

Quantum Amplitude Estimation Using Direction Of Arrival Estimation

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of and priority to U.S. Provisional Patent Application Serial No. 63/451,895, “Amplitude Estimation from Classical Signal Processing,” filed on March 13, 2023, the subject matter of which is incorporated herein by reference in its entirety.

BACKGROUND

1. Technical Field

[0002] This disclosure relates generally to quantum amplitude estimation (AE), and more particularly, to estimating the amplitude of a quantum oracle using direction of arrival (DOA) estimation.

2. Description of Related Art

[0003] Amplitude Estimation is a fundamental quantum algorithm with many applications. For example, it provides a quadratic speedup in Monte Carlo methods, which is applicable to applications relying on quadratic speedups.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Embodiments of the disclosure have other advantages and features which will be more readily apparent from the following detailed description and the appended claims, when taken in conjunction with the examples in the accompanying drawings, in which:

[0005] FIG. 1 is a plot diagram of different example arrays (one example physical array and two virtual arrays).

[0006] FIG. 2A is a plot of the number of oracle queries vs. estimation error, according to one or more embodiments.

[0007] FIG. 2B is a plot of the parallel query complexity vs. estimation error, according to one or more embodiments.

[0008] FIG. 3A is a plot of amplitude vs. number of oracle queries, according to one or more embodiments.

[0009] FIG. 3B is a plot of amplitude vs. parallel query complexity, according to one or more embodiments.

[0010] FIG. 4 is a flowchart of an example method for estimating an amplitude of a quantum oracle using direction of arrival (DOA) estimation, according to one or more embodiments.

[0011] FIGS. 5A-5B are block diagrams of a computing system including a classical computing system and a quantum computing system, according to one or more embodiments.

[0012] FIG. 5C-5D are block diagrams of components of a quantum computing system, according to one or more embodiments.

[0013] FIG. 5E is a flow chart that illustrates an example execution of a quantum routine on the computing system, according to one or more embodiments.

[0014] FIG. 6 is an example architecture of a classical computing system, according to one or more embodiments.

DETAILED DESCRIPTION

[0015] The figures and the following description relate to preferred embodiments by way of illustration only. It should be noted that from the following discussion, alternative embodiments of the structures and methods disclosed herein will be readily recognized as viable alternatives that may be employed without departing from the principles of what is claimed.

[0016] While on the surface two completely distinct disciplines, this disclosure demonstrates that quantum amplitude estimation, a core subroutine used in many quantum algorithms, can be mapped directly to algorithms in signal processing called direction of arrival (DOA) estimation, where the goal is to determine the direction of arrival of an incoming wave with the fewest possible measurements. This association enables use of the vast amount of signal processing algorithms to post-process measurements of the Grover iterator at predefined depths. Using an off-the-shelf DOA algorithm together with a sparse-array sampling approach, enables creation of a phase-estimation free fully parallel quantum amplitude estimation (AE) algorithm as further described below. In one example implementation, the algorithm has a total query complexity of $\sim 3.73/\epsilon$ and a parallel query complexity of $\sim 0.28/\epsilon$ at 95% confidence, which is a factor of $1.2 \times$ and $15 \times$ improvement over chebAE (see P. Rall and B. Fuller, Quantum 7, 937 (2023)), which to the knowledge of the inventors, constituted the previous best published results for amplitude estimation. Overall, the approach presented in this disclosure provides a robust and parallelizable method to performing quantum AE that uses concepts from classical signal processing.

I. INTRODUCTION

[0017] Quantum Amplitude Estimation is a fundamental quantum algorithm with many applications. For example, it provides a quadratic speedup in Monte Carlo methods, which is applicable to applications relying on quadratic speedups such as those encountered in the financial sector as well as a subroutine to improve the complexity of algorithms requiring one to estimate for example overlaps of states at the end of the quantum linear system algorithm.

[0018] An example setting is as follows: suppose access to a quantum unitary operator U such that $U|0^l\rangle = \cos\theta|x, 0\rangle + \sin\theta|x', 1\rangle$ for some x, x' quantum states on the $l - 1$ qubits and unknown $\theta \in [0, \pi/2]$ (Note that the l th qubit (with the 0/1 quantum state) may be referred to as the “last,” “final” or “target” qubit. However, this terminology is merely for convenience, and this qubit is not required to be last or final relative to other qubits (e.g., it is not required to be the last or final qubit on a quantum register)). The goal is to design an algorithm that finds θ (or the amplitude $\cos\theta$) up to an additive error $\varepsilon > 0$. Classically, one would simply estimate θ using $\mathcal{O}(1/\varepsilon^2)$ samples. Using quantum AE, the $\mathcal{O}(1/\varepsilon^2)$ samples (from the classical AE algorithm) can be improved to a scaling that instead uses $\mathcal{O}(1/\varepsilon)$ applications of the unitary U , which is a quadratic speedup.

[0019] To achieve the improved scaling, let R_0 be the reflection in $|0^l\rangle$, that is $R_0|x\rangle = -|x\rangle$ for $x \neq 0^l$ and $R_0|0^l\rangle = |0^l\rangle$. Let S_0 be the reflection in $|0\rangle$ in the last qubit, that is $S_0|x, 0\rangle = |x, 0\rangle$ and $S_0|x, 1\rangle = -|x, 1\rangle$ for all x . The Grover Operator (also referred to as the Grover oracle) G can be defined $G := UR_0U^{-1}S_0$ which has the following property:

$$|\phi_n\rangle := G^n U|0^l\rangle = \cos((2n+1)\theta)|x, 0\rangle + \sin((2n+1)\theta)|x', 1\rangle, \quad (1)$$

where n describes the number of sequential oracle calls to G (using the terminology from above, the l th qubit (with the 0/1 quantum state) is the “last,” “final” or “target” qubit). Note that AE was first introduced as a combination of Grover search and Quantum Phase Estimation (QPE). However, AE variants have been developed without requiring QPE, such as chebAE.

[0020] Generally, quantum algorithms for amplitude estimation without phase estimation take measurements of a quantum state at different values of n and use classical post-processing either at the end or iteratively to determine at what n to take samples next. The downside to the iterative approaches is that one has to switch between quantum and classical repetitions in a serial manner, which may be undesirable in practice (e.g., this increases the runtime).

[0021] In contrast, this disclosure describes a quantum AE algorithm where the number of iterations is known from the outset (in other words, classical processing isn't required after each iteration to determine at what n to take samples next), so every sample can be performed in parallel (e.g., using multiple quantum computing systems). In addition, the classical post-processing is robust to noise, allowing the quantum AE algorithm to take fewer samples to achieve low overall query complexity. Furthermore, the classical post-processing is classically efficient, thus not affecting the overall quantum AE algorithm complexity up to log factors. This disclosure draws from ideas in (classical) signal processing, in particular from algorithms used in determining the Direction Of Arrival (DOA) of an incoming signal.

[0022] In DOA, a set of sensors is placed at certain positions in space to detect an incoming signal from an unknown position. The measurement data is then used to determine the angle of arrival of the incoming signal relative to the position of the sensors. There are many algorithms for doing this, for example the Multiple Signal Classification (MUSIC) algorithm or Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) along with many variants and sampling techniques. For more information on MUSIC, see A. Barabell, J. Capon, D. DeLong, J. Johnson, and K. Senne, *Performance comparison of superresolution array processing algorithms. revised*, Tech. Rep. (MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1998). For more information on ESPRIT, see R. Roy and T. Kailath, IEEE Transactions on acoustics, speech, and signal processing 37, 984 (1989). For more information on variants and sampling techniques, see Z. Yang, J. Li, P. Stoica, and L. Xie, in *Academic Press Library in Signal Processing, Volume 7*, edited by R. Chellappa and S. Theodoridis (Academic Press, 2018) pp. 509–581.

[0023] The number of sensors and the spacing determines the accuracy or resolution with which one could estimate the DOA. Reducing (e.g., minimizing) the number of sensors while simultaneously achieving high accuracy is desired for high-performance. Using super-resolution methods allows the creation of virtual arrays from sparse arrays, where the effective sensor spacing is greater than the number of actual sensors.

[0024] One example are coprime arrays, where one combines two uniform samplers with sample spacings MT and NT where M and N are coprime integers and T has dimension of space or time. This allows one to generate $\mathcal{O}(MN)$ sample locations using only $\mathcal{O}(M + N)$ physical samples. This can be further improved using multiple level nested arrays to achieve $\mathcal{O}(N^{2q})$ virtual sensors for some integer q using just $\mathcal{O}(N)$ physical sensors. For more information, see P. Pal and P. P. Vaidyanathan, in *2011 Digital Signal Processing and Signal*

Processing Education Meeting (DSP/SPE) (2011) pp. 289–294 and P. Pal and P.

Vaidyanathan, *IEEE Transactions on Signal Processing* 60, 1253 (2011).

[0025] It turns out that there is a nearly one-to-one correspondence between the measurements one gets from DOA sensors and the measurements one obtains when measuring the quantum state of Eqn. 1. This allows the conversion of quantum measurements into the form of a signal vector that can be post-processed using these DOA algorithms to estimate the amplitude. Note that some of the example results herein use one type of sensor spacing approach (the $2q$ array) and one DOA algorithm (ESPRIT). However, there are a wide number of variations of different sampling strategies and DOA algorithms that can be used (e.g., to improve (e.g., optimize) the approaches herein). For more information on a $2q$ array, see P. Pal and P. Vaidyanathan, *IEEE Transactions on Signal Processing* 60, 1253 (2011). For more information on ESPRIT, see R. Roy and T. Kailath, *IEEE Transactions on acoustics, speech, and signal processing* 37, 984 (1989).

II. CLASSICAL SIGNAL PROCESSING BASED AE

A. Direction Of Arrival Estimation

[0026] Before introducing the super-resolution-based approaches, the basic theory of subspace-based DOA estimating will be discussed. In particular, this disclosure will review the ESPRIT algorithm. Suppose a linear array of sensors placed at positions $x_1, \dots, x_M$ and there is one source of incoming signal y with elevation angle $\omega \in (-\pi/2, \pi/2)$. Then, using the sensor at position x_n a measurement of $y_n = e^{i\omega x_n}$ can be obtained. This provides an approximation $y' = y + \epsilon$ where $y', y, \epsilon \in \mathbb{C}^n$ where ϵ is an error term due to imperfect measurements.

[0027] ESPRIT is an algorithm that uses the measurements y' to obtain the unknown angle ω . The positions x_n are assumed to be uniform, where some unit of distance is used such that $x_n = n - 1$. Such arrays are referred to herein as Uniform Linear Arrays (ULA).

Algorithm 1 ESPRIT

Setup: Measurements of a signal $y(x) = e^{i\omega x}$ on a uniformly spaced array of sensors.

Goal: An approximation of ω up to error $O(1/M)$ where M is the size of the array of sensors.

- 1: Form the Toeplitz matrix R with first row the vector of measurements $y \in \mathbb{C}^M$ and compute its singular value decomposition $R = U\Sigma V$.
- 2: Form the matrix S from the first 2 columns of U , the matrix S_1 from the first $M - 1$ rows S , and the matrix S_2 from the last $M - 1$ rows of S .

- 3: Compute $P = S_1^{-1}S_2$ where S_1^{-1} is the pseudo-inverse.
- 4: Output the angle of the first eigenvalue of P .

[0028] The full Singular Value Decomposition of the matrix R is not required. Instead, determining the top two eigenvectors is sufficient. Since R is Toeplitz, the Lanczos algorithm, together with efficient algorithms for performing matrix-vector multiplication of Toeplitz matrices, can be used to obtain these two eigenvectors in time $O(M \log(M))$.

B. Physical and virtual arrays

[0029] In DOA, the “physical array” refers to sensors placed at certain locations in a linear array. The *virtual array* concept is the idea that one can use the data from the physical array to get signal measurements at positions where there is no physical sensor by combining the measurements of the signal from the physical array in a certain way.

[0030] For example, many DOA algorithms, such as ESPRIT, uses measurements of the incoming signal at physical positions which are evenly spaced. The virtual array concept allows us to obtain such measurements by using fewer physical sensors in the following way. As an example, place the sensors at physical locations $x = (x_1, x_2, \dots, x_n)$, which is a subset of sensor positions to be used for ULAs, and consider the vector of signals at those positions $y_j = e^{i\omega x_j}$. Now, when the outerproduct of y is computed with itself, the following matrix $(yy^T)_{ij} = e^{i\omega(x_i - x_j)}$ is obtained. This is the value of the signal value at the physical location $x_i - x_j$. So suppose measurements y'_j of y_j at the locations x_j are known or determined, the value $(y'y'^T)_{ij}$ can be used as a measurement value of the signal y at location $x_i - x_j$ even though the signal physically at that location was never measured. This technique allows many more (virtual) measurements to be acquired than the number of physical sensors. Given a vector $x \in \mathbb{R}^n$ that represents a linear array of physical sensors where sensor i is placed at position x_i , for integer $q \geq 1$ its *virtual array* can be defined by taking repeated outer products q times.

[0031] **Definition II.1** (Virtual array). Let $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ and $q \geq 1$ an integer. The $2q$ -th order virtual array corresponding to x is given by

$$S_q := \{\sum_{i=1}^q x_{k_i} - \sum_{i=q+1}^{2q} x_{k_i} : k_i \in [n]\}. \quad (2)$$

For more information on Definition II.1, see P. Chevalier and A. Ferreol, IEEE Transactions on Signal Processing 47, 2592 (1999), and P. Chevalier, A. Ferreol, and L. Albera, IEEE Transactions on Signal Processing 54, 2986 (2006).

[0032] It may be desirable (e.g., important) to decrease (e.g., minimize) the size of the physical array while increasing (e.g., maximizing) the size of the virtual array. As demonstrated in the next section, the elements of the physical array x_j can correspond to the number of Grover operators applied.

[0033] Long virtual arrays can be created using the following theorem on choosing the physical sensor locations. For more information, see P. Pal and P. Vaidyanathan, IEEE Transactions on Signal Processing 60, 1253 (2011).

[0034] **Theorem II.2.** *Let $q, N_1, N_2, \dots, N_{2q}$ be positive integers and consider the following sets for $1 \leq i \leq 2q - 1$*

$$\{n \prod_{k=0}^{i-1} N_k : n = 1, 2, \dots, N_i - 1\}, \quad (3)$$

and the $2q$ -th set is given by $\{n \prod_{k=0}^{2q-1} N_k : n = 1, 2, \dots, N_{2q}\}$ where $N_0 := 1$. Let the physical array be the union of all these sets. Then the $2q$ -th order virtual array corresponding to this physical array contains a ULA of size

$$2 \prod_{k=0}^{2q} N_k - 1. \quad (4)$$

[0035] As an example, choose $N_1 = N_2 = \dots = N_{2q} = 2$ so that the union of all the sets in equation (3) is equal to $(2^j)_{j \in [2q]}$ and the corresponding virtual array has size 2^{2q+1} . FIG.

1 shows an example where $q = 2$ and $N_1 = N_2 = N_3 = N_4 = 2$. So, the physical array is given by $\{1, 2, 4, 8\}$ (also including the location 0). The second-order virtual array is the next level. The locations of the second-order virtual array are obtained by taking all the possible differences of pairs in the physical array. The 4-th order virtual array contains a ULA of size thirty-one. Note that, in this example, only measurements at the physical array locations were taken, whose size (the number of locations) is significantly smaller than the virtual arrays (the second and fourth order arrays) which is quantified by the above Theorem.

III. THE AE ALGORITHM

[0036] Consider the quantum state from Eqn. 1, which is obtained by executing a quantum circuit that performs n sequential oracle calls of the operator G :

$$|\phi_n\rangle = \cos((2n+1)\theta)|x, 0\rangle + \sin((2n+1)\theta)|x', 1\rangle.$$

When the target (also “last”) qubit of the state $|\phi_n\rangle$ is measured, the quantum state of $|0\rangle$ is obtained with probability $p_0(n)$ and the quantum state of $|1\rangle$ is obtained with probability $p_1(n)$, where:

$$p_0(n) := \cos^2((2n+1)\theta) \text{ and } p_1(n) := \sin^2((2n+1)\theta). \quad (5)$$

When measuring $|\phi_n\rangle$ in the X -basis (e.g., perform n sequential oracle calls of the operator G , apply a Hadamard quantum gate to the target qubit, and then measure the target qubit), the quantum states of $|0\rangle$ or $|1\rangle$ are obtained with probabilities:

$$p_0^X(n) := \frac{1}{2}(1 + \sin((2n + 1)2\theta)) \quad (6)$$

$$p_1^X(n) := \frac{1}{2}(1 - \sin((2n + 1)2\theta)) \quad (7)$$

so that

$$\frac{p_0^X(n) - p_1^X(n)}{p_0(n) - p_1(n)} = \tan((2n + 1)2\theta). \quad (8)$$

Approximations of $p_i(n)$ and $p_i^X(n)$ may be obtained by taking measurements of the target qubit and counting the total number of 0/1 measurement results relative to the total number of measurements. In general, more measurements result in higher accuracy estimates of the probability but at the expense of more oracles queries. In one example, for each depth take $N_{shots}(n) \sim c / \log n$, where $c > 1$ is some constant. An approximation of $(2n + 1)2\theta$ may be determined by taking the arctan of the left side of Eqn. 8. We can form the exponential $y(n) = e^{i(2n+1)2\theta} = e^{in4\theta+2i\theta}$ that we view as an incoming signal measured using a sensor placed at position n . Quantumly this may be done by measuring the quantum state of the target qubit of the quantum state $|\phi_n\rangle$. DOA algorithms are blind to the extra phase factor $e^{2i\theta}$ in $y(n)$, meaning it will extract just 4θ from measurements of $y(n)$.

[0037] In classical signal processing terms, a measurement of the signal $y(n) = e^{in4\theta+2i\theta}$ may be obtained by placing a physical sensor at location n . Quantumly, a measurement of the signal $y(n)$ may be obtained by taking repeated measurements of the target qubit of the quantum state $|\phi_n\rangle$ in the Z and X basis. Regarding the DOA algorithm (e.g., ESPRIT), it is agnostic to how the signal is obtained.

A. Determining Depths

[0038] This disclosure will now discuss at what depths to take measurements (“depths” and “measurements” were used as classical signal processing terms). Many DOA algorithms, such as the ESPRIT algorithm, expect measurements on a uniform linear array of length M , that is measurements of $y(n)$ for $n \in [M]$. Quantumly, one may obtain this by measuring the target qubit of the quantum state $|\phi_n\rangle$ at depths $n \in [M]$. However, this does not provide the expected quantum scaling $O(1/M)$ of the error, but rather the scaling is $O(1/\sqrt{M})$ in this case. This problem may be avoided by taking samples from a smaller set of depths and using the virtual array concept to obtain (virtual) measurements at all depths $n \in [M]$. For example, Theorem (II.2) may be applied to use the physical array given by the sequence $D =$

$(2^j)_{j \in [2q]}$ (however, other methods may be used to determine a set of depths). The query complexity of taking measurements of the target qubit of the quantum state $|\phi_n\rangle$ for all $n \in D$ is $O(2^{2q+1})$, while the size of the virtual array is also 2^{2q+1} by Theorem (II.2). Since the ESPRIT algorithm obtains the elevation angle with accuracy $O(1/M)$ if M is the length of the virtual array, this implies immediately the desired quantum scaling.

B. Example Algorithm

[0039] Example steps to implement amplitude estimation are provided below in the table labeled “Algorithm 2.” Furthermore, although Algorithm 2 uses ESPRIT in Step 4, other DOA estimation algorithms may be used at Step 4. Furthermore, the instruction in step 2 to “determine a set of depths such that the $2q$ -th order virtual array has size $M \approx 1/\epsilon$ ” is merely an example. There are other methods of determining a set of depths, such as the co-prime array sampling strategy previously discussed.

Algorithm 2 Amplitude Estimation from Classical Signal Processing (csAE)

Setup: Oracle U such that $U|0^l\rangle = \cos \theta |x, 0\rangle + \sin \theta |x', 1\rangle$ and error rate $\epsilon > 0$.

Goal: An approximation of $a = \cos(\theta)$ up to additive error ϵ .

- 1: Determine a set D of depths such that the $2q$ -th order virtual array has size $M \approx 1/\epsilon$.
 - 2: Estimate the state $|\phi_n\rangle$ for $n \in D$ in the Z and X basis and form the “signal” vector $y_n = e^{i\omega n}$ where ω_n is the arctan of the expression in (8) and where the $p_i(n)$ and $p_i^X(n)$ are replaced by their empirical values (from measurements of the target qubit).
 - 3: Form the signal vector $y \in \mathbb{C}^M$ by computing the signal values at the virtual locations.
 - 4: Use the ESPRIT algorithm with y as input to extract the angle $\hat{\omega}$.
 - 5: If $\hat{\omega} < 0$ return $\cos(\pi/2 - |\hat{\omega}|/4)$ and else return $\cos(\hat{\omega}/4)$.
-

IV. NUMERICS

[0040] In this section this disclosure shows the performance of an example implementation of the Amplitude Estimation from Classical Signal Processing (csAE) algorithm (e.g., Algorithm 2) and compares it against the state-of-the-art Amplitude Estimation algorithm “chebAE” (more information on chebAE can be found at P. Rall and B. Fuller, Quantum 7, 937 (2023)). This disclosure compares the algorithms on two different metrics: 1) query complexity and 2) parallel query complexity. Parallel query complexity refers to the number of times the oracle (G) is called in sequence. From the simulations conducted by the inventors, the csAE algorithm performs better on the query complexity and parallel query complexity. First, this disclosure compares the chebAE and csAE on these two metrics for estimating a fixed amplitude and varying estimation error.

[0041] FIG. 2A plots the query complexity of csAE and chebAE, and FIG. 2B plots the parallel complexity of csAE and chebAE, according to one or more embodiments. To make the plots of csAE, five hundred runs were simulated giving several different error rates (given by the vertical dots) and the triangles denoting the 95-th percentile of those runs. For chebAE, the code provided on github was used¹. The amplitude $a = 0.5$ is estimated with 95% confidence. The query complexity of chebAE is directly compared, represented by the stars, at the corresponding error rate. We also fit the function $N = C/\epsilon$ for these data points, where N represents the query complexity (or parallel query complexity) and ϵ the error rate. As illustrated, csAE outperforms chebAE at lower error rates except at the error rate $\sim 10^{-2}$ which is probably an outlier and would likely be fixed if the number of shots for this specific error rate were optimized. In terms of parallel query complexity, the csAE algorithm is at least one order of magnitude better.

[0042] FIGS. 3A and 3B plot the dependence of csAE and chebAE on the amplitude that is being estimated for a fixed order of magnitude error rate. Specifically, a target error rate is fixed, and the query complexity (FIG. 3A) and parallel query complexity (FIG. 3B) is plotted for different values of the amplitude. For chebAE an error rate is fixed, and the query complexity is obtained for different values of an amplitude (note: an average of two hundred runs was used and the error bars are also plotted). In the case of csAE, an array is specified by an integer n such that $1/2^n$ is roughly equal to the desired error rate. Therefore in this case, an integer n was selected, the csAE algorithm was run, and the resulting (95-percentile) error rates were computed. This results in csAE having a constant number of queries and time complexity. In FIGS. 3A and 3B, the 95-percentile error rates are (5.4×10^{-4} , 5.4×10^{-4} , 5.8×10^{-4} , 4.3×10^{-4} , 4.6×10^{-4} , 4.1×10^{-4} , 3.7×10^{-4} , 3.0×10^{-4} , 3.0×10^{-4}) for the amplitudes in this order.

A. Fits

[0043] Performing a one parameter model fit of the form $N = C/\epsilon$ (N number of queries and ϵ error rate at 95% confidence) for $\epsilon \in (7.9 \times 10^{-3}, 9.8 \times 10^{-6})$, the following constants for the query complexity and parallel query complexity are obtained at 95% confidence. For the csAE algorithm, the best constant fitting query complexity, we have $C = 3.73$, whereas for chebAE we have $C = 4.51$ with a maximum deviation of 16.55%. Fitting

¹ <https://github.com/qiskitcommunity/ChebAE/tree/main>

the parallel query complexity for the csAE algorithm, the inventors determined $C = 0.28$ while for chebAE the inventors determined $C = 4.27$.

VI. EXAMPLE METHODS

[0044] FIG. 4 is a flowchart of an example method 400 for estimating an amplitude of a first quantum oracle (e.g., U) using direction of arrival (DOA) estimation, according to one or more embodiments. In the example of FIG. 4, the method 400 is performed from the perspective of a computing system (e.g., 800) including one or more quantum computing systems (e.g., 820). The method 400 can include greater or fewer steps than described herein. Additionally, the steps can be performed in different order, or by different components than described herein. In some embodiments, the method 400 is performed by the computing system executing code stored on a (e.g., non-transitory) computer-readable storage medium that causes the computing system to perform the steps of method 400. Algorithm 2 is an example of method 400.

[0045] At step 405, the computing system determines a set of values (e.g., a set of positive non-zero integers without duplicates) that each correspond to a number of oracle calls of a second quantum oracle (e.g., the Grover operator G) to be performed (e.g., in sequence). For more example information, see Step 1 of Algorithm 2 and Section III.A. The second quantum oracle is based on the first quantum oracle (e.g., the second quantum oracle is the Grover operator G for the quantum oracle U).

[0046] For each value n in the set, the computing system performs steps 407-420 (e.g., see Step 2 of Algorithm 2).

[0047] At step 407, the computing system executes (e.g., by the quantum computing system), a first quantum circuit that performs n oracle calls of the second quantum oracle (e.g., in sequence) to generate a first output quantum state (e.g., on the target qubit (which is a qubit of the quantum computing system that executed the first quantum circuit)).

[0048] At step 410, the computing system estimates the first output quantum state. The estimate may be based on one or more measurements of the target qubit in a first basis (e.g., the Z basis). The one or more measurements may be taken by the quantum computing system(s) that executed the second quantum circuit. If multiple measurements of the target qubit are taken, step 407 may be performed before each measurement of the target qubit (e.g., a first measurement is taken after n oracle calls are performed and a second measurement is taken after another n oracle calls are performed).

[0049] At step 415, the computing system executes (e.g., by the quantum computing system or another quantum computing system) a second quantum circuit that performs n oracle calls of the second quantum oracle (e.g., in sequence) to generate a second output quantum state (e.g., on the target qubit (which is a qubit of the quantum computing system that executed the second quantum circuit)). In some embodiments, the second quantum circuit includes the first quantum circuit and a Hadamard quantum gate (e.g., applied to the target qubit after the n oracle calls).

[0050] At step 420, the computing system estimates the second quantum state. The estimate may be based on one or more measurements of the target qubit in a second basis (e.g., the X basis) different than the first basis. The one or more measurements may be taken by the quantum computing system(s) that executed the second quantum circuit. If multiple measurements of the target qubit are taken, step 415 may be performed before each measurement of the target qubit (e.g., a first measurement is taken after n oracle calls are performed and a second measurement is taken after another n oracle calls are performed).

[0051] At step 425, the computing system forms a (e.g., signal) vector based on the estimated output quantum states for each value n in the set (e.g., determined by the computing system performing steps 407-420 for each value n in the set). For example, see Steps 2-3 of Algorithm 2.

[0052] In some embodiments, the (e.g., signal) vector has the form $y_n = e^{i\omega_n}$, where ω_n is based on the estimated output quantum states for the value of n in the set. In some embodiments, ω_n is the arctan of: $\frac{p_0^X(n) - p_1^X(n)}{p_0(n) - p_1(n)}$, where $p_0(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the Z basis after performing n (e.g., sequential) oracle calls of the second quantum oracle, $p_1(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the Z basis after performing n (e.g., sequential) oracle calls of the second quantum oracle, $p_0^X(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the X basis after performing n (e.g., sequential) oracle calls of the second quantum oracle, and $p_1^X(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the X basis after performing n (e.g., sequential) oracle calls of the second quantum oracle. For more information, see Section III (e.g., Eqn. 8 and associated description).

[0053] In some embodiments, the computing system forming the (e.g., signal) vector includes the computing system forming an initial vector with initial vector components having the form y_n (e.g., $y_n = e^{i\omega_n}$), and includes the computing system expanding the

initial vector by computing additional vector components based on the initial vector components. In some embodiments, computing the additional vector components includes computing the outer product of the initial vector with itself. For example, the “initial vector” is $y_n = e^{i\omega_n}$ of Step 2 of Algorithm 2 and the “expanded initial vector” is the signal vector $y \in \mathbb{C}^M$ of Step 3 in Algorithm 2.

[0054] At step 430, the computing system applies the (e.g., signal) vector to the DOA estimation algorithm. For example, see Step 4 of Algorithm 2.

[0055] At step 435, the computing system estimates an amplitude of the first quantum oracle (e.g., U) based on output of the DOA estimation algorithm. For example, see Step 5 of Algorithm 2.

[0056] In some embodiments, estimating the amplitude of the first quantum oracle is not based on determinations from performing phase estimation. In some embodiments, the steps of method 400 do not include the computing system (or another computing system) performing phase estimation.

[0057] In some embodiments, one or more steps of method 400 are performed in parallel. For example, executing the first and second quantum circuits and measuring the corresponding output quantum states (e.g., steps 407-420) for a first value in the set is performed in parallel to executing the first and second quantum circuits and measuring the corresponding output quantum states (steps 407-420) for a second value in the set (different than the first value in the set). For example, steps 407-420 for a first value in the set are performed by a first quantum computing system of the computing system and steps 407-420 for a second value in the set are performed by a second quantum computing system (different than the first) of the computing system. In another example of parallelization, steps 407 and 410 for a first value in the set are performed by a first quantum computing system of the computing system and steps 415 and 420 for the first value in the set are performed by a second quantum computing system (different than the first) of the computing system.

[0058] Other aspects include components, devices, systems, improvements, methods, processes, applications, computer readable mediums, and other technologies related to any of the above.

VII. DESCRIPTION OF A COMPUTING SYSTEM

[0059] Embodiments described above may be implemented using one or more computing systems. Example computing systems are described below.

[0060] FIG. 5A is a block diagram that illustrates an embodiment of a computing system 800. In the example of FIG. 5A, the computing system 800 includes a classical computing system 810 (also referred to as a non-quantum computing system) and a quantum computing system 820, however a computing system may just include a classical computing system or a quantum computing system. An embodiment of the classical computing system 810 is described further with respect to FIG. 6. While the classical computing system 810 and quantum computing system 820 are illustrated together, they may be physically separate systems. For example, FIG. 5B illustrates an example cloud computing architecture where the computing system 810 and the quantum computing system 820 communicate via a network 857. The computing system 800 may include different or additional elements than illustrated (e.g., multiple quantum computing systems 820). In addition, the functions may be distributed among the elements in a different manner than described.

[0061] The classical computing system 810 may control the quantum computing system 820. For example, the classical computing system 810 generates and transmits instructions for the quantum computing system 820 to execute a quantum algorithm or quantum circuit. Although only one classical computing system 810 is illustrated in FIG. 5A, any number of classical computing system 810 or other external systems may be connected to the quantum computing system 820.

[0062] FIG. 5C is a block diagram that illustrates an embodiment of the quantum computing system 820. The quantum computing system 820 includes any number of quantum bits (“qubits”) 850 and associated qubit controllers 840. As illustrated in FIG. 5D, the qubits 850 may be in a qubit register of the quantum computing system 820 (or multiple registers). Qubits are further described below. A qubit controller 840 is a module that controls one or more qubits 850. A qubit controller 840 may include a classical processor such as a CPU, GPU, or FPGA. A qubit controller 840 may perform physical operations on one or more qubits 850 (e.g., it can perform quantum gate operations on a qubit 840). In the example of FIG. 5C, a separate qubit controller 840 is illustrated for each qubit 850, however a qubit controller 850 may control multiple (e.g., all) qubits 850 of the quantum computing system 820 or multiple controllers 850 may control a single qubit. For example, the qubit controllers 850 can be separate processors, parallel threads on the same processor, or some combination of both.

[0063] FIG. 5E is a flow chart that illustrates an example execution of a quantum routine on the computing system 800. The classical computing system 810 generates 860 a quantum program to be executed or processed by the quantum computing system 820. The

quantum program may include instructions or subroutines to be performed by the quantum computing system 820. In an example, the quantum program is a quantum circuit. The quantum computing system 820 executes 865 the program and computes 870 a result (referred to as a shot or run). Computing the result may include estimating a quantum state generated by the quantum computing system 820 that resulted from executing the program. Practically, this may be performed by measuring values of one or more of the qubits 850. The quantum computing system 820 typically performs multiple shots to accumulate statistics from probabilistic execution. The number of shots and any changes that occur between shots (e.g., parameter changes) may be referred to as a schedule. The schedule may be specified by the program. The result (e.g., quantum state data) (or accumulated results) is recorded 875 by the classical computing system 810. Results may be returned after a termination condition is met (e.g., a threshold number of shots occur). The classical computing system 810 may determine a quantity based on the received results.

[0064] The quantum computing system 820 exploits the laws of quantum mechanics in order to perform computations. A quantum processing device (QPU), a quantum computer, a quantum processor, and a quantum processing unit are each examples of a quantum computing system. The quantum computing system 800 can be a universal or a non-universal quantum computing system (a universal quantum computing system can execute any possible quantum circuit (subject to the constraint that the circuit doesn't use more qubits than the quantum computing system)). In some embodiments, the quantum computing system 800 is a gate model quantum computer. As previously described, quantum computing systems use so-called qubits, or quantum bits (e.g., 850A). While a classical bit always has a value of either 0 or 1, a qubit is a quantum mechanical system that can have a value of 0, 1, or a superposition of both values. Example physical implementations of qubits include superconducting qubits, spin qubits, trapped ions, arrays of neutral atoms, and photonic systems (e.g., photons in waveguides). For the purposes of this disclosure, a qubit may be realized by a single physical qubit or as an error-protected logical qubit that itself comprises multiple physical qubits. Additionally, the disclosure is not specific to qubits. The disclosure may be generalized to apply to quantum computing systems 820 whose building blocks are qudits (d -level quantum systems, where $d > 2$) or quantum continuous variables, rather than qubits.

[0065] A quantum circuit is an ordered collection of one or more gates. A sub-circuit may refer to a circuit that is a part of a larger circuit. A gate represents a unitary operation performed on one or more qubits. Quantum gates may be described using unitary matrices.

The depth of a quantum circuit is the least number of steps used to execute the circuit on a quantum computing system. The depth of a quantum circuit may be smaller than the total number of gates because gates acting on non-overlapping subsets of qubits may be executed in parallel. A layer of a quantum circuit may refer to a step of the circuit, during which multiple gates may be executed in parallel. In some embodiments, a quantum circuit is executed by a quantum computing system. In this sense, a quantum circuit can be thought of as comprising a set of instructions or operations that a quantum computing system can execute. To execute a quantum circuit on a quantum computing system, a user may inform the quantum computing system what circuit is to be executed. A quantum computing system may include both a core quantum device and a classical peripheral/control device (e.g., a qubit controller 840) that is used to orchestrate the control of the quantum device. It is to this classical control device that the description of a quantum circuit may be sent when one seeks to have a quantum computer execute a circuit.

[0066] The parameters of a parameterized quantum circuit may refer to parameters of the gates. For example, a gate that performs a rotation about the y axis may be parameterized by a real number that describes the angle of the rotation.

[0067] The description of a quantum circuit to be executed on one or more quantum computing systems may be stored in a non-transitory computer-readable storage medium. The term “computer-readable storage medium” should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, or associated caches and servers) able to store instructions. The term “computer-readable medium” shall also be taken to include any medium that is capable of storing instructions for execution by the quantum computing system and that cause the quantum computing system to perform any one or more of the methodologies disclosed herein. The term “computer-readable medium” includes, but is not limited to, data repositories in the form of solid-state memories, optical media, and magnetic media.

[0068] FIG. 6 is an example architecture of a classical computing system 810, according to an embodiment. The quantum computing system 820 may also have one or more components described with respect to FIG. 6. FIG. 6 depicts a high-level block diagram illustrating physical components of a computer system used as part or all of one or more entities described herein, in accordance with an embodiment. A computer may have additional, less, or variations of the components provided in FIG. 6. Although FIG. 6 depicts a computer 900, the figure is intended as functional description of the various features which may be present in computer systems than as a structural schematic of the implementations

described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated.

[0069] Illustrated in FIG. 6 is a set of one or more processors 902 coupled to a chipset 904. Also coupled to the chipset 904 are a memory 906, a storage device 908, a keyboard 910, a graphics adapter 912, a pointing device 914, and a network adapter 916. A display 918 is coupled to the graphics adapter 912. In one embodiment, the functionality of the chipset 904 is provided by a memory controller hub 920 and an I/O hub 922. In another embodiment, the memory 906 is coupled directly to the set of one or more processors 902 instead of the chipset 904. In some embodiments, the computer 900 includes one or more communication buses for interconnecting these components. The one or more communication buses optionally include circuitry (sometimes called a chipset) that interconnects and controls communications between system components.

[0070] The storage device 908 is any non-transitory computer-readable storage medium, such as a hard drive, compact disk read-only memory (CD-ROM), DVD, or a solid-state memory device or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Such a storage device 908 can also be referred to as persistent memory. The pointing device 914 may be a mouse, track ball, or other type of pointing device, and is used in combination with the keyboard 910 to input data into the computer 900. The graphics adapter 912 displays images and other information on the display 918. The network adapter 916 couples the computer 900 to a local or wide area network.

[0071] The memory 906 holds instructions and data used by the set of one or more processors 902. The memory 906 can be non-persistent memory, examples of which include high-speed random access memory, such as DRAM, SRAM, DDR RAM, ROM, EEPROM, flash memory.

[0072] As is known in the art, a computer 900 can have different or other components than those shown in FIG. 6. In addition, the computer 900 can lack certain illustrated components. In one embodiment, a computer 900 acting as a server may lack a keyboard 910, pointing device 914, graphics adapter 912, or display 918. Moreover, the storage device 908 can be local or remote from the computer 900 (such as embodied within a storage area network (SAN)).

[0073] As is known in the art, the computer 900 is adapted to execute computer program modules for providing functionality described herein. As used herein, the term

“module” refers to computer program logic utilized to provide the specified functionality. Thus, a module can be implemented in hardware, firmware, or software. In one embodiment, program modules are stored on the storage device 908, loaded into the memory 906, and executed, individually or collectively, by a set of one or more processors (e.g., 902).

VIII. Additional Considerations

[0074] The disclosure above describes example embodiments for purposes of illustration only. Any features that are described as essential, important, or otherwise implied to be required should be interpreted as only being required for that embodiment and are not necessarily included in other embodiments.

[0075] Some portions of above disclosure describe the embodiments in terms of algorithmic processes or operations. These algorithmic descriptions and representations are commonly used by those skilled in the computing arts to convey the substance of their work effectively to others skilled in the art. These operations, while described functionally, computationally, or logically, are understood to be implemented by computer programs comprising instructions for execution by a processor or equivalent electrical circuits, microcode, or the like. Furthermore, it has also proven convenient at times, to refer to these arrangements of functional operations as modules, without loss of generality. In some cases, a module can be implemented in hardware, firmware, or software.

[0076] As used herein, any reference to “one embodiment” or “an embodiment” means that a particular element, feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment. The appearances of the phrase “in one embodiment” in various places in the specification are not necessarily all referring to the same embodiment. Similarly, use of “a” or “an” preceding an element or component is done merely for convenience. This description should be understood to mean that one or more of the elements or components are present unless it is obvious that it is meant otherwise. As used herein, the terms “comprises,” “comprising,” “includes,” “including,” “has,” “having” or any other variation thereof, are intended to cover a non-exclusive inclusion. For example, a process, method, article, or apparatus that comprises a list of elements is not necessarily limited to only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. Further, unless expressly stated to the contrary, “or” refers to an inclusive or and not to an exclusive or. For example, a condition A or B is satisfied by any one of the following: A is true (or present) and B is false (or not present), A is false (or not present) and B is true (or present), and both A and B are true (or present).

[0077] In addition, use of the “a” or “an” are employed to describe elements and components of the embodiments. This is done merely for convenience and to give a general sense of the disclosure. This description should be read to include one or at least one and the singular also includes the plural unless it is obvious that it is meant otherwise. Where values are described as “approximate” or “substantially” (or their derivatives), such values should be construed as accurate +/- 10% unless another meaning is apparent from the context. From example, “approximately ten” should be understood to mean “in a range from nine to eleven.”

[0078] Alternative embodiments are implemented in computer hardware, firmware, software, and/or combinations thereof. Implementations can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps can be performed by a programmable processor executing a program of instructions to perform functions by operating on input data and generating output. As used herein, ‘processor’ may refer to one or more processors. Embodiments can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random-access memory. Generally, a computer will include one or more mass storage devices for storing data files; such devices include magnetic disks, such as internal hard disks and removable disks; magneto-optical disks; and optical disks. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits) and other forms of hardware.

[0079] Although the above description contains many specifics, these should not be construed as limiting the scope of the disclosure but merely as illustrating different examples. It should be appreciated that the scope of the disclosure includes other embodiments not

discussed in detail above. Various other modifications, changes, and variations which will be apparent to those skilled in the art may be made in the arrangement, operation, and details of the methods and apparatuses disclosed herein without departing from the spirit and scope of the disclosure.

WHAT IS CLAIMED IS:

1. A non-transitory computer-readable storage medium storing instructions for estimating an amplitude of a first quantum oracle using a direction of arrival (DOA) estimation algorithm, the instructions, when executed by a computing system including a quantum computing system, causes the computing system to perform operations comprising:
 - determining a set of values each corresponding to a number of oracle calls of a second quantum oracle to be performed, the second quantum oracle based on the first quantum oracle;
 - for each value n in the set:
 - executing a first quantum circuit that performs n oracle calls of the second quantum oracle to generate a first output quantum state;
 - estimating the first output quantum state based on a measurement of the target qubit in a first basis;
 - executing a second quantum circuit that performs n oracle calls of the second quantum oracle to generate a second output quantum state; and
 - estimating the second output quantum state based on a measurement of the target qubit in a second basis different than the first basis;
 - forming a signal vector based on the estimated output quantum states for each value in the set;
 - applying the signal vector to the DOA estimation algorithm; and
 - estimating an amplitude of the first quantum oracle based on an output of the DOA estimation algorithm.
2. The non-transitory computer-readable storage medium of claim 1, wherein the operations do not include performing phase estimation to estimate the amplitude of the first quantum oracle.
3. The non-transitory computer-readable storage medium of claim 1, wherein executing the first and second quantum circuits and measuring the corresponding output quantum states for a first value in the set is performed in parallel to executing the first and second quantum circuits and measuring the corresponding output quantum states for a second value in the set.
4. The non-transitory computer-readable storage medium of claim 1, wherein the second quantum circuit includes the first quantum circuit and a Hadamard quantum gate.
5. The non-transitory computer-readable storage medium of claim 1, wherein the signal vector has the form $y_n = e^{i\omega_n}$, where ω_n is based on the estimated output quantum states for the value of n in the set.

6. The non-transitory computer-readable storage medium of claim 5, wherein ω_n is the arctan of: $\frac{p_0^X(n) - p_1^X(n)}{p_0(n) - p_1(n)}$, where $p_0(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the Z basis after performing n oracle calls of the second quantum oracle, $p_1(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the Z basis after performing n oracle calls of the second quantum oracle, $p_0^X(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the X basis after performing n oracle calls of the second quantum oracle, and $p_1^X(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the X basis after performing n oracle calls of the second quantum oracle.
7. The non-transitory computer-readable storage medium of claim 1, wherein forming the signal vector comprises:
 - forming an initial vector with initial vector components having the form y_n ; and
 - expanding the initial vector by computing additional vector components based on the initial vector components.
8. The non-transitory computer-readable storage medium of claim 7, wherein computing the additional vector components comprises computing the outer product of the initial vector with itself.
9. The non-transitory computer-readable storage medium of claim 1, wherein executing the first quantum circuit is performed by a first quantum computer and executing the second quantum circuit is performed by a second quantum computer different than the first quantum computer.
10. The non-transitory computer-readable storage medium of claim 1, wherein DOA estimation algorithm is the MULTiple Signal Classification (MUSIC) algorithm or the Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) algorithm.
11. A method for estimating an amplitude of a first quantum oracle using a direction of arrival (DOA) estimation algorithm, the method comprising:
 - determining a set of values each corresponding to a number of oracle calls of a second quantum oracle to be performed, the second quantum oracle based on the first quantum oracle;
 - for each value n in the set:
 - causing a quantum computing system to execute a first quantum circuit that performs n oracle calls of the second quantum oracle to generate a first output quantum state;

- causing the quantum computing system to measure the target qubit in a first basis;
 estimating the first output quantum state based on the measurement of the target qubit in the first basis;
 causing the quantum computing system or a different quantum computing system to execute a second quantum circuit that performs n oracle calls of the second quantum oracle to generate a second output quantum state; and
 causing the quantum computing system or the different quantum computing system to measure the target qubit in a second basis different than the first basis;
 estimating the second output quantum state based on the measurement of the target qubit in the second basis;
 forming a signal vector based on the measured outputs for each value in the set;
 applying the signal vector to the DOA estimation algorithm; and
 estimating an amplitude of the first quantum oracle based on an output of the DOA estimation algorithm.
12. The method of claim 11, wherein estimating an amplitude of the first quantum oracle is performed without phase estimation.
13. The method of claim 11, wherein execution of the first and second quantum circuits and for a first value in the set is performed in parallel to execution the first and second quantum circuits for a second value in the set.
14. The method of claim 11, wherein the second quantum circuit includes the first quantum circuit and a Hadamard quantum gate.
15. The method of claim 11, wherein the signal vector has the form $y_n = e^{i\omega_n}$, where ω_n is based on the estimated output quantum states for the value of n in the set.
16. The method of claim 15, wherein ω_n is the arctan of: $\frac{p_0^X(n) - p_1^X(n)}{p_0(n) - p_1(n)}$, where $p_0(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the Z basis after performing n oracle calls of the second quantum oracle, $p_1(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the Z basis after performing n oracle calls of the second quantum oracle, $p_0^X(n)$ is the estimated probability of measuring the target qubit in the $|0\rangle$ state in the X basis after performing n oracle calls of the second quantum oracle, and $p_1^X(n)$ is the estimated probability of measuring the target qubit in the $|1\rangle$ state in the X basis after performing n oracle calls of the second quantum oracle.
17. The method of claim 11, wherein forming the signal vector comprises:

forming an initial vector with initial vector components having the form y_n ; and
expanding the initial vector by computing additional vector components based on the
initial vector components.

18. The method of claim 17, wherein computing the additional vector components comprises computing the outer product of the initial vector with itself.
19. The method of claim 11, wherein executing the first quantum circuit is performed by a first quantum computer and executing the second quantum circuit is performed by a second quantum computer different than the first quantum computer.
20. The method of claim 11, wherein DOA estimation algorithm is the Multiple Signal Classification (MUSIC) algorithm or the Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) algorithm.