encoding of the linear function rather than the sinusoidal function. To achieve this, we have proposed a method based on the Hadamard-Walsh Transform. The cost of this replacement in the block encoding can me mainly expressed in terms of adding $n + 1$ extra ancillary qubits to the circuit, see Tab. 4 for comparison. From this table we can observe that or the encoding of a polynomial without error, our algorithm scales with the same complexity as [41, 42] with the er-

	Sine encoding [41, 42]	**This work QSVT**	**MPS**
Block encoding	1+2 Ancillas $\mathcal{O}(n)$ Gates	(n+2)+2 Ancillas $\mathcal{O}(n)$ Gates [78, 79]	n/a
Exact block encoding poly degree d	n/a	$\mathcal{O}(nd)$	$\mathcal{O}(n2^d)$
Approximated block encoding poly degree d	$\mathcal{O}(n\ d(\delta_\infty))$ [41] $d(\delta_\infty) \sim \frac{\log(B/\delta_\infty)}{\nu}$ $\nu = 1 - \sin((2^n-1)/2^n)$	$\mathcal{O}(d\ k_0(\delta_\infty))$ $k_0(\delta_\infty) \sim \log\left(\frac{D(d^2-d)}{2\delta_\infty M}\right)$	$\mathcal{O}(n)$ $(\chi = 2)$ Non controllable error
Amplitude Amplification	$\mathcal{O}(n\ d(\delta_\infty)/\mathcal{F})$	$\mathcal{O}(k_0(\delta_\infty)d/\mathcal{F})$	n/a

Table 4: Comparison between the methodology proposed in Refs. [41, 42] and the current work, with $||P_{\text{exact}}(x)||_\infty = M$, $\mathcal{F}$ the filling ratio, $D = \max_k\{|c_k|\}$ and $\delta_\infty = ||P(j)/||P(j)||_\infty - \tilde{P}(j)/||\tilde{P}(j)||_\infty||_\infty$, where $P = \sum_{k=0}^{d} c_k x^k$ is the exact polynomial and $\tilde{P}$ its approximation, and $B \geq \sum_k |c_k|$. We neglect amplitude amplification and precomputational errors and complexities. In both cases where QSVT is used we have assumed no error in the classical preporcesing of the angles needed to implement the polynomial transformation.

ror resulting from the arcsin approximation. In order to explore the polynomial degree overhead needed in the methodology presented in Ref [41] to encode a polynomial into amplitudes of a quantum state we present Tab. 5. Additionally, our methodology allows the introduction of a controllable error that reduces the complexity of the exact case for the encoding of polynomial functions. In the general case of loading an arbitrary function the trade off between the additional resources of our protocol versus the approximation of arcsin should be taken into account. For instance, [41, 42] are more efficient for loading trigonometric polynomials.

In addition to QSVT-based methods, there have been approaches utilizing matrix product states (MPS) for the loading of smooth differential real-valued functions (SDR) into quantum state amplitudes [60, 61, 63, 69]. Ref. [61] shows

Degree d	**Fidelity**
4	0.0736
8	0.4969
10	0.8477
12	0.9672
14	0.9931
16	0.9985
18	0.9997
20	0.9999

Table 5: Fidelities corresponding to different degrees of the approximation polynomial employed to load $P(x) = \frac{1}{C_p}(x - \frac{1}{2^n-1})(x - \frac{20}{2^n-1})(x - \frac{50}{2^n-1})(x - \frac{60}{2^n-1})$ using the method outlined in Ref. [41].

that for such functions, favorable outcomes can be obtained by employing a fixed bond dimension of two, attributed to the logarithmic scaling of entanglement entropy in these cases [62]. In this paper, we have explored the resource requirements of this approach and compare it to our linear function+QSVT approach, specially in the case of loading of the linear function, for which we have analytical expressions for fidelity and error propagation.

Considering a different perspective, it was recently proposed the use of the Hadamard-Walsh series [58] for amplitude encoding. This proposal leverages the fact that for functions whose derivative is bounded, the error resulting from truncating their discrete Hadamard-Walsh Series is exponentially suppressed with the index of the truncation [103]. The authors use Hamiltonian simulation techniques presented in [99] to achieve a Hadamard-Walsh approximated simulation of the unitary $U = e^{-i\hat{f}\epsilon_0}$ with error ϵ_1, where $\hat{f} = \sum_x f(x)\,|x\rangle\langle x|$ is the operator corresponding to the target function to be encoded. Finally they use an ancillary qubit to generate the operator $-i(I - e^{-i\hat{f}\epsilon_0})$ acting on the state $\sum_x |x\rangle\,|1\rangle$, which approximates the target state to first order of ϵ_0 and introduces a protocol conditioned to the probability of measuring the ancillary qubit in the state $|1\rangle$. Therefore the total protocol introduces two sources of error, ϵ_1 corresponding to the truncation of the Hadamard-Walsh series and ϵ_0 corresponding to the Taylor expansion. This technique introduces an error for loading functions even for those cases in which $\epsilon_1 = 0$, as in the linear case. By contrast our subroutine

that uses the DHWT leverages the sparsity of the series corresponding to polynomial functions to efficiently generate quantum circuits that encode the states directly into the amplitudes. Additionally, this state loading algorithm based on the Hadamard-Walsh transform is not as efficient as our methodology for the application of loading the linear function, which is an important factor when loading the linear function for the derivative pricing use case.

Finally, we remark that our method is not limited by the condition of a bounded derivative of the target function, as is the case for the MPS [61] or the Hamiltonian simulation based on DHWT [58].

6 Conclusions

In this article, we have considered the problem of loading real polynomials into a quantum computer, with a particular focus on the encoding of the linear function. We have presented and reviewed two methodologies based on different approaches. The first method is based on matrix product states, and even though it offers competitive results for some particular cases [61], it is difficult to precisely control the error the method incurs. The second algorithm introduced in this paper utilizes the discrete Hadamard-Walsh transform (DHWT) to achieve the block encoding of the amplitudes which is fed into the quantum singular value transformation (QSVT) in order to apply a polynomial transformation to the amplitudes.This technique is able to exactly encode polynomials with same complexity as previous works [41,42] that incurred into an approximation error. Furthermore we have been able to reduce the complexity of the protocol via introducing a controllable error.

Using this technique based on the DHWT, the coefficients of the Hadamard-Walsh series of a given function are loaded into a quantum state and the inverse discrete Hadamard-Walsh transform (DHWT) is applied to achieve the amplitude encoding of the target function. This idea constitutes a novel and promising approach for functions whose Hadamard-Walsh series can be efficiently encoded, as for instance when it sparse [54] or efficiently truncated [99].

We would like to remark that even though our work has been focused on encoding real polynomials, it could be easily extended to load polynomials valued on complex values, i.e. $P : \mathbb{C} \to \mathbb{C}$, multivariate polynomials or even non-linear functions approximated with polynomials [41, 78, 79]. Future work will consider the feasibility of using our DHWT-based method to load highly discontinuous square wave-like functions.

Acknowledgements

We thank N. Guo and A. Rattew for the useful discussions regarding the amplitude transformations via the QSVT. We thank M. Cea-Fernandez for the discussions on the matrix product states. The authors acknowledge financial support from OpenSuperQ+100 (Grant No. 101113946) of the EU Flagship on Quantum Technologies, as well as from the EU FET-Open project EPIQUS (Grant No. 899368), also from Project Grant No. PID2021-125823NA-I00 595 and Spanish Ramón y Cajal Grant No. RYC-2020-030503-I funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe" and "ERDF Invest in your Future," this project has also received support from the Spanish Ministry for Digital Transformation and of Civil Service of the Spanish Government through the QUANTUM ENIA project call - Quantum Spain, and by the EU through the Recovery, Transformation and Resilience Plan – NextGenerationEU within the framework of the Digital Spain 2026 Agenda, we acknowledge funding from Basque Government through Grant No. IT1470-22 and the IKUR Strategy under the collaboration agreement between Ikerbasque Foundation and BCAM on behalf of the Department of Education of the Basque Government, as well as from and UPV/EHU Ph.D. Grant No. PIF20/276. PR acknowledges financial support from the CDTI within the Misiones 2021 program and the Ministry of Science and Innovation under the Recovery, Transformation and Resilience Plan—Next Generation EU under the project "CUCO: Quantum Computing and its Application to Strategic Industries"

References

[1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fer-

nando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. "Quantum supremacy using a programmable superconducting processor". Nature **574**, 505–510 (2019).

[2] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. "Strong quantum computational advantage using a superconducting quantum processor". Physical Review Letters**127** (2021).

[3] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. "Quantum computational advantage using photons". Science **370**, 1460–1463 (2020).

[4] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. "Logical quantum processor based on reconfigurable atom arrays". Nature (2023).

[5] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". Phys. Rev. Lett. **103**, 150502 (2009).

[6] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. "Quantum algorithm for systems of linear equations with exponentially improved dependence on precision". SIAM Journal on Computing **46**, 1920–1950 (2017).

[7] Nathan Wiebe, Daniel Braun, and Seth Lloyd. "Quantum algorithm for data fitting". Phys. Rev. Lett. **109**, 050505 (2012).

[8] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. "Preconditioned quantum linear system algorithm". Phys. Rev. Lett. **110**, 250504 (2013).

[9] Artur Scherer, Benoît Valiron, Siun-Chuon Mau, Scott Alexander, Eric van den Berg, and Thomas E. Chapuran. "Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2d target". Quantum Information Processing**16** (2017).

[10] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. "Quantum computational finance: Monte carlo pricing of financial derivatives". Phys. Rev. A **98**, 022321 (2018).

[11] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. "Option pricing using quantum computers". Quantum **4**, 291 (2020).

[12] Ana Martin, Bruno Candelas, Á ngel Rodríguez-Rozas, José D. Martín-Guerrero, Xi Chen, Lucas Lamata, Román Orús, Enrique Solano, and Mikel Sanz. "Toward pricing financial derivatives with an IBM quantum computer". Physical Review Research**3** (2021).

[13] Javier Gonzalez-Conde, Ángel Rodríguez-Rozas, Enrique Solano, and Mikel Sanz. "Efficient Hamiltonian simulation for solving option price dynamics". Phys. Rev. Research **5**, 043220 (2023).

[14] Dylan Herman, Cody Googin, Xiaoyuan Liu, Yue Sun, Alexey Galda, Ilya Safro, Marco Pistoia, and Yuri Alexeev. "Quantum computing for finance". Nature Reviews Physics (2023).

[15] Román Orús, Samuel Mugel, and Enrique Lizaso. "Quantum computing for finance: Overview and prospects". Reviews in Physics **4**, 100028 (2019).

[16] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. "Quantum computing for finance: State-of-the-art and future prospects". IEEE Transactions on Quantum Engineering **1**, 1–24 (2020).

[17] Gabriele Agliardi, Corey O'Meara, Kavitha Yogaraj, Kumar Ghosh, Piergiacomo Sabino, Marina Fernández-Campoamor, Giorgio Cortiana, Juan Bernabé-Moreno, Francesco Tacchino, Antonio Mezzacapo, and Omar Shehab. "Quadratic quantum speedup in evaluating bilinear risk functions" (2023). arXiv:2304.10385.

[18] Sarah K. Leyton and Tobias J. Osborne. "A quantum algorithm to solve nonlinear differential equations" (2008). arXiv:0812.4423.

[19] Dominic W. Berry, Andrew M. Childs, Aaron Ostrander, and Guoming Wang. "Quantum algorithm for linear differential equations with exponentially improved dependence on precision". Communications in Mathematical Physics **356**, 1057–1081 (2017).

[20] Jin-Peng Liu, Herman Øie Kolden, Hari K. Krovi, Nuno F. Loureiro, Konstantina Trivisa, and Andrew M. Childs. "Efficient quantum algorithm for dissipative nonlinear differential equations". Proceedings of the National Academy of Sciences**118** (2021).

[21] Benjamin Zanger, Christian B. Mendl, Martin Schulz, and Martin Schreiber. "Quantum algorithms for solving ordinary differential equations via classical integration methods". Quantum **5**, 502 (2021).

[22] Juan José García-Ripoll. "Quantum-inspired algorithms for multivariate analysis: from interpolation to partial differential equations". Quantum **5**, 431 (2021).

[23] Pablo Rodriguez-Grasa, Ruben Ibarrondo, Javier Gonzalez-Conde, Yue Ban, Patrick Rebentrost, Mikel Sanz. "Quantum approximated cloning-assisted density matrix exponentiation" (2023). arXiv:2311.11751.

[24] Dong An, Di Fang, Stephen Jordan, Jin-Peng Liu, Guang Hao Low, and Jiasu Wang, "Efficient quantum algorithm for nonlinear reaction-diffusion equations and energy estimation," (2022). arXiv:2305.11352.

[25] Dylan Lewis, Stephan Eidenbenz, Balasubramanya Nadiga, and Yiğit Subaşı, "Limitations for quantum algorithms to solve turbulent and chaotic systems," (2023) arXiv:2307.09593.

[26] Yen Ting Lin, Robert B. Lowrie, Denis Aslangil, Yiğit Subaşı, and Andrew T. Sornborger , "Koopman-von Neumann mechanics and the Koopman representation: A perspective on solving nonlinear dynamical systems with quantum computers," (2022) arXiv:2202.02188.

[27] Shi Jin, Nana Liu, and Yue Yu, "Time complexity analysis of quantum algorithms via linear representations for nonlinear ordinary and partial differential equations," Journal of Computational Physics, vol. 487, p. 112149, (2023).

[28] Ilon Joseph, "Koopman–von Neumann approach to quantum simulation of nonlinear

classical dynamics," Phys. Rev. Res., vol. 2, p. 043102, (2020).

[29] David Jennings, Matteo Lostaglio, Robert B. Lowrie, Sam Pallister, and Andrew T. Sornborger, "The cost of solving linear differential equations on a quantum computer: fast-forwarding to explicit resource counts," (2023) arXiv:2309.07881.

[30] David Jennings, Matteo Lostaglio, Sam Pallister, Andrew T Sornborger, and Yiğit Subaşı, "Efficient quantum linear solver algorithm with detailed running costs," (2023) arXiv:2305.11352.

[31] Javier Gonzalez-Conde and Andrew T. Sornborger "Mixed Quantum-Semiclassical Simulation," (2023) arXiv:2308.16147.

[32] Dimitrios Giannakis, Abbas Ourmazd, Philipp Pfeffer, Joerg Schumacher, and Joanna Slawinska, "Embedding classical dynamics in a quantum computer," Phys. Rev. A, vol. 105, p. 052404, (2022).

[33] François Gay-Balmaz and Cesare Tronci, "Evolution of hybrid quantum–classical wavefunctions," Physica D: Nonlinear Phenomena, vol. 440, p. 133450, (2022).

[34] Denys I. Bondar, François Gay-Balmaz and Cesare Tronci, "Koopman wavefunctions and classical–quantum correlation dynamics," Proceedings of the Royal Society A, vol. 475, no. 2229, p. 20180879, (2019).

[35] John Preskill. "Quantum computing in the NISQ era and beyond". Quantum **2**, 79 (2018).

[36] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. "Supervised learning with quantum-enhanced feature spaces". Nature **567**, 209–212 (2019).

[37] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. "A rigorous and robust quantum speed-up in supervised machine learning". Nature Physics **17**, 1013–1017 (2021).

[38] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. "Effect of data encoding on the expressive power of variational quantum-machine-learning models". Phys. Rev. A **103**, 032430 (2021).

[39] Maria Schuld and Francesco Petruccione. "Quantum models as kernel methods". Pages 217–245. Springer International Publishing. Cham (2021).

[40] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. "Quantum embeddings for machine learning" (2020). arXiv:2001.03622.

[41] Sam McArdle, András Gilyén, and Mario Berta. "Quantum state preparation without coherent arithmetic" (2022). arXiv:2210.14892.

[42] H. Li, H. Ni, L. Ying. "On efficient quantum block encoding of pseudo-differential operators". Quantum **7**, 1031 (2023).

[43] Mikko Mottonen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. "Transformation of quantum states using uniformly controlled rotations" (2004). arXiv:quant-ph/0407010.

[44] Xiaoming Sun, Guojing Tian, Shuai Yang, Pei Yuan, and Shengyu Zhang. "Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis" (2023). arXiv:2108.06150.

[45] Xiao-Ming Zhang, Man-Hong Yung, and Xiao Yuan. "Low-depth quantum state preparation". Phys. Rev. Res. **3**, 043200 (2021).

[46] Israel F. Araujo, Daniel K. Park, Francesco Petruccione, and Adenilton J. da Silva. "A divide-and-conquer algorithm for quantum state preparation". Scientific Reports**11** (2021).

[47] Jian Zhao, Yu-Chun Wu, Guang-Can Guo, and Guo-Ping Guo. "State preparation based on quantum phase estimation" (2019). arXiv:1912.05335.

[48] Lov K. Grover. "Synthesis of quantum superpositions by quantum computation". Phys. Rev. Lett. **85**, 1334–1337 (2000).

[49] Yuval R. Sanders, Guang Hao Low, Artur Scherer, and Dominic W. Berry. "Black-box quantum state preparation without arithmetic". Phys. Rev. Lett. **122**, 020502 (2019).

[50] Johannes Bausch. "Fast Black-Box Quantum State Preparation". Quantum **6**, 773 (2022).

[51] Lov Grover and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions" (2002). arXiv:quant-ph/0208112.

[52] Arthur G. Rattew and Bálint Koczor. "Preparing arbitrary continuous functions in quantum registers with logarithmic complexity" (2022). arXiv:2205.00519.

[53] Shengbin Wang, Zhimin Wang, Runhong He, Shangshang Shi, Guolong Cui, Ruimin Shang, Jiayun Li, Yanan Li, Wendong Li, Zhiqiang Wei, and Yongjian Gu. "Inverse-coefficient black-box quantum state preparation". New Journal of Physics **24**, 103004 (2022).

[54] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. "Quantum state preparation with optimal circuit depth: Implementations and applications". Phys. Rev. Lett. **129**, 230504 (2022).

[55] Gabriel Marin-Sanchez, Javier Gonzalez-Conde, and Mikel Sanz. "Quantum algorithms for approximate function loading". Phys. Rev. Research. **5**, 033114 (2023).

[56] Kouhei Nakaji, Shumpei Uno, Yohichi Suzuki, Rudy Raymond, Tamiya Onodera, Tomoki Tanaka, Hiroyuki Tezuka, Naoki Mitsuda, and Naoki Yamamoto. "Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators". Phys. Rev. Res. **4**, 023136 (2022).

[57] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum generative adversarial networks for learning and loading random distributions". npj Quantum Information **5**, 103 (2019).

[58] Julien Zylberman and Fabrice Debbasch. "Efficient quantum state preparation with walsh series" (2023). arXiv:2307.08384.

[59] Mudassir Moosa, Thomas W. Watts, Yiyou Chen, Abhijat Sarma, and Peter L. McMahon. "Linear-depth quantum circuits for loading fourier approximations of arbitrary functions" . In Quantum Science and Technology (Vol. 9, Issue 1, p. 015002) (2023).

[60] Lars Grasedyck. "Polynomial approximation in hierarchical tucker format by vector - tensorization" (2010). Mathematics, Computer Science

[61] Adam Holmes and A. Y. Matsuura. "Efficient quantum circuits for accurate state preparation of smooth, differentiable functions" (2020). arXiv:2005.04351.

[62] Adam Holmes and A. Y. Matsuura. "Entanglement properties of quantum superpositions of smooth, differentiable functions" (2020). arXiv:2009.09096.

[63] Ar A Melnikov, A A Termanova, S V Dolgov, F Neukart, and M R Perelshtein. "Quantum state preparation using tensor networks". Quantum Science and Technology **8**, 035027 (2023).

[64] Rohit Dilip, Yu-Jie Liu, Adam Smith, and Frank Pollmann. "Data compression for quantum machine learning". Phys. Rev. Res. **4**, 043007 (2022).

[65] Sheng-Hsuan Lin, Rohit Dilip, Andrew G. Green, Adam Smith, and Frank Pollmann. "Real- and imaginary-time evolution with compressed quantum circuits". PRX Quantum**2** (2021).

[66] Michael Lubasch, Pierre Moinier, and Dieter Jaksch. "Multigrid renormalization". Journal of Computational Physics **372**, 587–602 (2018).

[67] Michael Lubasch, Jaewoo Joo, Pierre Moinier, Martin Kiffner, and Dieter Jaksch. "Variational quantum algorithms for nonlinear problems". Phys. Rev. A **101**, 010301 (2020).

[68] Nikita Gourianov, Michael Lubasch, Sergey Dolgov, Quincy Y. van den Berg, Hessam Babaee, Peyman Givi, Martin Kiffner, and Dieter Jaksch. "A quantum-inspired approach to exploit turbulence structures". Nature Computational Science **2**, 30–37 (2022).

[69] Jason Iaconis, Sonika Johri, and Elton Yechao Zhu. "Quantum state preparation of normal distributions using matrix product states" (2023). arXiv:2303.01562.

[70] Vanio Markov, Charlee Stefanski, Abhijit Rao, and Constantin Gonciulea. "A generalized quantum inner product and applications to financial engineering" (2022). arXiv:2201.09845.

[71] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. "Option pricing using quantum computers". Quantum **4**, 291 (2020).

[72] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. "Methodology of resonant equiangular composite quantum gates". Phys. Rev. X **6**, 041067 (2016).

[73] Guang Hao Low and Isaac L. Chuang. "Optimal hamiltonian simulation by quantum signal processing". Phys. Rev. Lett. **118**, 010501 (2017).

[74] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". Quantum **3**, 163 (2019).

[75] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In Proceedings of the 51st Annual ACM SIGACT Symposium on Theoryof Computing ACM (2019).

[76] Ewin Tang and Kevin Tian. "A cs guide to the quantum singular value transformation" (2023). arXiv:2302.14324.

[77] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. "Efficient phase-factor evaluation in quantum signal processing". Phys. Rev. A **103**, 042419 (2021).

[78] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. "Nonlinear transformation of complex amplitudes via quantum singular value transformation" (2021) arXiv:2107.10764

[79] Arthur G. Rattew and Patrick Rebentrost "Non-Linear Transformations of Quantum Amplitudes: Exponential Improvement, Generalization, and Applications" (2023) arXiv:2309.09839.

[80] W. Fraser. "A Survey of Methods of Computing Minimax and Near-Minimax Polynomial Approximations for Functions of a Single Independent Variable", Journal of the ACM **12**, 295 (1965)

[81] E. Y. Remez, "General computational methods of Chebyshev approximation: The problems with linear real parameters", US Atomic Energy Commission, Division of Technical Information (1962).

[82] Román Orús. "A practical introduction to tensor networks: Matrix product states and projected entangled pair states". Annals of Physics (New York) (2014).

[83] Guifré Vidal. "Efficient classical simulation of slightly entangled quantum computations". Physical Review Letters**91** (2003).

[84] F. Verstraete, V. Murg, and J.I. Cirac. "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems". Advances in Physics **57**, 143–224 (2008).

[85] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. "Matrix product state representations". Quantum Info. Comput. 7, 5, 401–430. (2007)

[86] Shi-Ju Ran. "Encoding of matrix product states into quantum circuits of one- and two-qubit gates". Physical Review A**101** (2020).

[87] Daniel Malz, Georgios Styliaris, Zhi-Yuan Wei, and J. Ignacio Cirac. "Preparation of matrix product states with log-depth quantum circuits". Phys. Rev. Lett. 132, 040404 (2024).

[88] J. L. Walsh. "A closed set of normal orthogonal functions". American Journal of Mathematics **45**, 5–24 (1923)..

[89] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. "Singular value decomposition and principal component analysis". Pages 91–109. Springer US. Boston, MA (2003).

[90] Ivan Oseledets. "Constructive representation of functions in low-rank tensor formats". Constructive Approximation**37** (2010).

[91] Norbert Schuch, Michael M. Wolf, Frank Verstraete, and J. Ignacio Cirac. "Entropy scaling and simulability by matrix product states". Physical Review Letters**100** (2008).

[92] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Annals of Physics **326**, 96–192 (2011).

[93] Carl Eckart and G. Marion Young. "The approximation of one matrix by another of lower rank". Psychometrika **1**, 211–218 (1936).

[94] Manuel S. Rudolph, Jing Chen, Jacob Miller, Atithi Acharya, and Alejandro Perdomo-Ortiz. "Decomposition of matrix product states into shallow quantum circuits" (2022). arXiv:2209.00595.

[95] C. Schön, E. Solano, F. Verstraete, J. I. Cirac, and M. M. Wolf. "Sequential generation of entangled multiqubit states". Phys. Rev. Lett. **95**, 110503 (2005).

[96] Vivek V. Shende, Igor L. Markov, and Stephen S. Bullock. "Minimal universal two-qubit controlled-NOT-based circuits". Physical Review A**69** (2004).

[97] 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". Physical Review A **52**, 3457–3467 (1995).

[98] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Annals of Physics **326**, 96–192 (2011).

[99] Jonathan Welch, Daniel Greenbaum, Sarah Mostame, and Alan Aspuru-Guzik. "Efficient quantum circuits for diagonal unitaries without ancillas". New Journal of Physics **16**, 033040 (2014).

[100] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. "The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation". In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). Volume 132 of Leibniz International Proceedings in Informatics (LIPIcs), pages 33:1–33:14. Dagstuhl, Germany (2019). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[101] T. Constantinescu. "Schur parameters, factorization, and dilation problems". Operator Theory: Advances and Applications. Birkhäuser Verlag. (1996).

[102] Shengbin Wang, Zhimin Wang, Wendong Li, Lixin Fan, Guolong Cui, Zhiqiang Wei, and Yongjian Gu. "Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method". Quantum Information Processing**19** (2020).

[103] Chung-Kwong Yuen. "Function approximation by walsh series". IEEE Transactions on Computers **C-24**, 590–598 (1975).

[104] Rui Chao, Dawei Ding, Andras Gilyen, Cupjin Huang, and Mario Szegedy. "Finding angles for quantum signal processing with machine precision" (2020). arXiv:2003.02831.

[105] Jeongwan Haah. "Product Decomposition of Periodic Functions in Quantum Signal Processing". Quantum **3**, 190 (2019).

Appendix A Proofs

In this appendix we derive the expression shown in Eq. 11. Let us first consider the exact state corresponding to the Walsh Hadamard series

$$\left|\tilde{\Phi}_L\right\rangle = \frac{1}{\tilde{C}} \sum_{|k|_b \leq 1} x_k^{(n)} |k\rangle = x_n^{(0)} |0...0\rangle + x_n^{(2^0)} |0...1\rangle + ... + x_n^{(2^{n-1})} |1...0\rangle$$

with $\tilde{C} = 2^{n/2}\sqrt{\left(\frac{2^n-1}{2}\right)^2 + \frac{2^{2n}-1}{12}}$. If we truncate keeping the largest k_0 values with $|k|_b = 1$ and $x_0^{(n)}$, the renormalized state results

$$\left|\tilde{\Phi}_L^{k_0}\right\rangle = \frac{1}{\tilde{C}_{k_0}} \sum_{|k|_b \leq 1} x_n^{(k)} |k\rangle = x_n^0 |0.....0\rangle + x_n^{2^{n-k_0}} |0...1...0\rangle + ... + x_n^{(2^{n-1})} |1.....0\rangle$$

with $\tilde{C}_{k_0} = 2^{n/2}\sqrt{\left(\frac{2^n-1}{2}\right)^2 + \frac{1}{12}\left(2^{2n} - 2^{2(n-k_0)}\right)}$. When we calculate

$$\left|\langle\tilde{\Phi}_L|\tilde{\Phi}_L\rangle_{k_0}\right|^2 = \left(\frac{\tilde{C}_{k_0}}{\tilde{C}}\right)^2$$

and manipulating this expression, this lead to Eq. 11.

Appendix B Variational circuit for linear function

In this article, we investigated the loading of a linear function using an MPS (Matrix Product State) with a bond dimension of $\chi = 1$, which yielded remarkably high fidelities. These high fidelities suggest that the linear function closely resembles a product state, given that an MPS with $\chi = 1$ can be efficiently constructed using single qubit rotations $R_y(\theta)$. Building on this insight, our focus in this section is to develop a variational algorithm to find the optimal angles that maximize the fidelity of our linear function approximation within a product state framework.

To initiate the process, we derived the angles from the MPS with bond dimension 1 for a system of $n = 8$ qubits. Subsequently, we performed an analytical fitting for the angles, given by

$$\theta_q = \exp\left(\exp\left(-q^{0.9}/1.23\right) - 0.24\right), \tag{23}$$

where q ranges from 1 to 8, representing each qubit. The fitting results are depicted in Fig. 6.

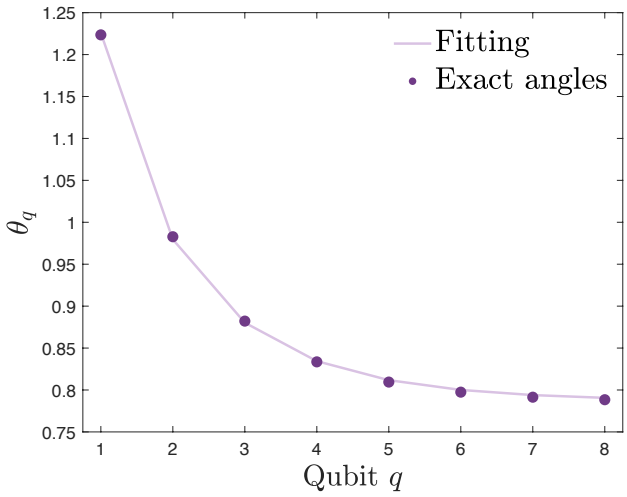

Figure 6: Angles of the single qubit rotations to load the linear function through an MPS with $\chi = 1$ for $n = 8$ and its analytical fitting.

Moving on to our variational model, we utilized a system with $n = 6$ qubits, aiming to maximize the fidelity of the linear function. We initialized the angles based on the fitting provided by Eq. 23. To optimize this process, we employed gradient descent and employed the squared L_2 norm as the loss function to quantify the disparity between the exact linear function and the predicted state. For the $n = 6$ qubit system, the fidelity between the exact linear function and the one obtained from the MPS with $\chi = 1$ was $F_{\text{MPS}} = 0.9875747$. After the training, our variational model achieved an improved fidelity of $F_{\text{var}} = 0.9875750$.

Appendix C Block Encoding of Real-valued Statevector Amplitudes

In this paper we are primarily concerned applying the methods of Ref. [78] to loading polynomial functions into quantum registers. In order to achieve this, we introduced an amplitude encoding of the linear function $U_L : |0\rangle_n \mapsto |\Phi_L\rangle \propto \sum_j j|j\rangle_n$. The unitary U_L is used to construct a block-encoding U_A of a matrix $A \propto \text{diag}(0, 1, ..., 2^n - 1)$ to which the either the QSVT can be applied to transform the diagonal entries by an arbitrary polynomial; alternatively products and linear combinations of these block-encodings can be used to more efficiently encode low degree polynomials. In this appendix, we consider how the methods of Ref. [78] have been improved upon by Ref. [79] in order to efficiently construct the block-encoding U_A.

When constructing the block-encoding U_A, we are concerned with the special case where the statevector amplitude loading unitary $U|0\rangle_n = \sum_j \psi_j|j\rangle_n$ has real-valued amplitudes $\psi_j \in \mathbb{R}$ i.e., $U = U_L$. In order to construct U_A, we first must define several unitary operators. Following the notation introduced in Ref. [79], we first define the operator W_0

$$W_0 := (\mathcal{I}_n \otimes H \otimes \mathcal{I}_n)CU_C(\mathcal{I}_n \otimes H \otimes \mathcal{I}_n) \tag{24}$$

where U_C and C are defined as

$$U_C := (U \otimes |0\rangle\langle 0|_1 + \mathcal{I}_n \otimes |1\rangle\langle 1|_1) \otimes \mathcal{I}_n \tag{25}$$

$$C := \mathcal{I}_n \otimes |0\rangle\langle 0|_1 \otimes \mathcal{I}_n + \sum_{k,j=0}^{2^n-1} |j \oplus k\rangle\langle j|_n \otimes |1\rangle\langle 1|_1 \otimes |k\rangle\langle k|_n \tag{26}$$

U_C is implemented using a controlled version of the amplitude loading unitary U and C is "controlled-copy" circuit which can be implemented with a cascade of n Toffoli gates, see Ref. [79] for more details.

The W_0 operator sends states $|0\rangle_n|0\rangle_1|k\rangle_n$ to $|\Phi_k^0\rangle = \frac{1}{2}((|\psi\rangle_n + |k\rangle_n)|0\rangle_1 + (|\psi\rangle_n - |k\rangle_n)|1\rangle_1)|k\rangle_n$. The W_0 operator is used to construct another operator G_0 which is defined as

$$G_0 := W_0((\mathcal{I}_{n+1} - 2|0\rangle\langle 0|_{n+1}) \otimes \mathcal{I}_n)W_0^\dagger(\mathcal{I}_n \otimes Z \otimes \mathcal{I}_n) \tag{27}$$

The operator G_0 has the important property that $|\Phi_k^0\rangle$ are its eigenvectors with eigenvalues $\psi_j \in \mathbb{R}$. G_0 and its inverse $G_0^\dagger$ can then be used to construct the desired $(1, n + 2, 0)$-block-encoding U_A of $A = \text{diag}(\psi_0, \psi_1, ..., \psi_{2^n-1})$,

$$U_A := (XZX \otimes \mathcal{I}_{2n+1})(H \otimes W_0^\dagger)(|0\rangle\langle 0|_1 \otimes G_0 + |1\rangle\langle 1|_1 \otimes G_0^\dagger)(H \otimes W_0) \tag{28}$$

As stated in Ref. [79], U_A can be implemented with $O(n)$ circuit depth, 3 queries to a controlled-U gate and 3 queries to an inverse controlled-U gate.

Additionally, one can also consider some alternative ways to achieve the block encoding of the linear function

- Using the rotation oracle acting on $\sum_x |x\rangle |0\rangle$ [48]:

$$\mathfrak{rot} |x\rangle |0\rangle \rightarrow |x\rangle (\sin(\theta) |0\rangle + \cos(\theta) |1\rangle),$$

  where θ is a m bit approximation to $\arcsin(x)$. To implement this procedure, the quantum computer would calculate θ, store the value in m ancillary registers, and use that register as the control for a sequence of rotation operations on the qubit. Therefore this step incurs into the use of coherent arithmetics.

- Using the comparing oracle acting on $\sum_x |x\rangle |0\rangle^n |0\rangle$. We firstly implement the Amplitude oracle that encodes the uniform superposition in a secondary ancilla register $|x\rangle$,

$$\mathfrak{amp} |x\rangle |0\rangle^n |0\rangle \rightarrow \frac{1}{2^n} \sum_y |x\rangle |y\rangle |0\rangle .$$

  This can be straightforwardly done by applying a layer of Hadamard gates to n ancillary qubits.

  Next we use the comparison oracle [49]

$$\mathfrak{comp} |x\rangle |y\rangle |0\rangle \rightarrow \begin{cases} |x\rangle |y\rangle |0\rangle & if \quad x < y \\ \\ |x\rangle |y\rangle |1\rangle & if \quad x \geq y \end{cases}$$

  By composing this two oracles, $\mathfrak{comp} \circ \mathfrak{amp}$, and apply them to the initial state $\sum_x |x\rangle |0\rangle^n |0\rangle$, and subsequently erasing the n-ancilla register, one obtain the state $|x\rangle (x/2^n |0\rangle + \sqrt{1-(x/2^n)^2} |1\rangle)$. The complexity of this protocol is $\mathcal{O}(n)$ and uses $n+1$ ancillas for the block encoding. Therefore, this is an equivalent methodology to achieve the same result that we have presented in the main text.

- Using Hamiltonian simulation techniques applied to the unitary dilation of B. As we already mention in the main text, by applying the unitary dilation technique [101] to $B = \frac{1}{2^n-1} \sum_{n=0}^{2^n-1} j |j\rangle \langle j|$, given $\|B\| \leq 1$. This operation would require an efficient simulation of the Hamiltonian $H = \hat{\sigma}^y \otimes \arccos(B)$. This leads to the block encoding unitary

$$U = \begin{pmatrix} B & \sqrt{1-B^2} \\ \sqrt{1-B^2} & -B \end{pmatrix} = (\hat{\sigma}^z \otimes \mathbb{I}) \exp(i\hat{\sigma}^y \otimes H),$$

  which is a $(1,1,0)$-block encoding, and could be efficiently implemented with the techniques shown in Ref. [102].

Appendix D Additional numerical examples

Here we provide two more numerical experiments for $n = 6$ qubits for two different polynomials.

Method	$P_1(x)$ L_2 **norm**	$P_1(x)$ **Fidelity poly**	$P_1(x)$ $\mathcal{F}$	$P_2(x)$ L_2 **norm**	$P_2(x)$ **Fidelity poly**	$P_2(x)$ $\mathcal{F}$
Lin MPS ($\chi = 1$) + QSVT	0.0790	0.6408	-	0.0570	0.8026	-
DHWT ($k_0 = 1$) + QSVT	0.1715	0.0034	0.7234	0.2357	0.6044	0.8539
DHWT ($k_0 = 2$) + QSVT	0.1889	0.0202	0.5801	0.1522	0.0671	0.7884
DHWT ($k_0 = 3$) + QSVT	0.1339	0.1818	0.5542	0.0700	0.7112	0.7010
DHWT ($k_0 = 4$) + QSVT	0.0647	0.7498	0.5023	0.0304	0.9418	0.6677
DHWT ($k_0 = 5$) + QSVT	0.0252	0.9598	0.4730	0.0115	0.9915	0.6539
DHWT ($k_0 = 6$) + QSVT	0	1	0.4621	0	1	0.6551
Direct Pol MPS ($\chi = 1$)	0.0885	0.5619	-	0.0763	0.6622	-
Direct Pol MPS ($\chi = 2$)	0.0291	0.9467	-	0.0105	0.9930	-
Direct Pol MPS ($\chi = 3$)	0.0134	0.9885	-	0.0014	0.9999	
Direct Pol MPS ($\chi = 4$)	0	1	-	0	1	-

Table 6: Error in L_2 Norm and Fidelities for Different Loading Methods of two different polynomials $P_1(x) = \frac{1}{C_p}(x - 2/(2^n-1))(x - 16/(2^n-1))(x - 40/(2^n-1))(x - 50/(2^n-1))(x - 62/(2^n-1))$ and $P_1(x) = \frac{1}{C_p}(x - 2/(2^n-1))(x - 32/(2^n-1))(x - 60/(2^n-1))$. We also provide the filling ratios $\mathcal{F}$ for the different k_0 values

.

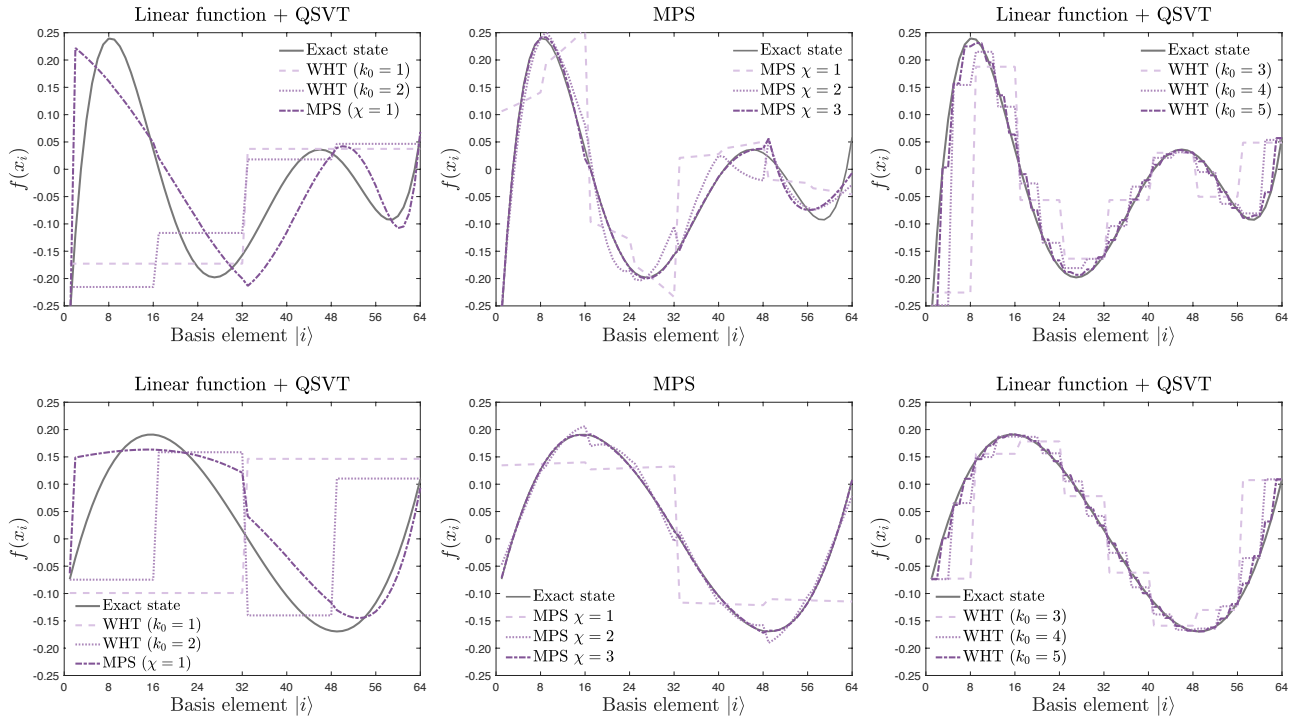

Figure 7: Various approaches for loading polynomial functions $P_1(x) = \frac{1}{C_p}(x - \frac{2}{2^n-1})(x - \frac{16}{2^n-1})(x - \frac{40}{2^n-1})(x - \frac{50}{2^n-1})(x - \frac{62}{2^n-1})$ in the upper portion of the figure and $P_2(x) = \frac{1}{C_p}(x - \frac{2}{2^n-1})(x - \frac{32}{2^n-1})(x - \frac{60}{2^n-1})$ in the lower portion for $n = 6$ qubits.