Partially Unitary Learning

Mikhail Gennadievich Belov*

Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, GSP-1, Moscow, Vorob'evy Gory, 119991, Russia

Vladislav Gennadievich Malyshkin †

Ioffe Institute, Politekhnicheskaya 26, St Petersburg, 194021, Russia

(Dated: May, 14, 2024)

`$Id: PartiallyUnitaryLearning.tex,v 1.400 2024/11/14 19:45:25 mal Exp $`

The problem of an optimal mapping between Hilbert spaces IN of $|\psi\rangle$ and OUT of $|\phi\rangle$ based on a set of wavefunction measurements (within a phase) $\psi_l \to \phi_l$, $l = 1 \ldots M$, is formulated as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \left|\langle\phi_l|\mathcal{U}|\psi_l\rangle\right|^2$ subject to probability preservation constraints on $\mathcal{U}$ (partial unitarity). The constructed operator $\mathcal{U}$ can be considered as an IN to OUT quantum channel; it is a partially unitary rectangular matrix (an isometry) of dimension $\dim(OUT) \times \dim(IN)$ transforming operators as $A^{OUT} = \mathcal{U} A^{IN} \mathcal{U}^\dagger$. An iterative algorithm for finding the global maximum of this optimization problem is developed, and its application to a number of problems is demonstrated. A software product implementing the algorithm is available from the authors.

* mikhail.belov@tafs.pro

† malyshki@ton.ioffe.ru

Dedicated to Professor Arthur McGurn on the occasion of his 75th birthday.

I. INTRODUCTION

Progress in machine learning (ML) knowledge representation, from linear regression coefficients, perceptron weights[1], statistical learning[2], and logical approaches[3] to support vector machines[4], rules and decision trees[5], fuzzy logic[6, 7], and deep learning[8] has defined the direction of ML development over the last four decades. Recently, knowledge representation in the form of a unitary operator has started to attract significant attention[9–11]. The problem of learning unitary matrices is also useful in various other fields. For example it can be applied to the quantum mechanics inverse problem[12, 13], investigating the dynamics of quantum many-body systems[14–18], quantum computing[19–22], light coherence[23], market dynamics[24] and other fields.

The techniques used for unitary learning differ in unitary matrix representation, input data, and quality criteria. A substantial number of existing works[23, 25, 26] use the Frobenius L^2 norm of the difference between the target and current matrix, $\|\mathcal{U}-\mathcal{V}\|^2$. The main advantage of this approach is the applicability of first-order gradient optimization, but it has all the limitations of L^2 minimization approaches. A better option is to use the fidelity of the target states $|\langle\phi\,|\,\mathcal{U}\,|\,\psi\rangle|^2$; this approach is utilized in [9, 27] and many others. An important advantage of this approach is that the multiplication of the source $|\psi\rangle$ or target $|\phi\rangle$ by a random phase does not change the fidelity. A disadvantage is that the operator $\mathcal{U}$ itself can only be determined within a phase. There is a very interesting approach to representing a unitary matrix of dimension n in a recursive way by splitting it into $[n/2]$-sized matrices and continuing the process recursively [28]. This includes a unitary representation $\mathcal{U}\approx V_1(U_A\otimes U_B)V_0$, which requires the introduction of pre- and post-processing operators V_1 and V_0, allowing for the identification of an appropriate cost function.

The algorithm developed in this paper is applicable only when the objective function is a quadratic function of $\mathcal{U}$, as in the form given by the fidelity (5) or the cost function (37) below. This form allows the optimization problem to be formulated as a novel algebraic problem (39) and avoids the difficult challenge of unitary matrix parametrization. In our previous work[29] the problem of unitary learning was generalized to partially unitary operators. This operator maps two Hilbert spaces of different dimensions, whereas a unitary operator maps

a Hilbert space into itself. The problem is to maximize the fidelity of a mapping between the Hilbert spaces IN of $|\psi\rangle$ and OUT of $|\phi\rangle$ based on a set of wavefunction measurement (within a phase) observations $\psi_l \to \phi_l$, $l = 1 \dots M$, as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \left|\langle \phi_l \left| \mathcal{U} \right| \psi_l \rangle\right|^2$ subject to probability preservation constraints on $\mathcal{U}$ (partial unitarity). This problem is reduced to a problem of maximizing a quadratic form on $\mathcal{U}$ subject to quadratic form constraints. This is a variant of the QCQP (Quadratically Constrained Quadratic Program) problem. This problem is non-convex, exhibiting local extrema and multiple saddle points. In this work, an iterative algorithm for finding the global maximum of this optimization problem is developed, and its application to a number of problems is demonstrated.

The quantum mechanics inverse problem of reconstructing $\mathcal{U}$ from measured wavefunction observations $\psi_l \to \phi_l$, $l = 1 \dots M$ is transformed into a new algebraic problem (39). This problem is analogous to the Schrödinger equation, where instead of a Hamiltonian, there is a superoperator S, the "eigenvector" $\mathcal{U}$ corresponds to a unitary operator, and the "eigenvalue" λ corresponds to a Hermitian matrix. The solution to the quantum mechanics inverse problem can be found by solving this Schrödinger-like equation. This represents the most important result of the study. Currently, only a numerical solution is available. For software availability, please refer to Appendix C; all references to code in the paper correspond to this software.

II. FORMULATION OF THE PROBLEM

Consider the quantum mechanics inverse problem. It may be broadly described as a problem of determining the internal structure (e.g. Hamiltonian) of a system from wavefunction measurements. A number of other problems in statistics, machine learning, data analysis, etc., can be converted to a problem of the following form. Let there be two Hilbert spaces of dimensions n and D, corresponding to states $\psi(\mathbf{x})$ and $\phi(\mathbf{f})$ in some $\mathbf{x}$- and $\mathbf{f}$- bases x_k and f_j where $D \leq n$

$$\psi(\mathbf{x}) = \sum_{k=0}^{n-1} \alpha_k x_k \tag{1a}$$

$$\phi(\mathbf{f}) = \sum_{j=0}^{D-1} \beta_j f_j \tag{1b}$$

There is a scalar product operation $\langle\cdot\rangle$ in each Hilbert space allowing to calculate scalar product inside the basis $\langle\psi\,|\,\psi'\rangle = \sum_{k,q=0}^{n-1} \alpha_k^* \langle x_k\,|\,x_q\rangle \alpha'_q$ and $\langle\phi\,|\,\phi'\rangle = \sum_{j,i=0}^{D-1} \beta_j^* \langle f_j\,|\,f_i\rangle \beta'_i$, but *not across the bases*: the $\langle x_k\,|\,f_j\rangle$ cannot be calculated! Assume we have $l = 1 \ldots M$ wavefunction pairs (typically $M \gg n$) as "observations":

$$\psi_l(\mathbf{x}) \to \phi_l(\mathbf{f}) \qquad\qquad \text{weight } \omega^{(l)} \tag{2}$$

$$1 = \langle\psi_l\,|\,\psi_l\rangle = \langle\phi_l\,|\,\phi_l\rangle \tag{3}$$

These represent M mappings from Hilbert space $\mathbf{x}$ to Hilbert space $\mathbf{f}$. The weights $\omega^{(l)}$ are typically all equal to 1. However, they can be set to different values if the observations are made with different accuracy, this is particularly convenient for classical systems. Another use of the weights is the possible generalization from a discrete sum $\sum_{l=1}^{M} \omega^{(l)}$ to a general measure $\int d\omega$.[1]

The inverse problem is to find a partially unitary operator $\mathcal{U}$ (a matrix u_{jk} of dimension $D \times n$) acting from $\mathbf{x}$ to $\mathbf{f}$

$$f_j = \sum_{k=0}^{n-1} u_{jk} x_k \qquad\qquad j = 0 \ldots D-1 \tag{4}$$

that maximizes the total probability (fidelity)

$$\mathcal{F} = \sum_{l=1}^{M} \omega^{(l)} \Big| \langle\phi_l\,|\,\mathcal{U}\,|\,\psi_l\rangle \Big|^2 \underset{\mathcal{U}}{\longrightarrow} \max \tag{5}$$

$$\langle f_j\,|\,f_{j'}\rangle = \sum_{k,k'=0}^{n-1} u_{jk} \langle x_k\,|\,x_{k'}\rangle u^*_{j'k'} \tag{6}$$

subject to scalar product preservation (6) where $j, j' = 0 \ldots D-1$. The operator $\mathcal{U}$ acts from Hilbert space $\mathbf{x}$ to Hilbert space $\mathbf{f}$, and it can be viewed as a memoryless quantum channel. The functional (5) has matrix element [2] $\langle\phi_l\,|\,\mathcal{U}\,|\,\psi_l\rangle$ absolute value squared, which allows expressing

[1] Some authors use $\omega^{(l)} = 1/M$ to normalize the fidelity to the range $[0:1]$. However, this approach is inconvenient because $\mathcal{F}$ is no longer an extensive quantity in the sense that, for two sets of observations, the total fidelity is no longer the sum of the two — losing its additive property. For this reason, normalizing to the number of observations is preferred over normalizing to $[0:1]$.

[2] Note that the bra-ket notation in $\langle\phi\,|\,\mathcal{U}\,|\,\psi\rangle$ (and (3)) may differ from the measured sample average, such as (10) (which is the sum over $l = 1 \ldots M$). They are the same in Section II B but different in other examples; this is an important advance from [30], where two averages were always considered the same. For example, if in (6) the $\langle f_j\,|\,f_{j'}\rangle$ and $\langle x_k\,|\,x_{k'}\rangle$ are considered as sample averages, the Gram matrix quantum channel of Section II B is obtained. If they are postulated to be unit matrices, the traditional unitary learning of Section II D is obtained. The $S_{jk;j'k'}$ is always calculated from the measured sample as the sum over $l = 1 \ldots M$.

the fidelity only with $\langle f_j \,|\, f_{j'} \rangle$, $\langle x_k \,|\, x_{k'} \rangle$ and fourth order terms $\langle f_j x_k \,|\, f_{j'} x_{k'} \rangle$, without using unavailable "cross-moments" $\langle x_k \,|\, f_j \rangle$. The functional also exhibits proper wavefunction phase invariance. A specific aspect of quantum measurement is that the wavefunction itself cannot be measured; only the squared modulus of the wavefunction (probability density) can be obtained from a measurement operation. The $|\psi|^2$ is observable, whereas ψ is not. Measured wavefunction pairs (2) should be considered as measured $|\psi|^2$ and $|\phi|^2$ with the square root applied. The obtained result is possibly multiplied by an unknown random phase $\exp(i\xi_l)$. The optimization problem (5) of finding the optimal quantum channel $\mathcal{U}$ is invariant with respect to random phases introduced to measured wavefunctions (2). Without loss of generality, we can consider the Hilbert space bases to be orthogonal:

$$\delta_{jj'} = \langle f_j \,|\, f_{j'} \rangle \tag{7a}$$

$$\delta_{kk'} = \langle x_k \,|\, x_{k'} \rangle \tag{7b}$$

if this is not the case — an orthogonalizing procedure can be applied, see Appendices A and E of [30] or one can simply apply an orthogonalization method of Gram–Schmidt type, see `com/polytechnik/algorithms/DemoRecoverMapping.java:GramSchmidtTest` for an implementation. For orthogonal bases, condition (6) is exactly the unitary condition if $n = D$. If $D < n$ we name it the condition of partial unitarity.

$$\delta_{jj'} = \sum_{k=0}^{n-1} u_{jk} u^*_{j'k} \qquad\qquad j, j' = 0 \ldots D-1 \tag{8}$$

The meaning of $\mathcal{F}$ (5) is a weighted sum (with $\omega^{(l)}$ weights) of the possible similarity between $|\mathcal{U}|\psi_l\rangle$ and $|\phi_l\rangle$. By construction, we cannot directly project the states belonging to different Hilbert spaces as $\langle \phi_l \,|\, \psi_l \rangle$. Additionally, such a direct projection will not be invariant with respect to random phases of measured wavefunctions. The only possible way to transform a state from $\mathbf{x}$ to $\mathbf{f}$ is to apply operator $\mathcal{U}$ (4). This is a quantum channel that links two different Hilbert spaces. The states $\psi_l(\mathbf{x})$ and $\phi_l(\mathbf{f})$ in basis (1) are defined with $l = 1 \ldots M$ coefficients $\alpha_k^{(l)}$ and $\beta_j^{(l)}$. For orthogonalized basis (7) the functional (5) takes the form

$$\mathcal{F} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u^*_{j'k'} \tag{9}$$

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} \beta_j^{(l)} \alpha_k^{(l)} \beta_{j'}^{(l)\,*} \alpha_{k'}^{(l)\,*} \tag{10}$$

This expression is obtained by converting $\psi_l(\mathbf{x})$ to a function in $\mathbf{f}$-space using (4) and then projecting it onto $\phi_l(\mathbf{f})$. For non-orthogonal bases (7), the expression for $S_{jk;j'k'}$ is more complicated and less convenient to use. The tensor $S_{jk;j'k'} = S^*_{j'k';jk}$ is Hermitian by construction. The original problem is now reduced to maximizing the functional in (9) subject to the partial unitarity constraint in (8). There are a number of practical problems that can be reduced to this optimization problem.

A. A Quantum System Time Evolution

Consider a quantum system with a time-independent Hamiltonian H. Its time evolution

$$\mathcal{U} = \exp\left[-i\frac{t}{\hbar}H\right] \tag{11}$$

$$\left|\psi^{(t)}\right\rangle = \left|\mathcal{U}\right|\psi^{(t=0)}\rangle \tag{12}$$

Assume we have a long sequence $l = 1 \dots M$ of system observations made equidistantly at time moments $t_l = \tau l$

$$\psi_l(\mathbf{x}) \to \psi_{l+1}(\mathbf{x}) \qquad\qquad \text{weight } \omega^{(l)} = 1 \tag{13}$$

Measured wavefunctions can possibly have random phases. Given the basis (1a), a measured sample is a sequence of $l = 1 \dots M$ coefficients $\alpha_k^{(l)}$ that define the actual wavefunction within a random phase $\exp(i\xi_l)$. To obtain the optimization problem of the previous section, we put $D = n$, $\beta_j^{(l)} = \alpha_k^{(l+1)}$, and $\mathcal{U}$ is a unitary operator of wavefunction time shift τ.

$$|\psi_{l+1}\rangle = |\mathcal{U}|\psi_l\rangle \tag{14}$$

The $S_{jk;j'k'}$ is then

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} \alpha_j^{(l+1)} \alpha_k^{(l)} \alpha_{j'}^{(l+1)\,*} \alpha_{k'}^{(l)\,*} \tag{15}$$

The result of the optimization problem (9) subject to the unitary ($n = D$) constraint (8) is a unitary time shift operator $\mathcal{U}$. To obtain the Hamiltonian from $\mathcal{U}$ requires taking the logarithm of a unitary matrix. This is a different problem that requires separate consideration[31].

$$H = i\frac{\hbar}{\tau}\ln\mathcal{U} \tag{16}$$

B. A Classical System $\mathbf{x} \to \mathbf{f}$ Vector Mapping

Consider classical system $\mathbf{x}^{(l)} \to \mathbf{f}^{(l)}$ vectors mapping with the weights $\omega^{(l)}$, $l = 1 \dots M$, and all measurements are real numbers.

$$(x_0, x_1, \dots, x_k, \dots, x_{n-1})^{(l)} \to (f_0, f_1, \dots, f_j, \dots, f_{D-1})^{(l)} \tag{17}$$

This is a mapping of some observable to observable classical measurements of real vectors, for example let $\mathbf{x}$ be attributes and $\mathbf{f}$ be class labels of some dataset used in machine learning classification problem of vector type. A few example of $\mathbf{f}(\mathbf{x})$ constructed models include linear regression, the Radon-Nikodym approximation[30], a logical function, a neural network model, etc.

We want to convert these data to $\psi_l(\mathbf{x}) \to \phi_l(\mathbf{f})$ from (2) to construct a partially unitary operation $\mathcal{U}$ converting a state from $\mathbf{x}$ to $\mathbf{f}$. Let us define an average in both Hilbert spaces as the sum over the data sample (17)

$$\langle h \rangle = \sum_{l=1}^{M} \omega^{(l)} h_l \tag{18}$$

$$G^{\mathbf{x}}_{kk'} = \langle x_k \,|\, x_{k'} \rangle \tag{19}$$

$$G^{\mathbf{f}}_{jj'} = \langle f_j \,|\, f_{j'} \rangle \tag{20}$$

where h is a function on $\mathbf{x}$ or $\mathbf{f}$. When (18) is applied to $h_l = x^{(l)}_k x^{(l)}_{k'}$ or $f^{(l)}_j f^{(l)}_{j'}$ we obtain the Gram matrix (19) or (20). For complex space h_l would be $x^{(l)}_k x^{(l)\,*}_{k'}$ or $f^{(l)}_j f^{(l)\,*}_{j'}$, but in this section x_k and f_j are all real. As we discussed above, the $\mathbf{x}$ and $\mathbf{f}$ bases can be always orthogonalized (7) with a basis linear transform; the Gram matrix is equal to the unit matrix in the case of an orthogonal basis. Note that with these classical data, we can calculate "cross-moments" $\langle x_k \,|\, f_j \rangle$. An example of a model that uses cross-moments is linear regression

$$\min_{\gamma_k} \left\langle \left| f_j - \sum_{k=0}^{n-1} \gamma_k x_k \right|^2 \right\rangle \tag{21}$$

$$f_j(\mathbf{x}) \approx \sum_{k,k'=0}^{n-1} x_k G^{\mathbf{x};\,-1}_{kk'} \langle f_j \,|\, x_{k'} \rangle \tag{22}$$

As we use quantum channel ideology, our model should not depend on "cross-moments", only the partially unitary operator $\mathcal{U}$ (4) can link $\mathbf{x}$ and $\mathbf{f}$.

To construct a wavefunction $\psi_{\mathbf{y}}(\mathbf{x})$ localized at $\mathbf{x}=\mathbf{y}$ consider a localized state $\psi_{\mathbf{y}}(\mathbf{x})$

$$\begin{aligned}\psi_{\mathbf{y}}(\mathbf{x}) &= \sqrt{K(\mathbf{y})}\sum_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} x_k \\ &= \frac{\sum\limits_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} x_k}{\sqrt{\sum\limits_{i,k=0}^{n-1} y_i G^{\mathbf{x};\,-1}_{ik} y_k}} = \frac{\sum\limits_{i=0}^{n-1}\psi^{[i]}(\mathbf{y})\psi^{[i]}(\mathbf{x})}{\sqrt{\sum\limits_{i=0}^{n-1}\left[\psi^{[i]}(\mathbf{y})\right]^2}} \end{aligned} \tag{23}$$

$$K(\mathbf{x}) = \frac{1}{\sum\limits_{i,k=0}^{n-1} x_i G^{\mathbf{x};\,-1}_{ik} x_k} = \frac{1}{\sum\limits_{i=0}^{n-1}\left[\psi^{[i]}(\mathbf{x})\right]^2} \tag{24}$$

The $\psi_{\mathbf{y}}(\mathbf{x})$ is a function on $\mathbf{x}$ localized at a given $\mathbf{y}$, it is a normalized reproducing kernel, $1=\langle\psi_{\mathbf{y}}\,|\,\psi_{\mathbf{y}}\rangle$. In (23) it is written in two bases: the original x_k, for which $\langle x_q x_k\rangle = G^{\mathbf{x}}_{qk}$, and in some orthogonalized basis $\left|\psi^{[i]}\right\rangle$, such that $\left\langle\psi^{[q]}\,\middle|\,\psi^{[k]}\right\rangle=\delta_{qk}$. The $K(\mathbf{x})$ is the Christoffel function. Localized states in $\mathbf{f}$-space can be obtained with n to D and x to f replacement. With these localized states, we obtain $\psi_l(\mathbf{x})\to\phi_l(\mathbf{f})$ mapping for classical data (17) with observation weights $\omega^{(l)}$, $l=1\ldots M$

$$\sqrt{K(\mathbf{x}^{(l)})}\sum_{i,k=0}^{n-1} x^{(l)}_i G^{\mathbf{x};\,-1}_{ik} x_k \to \sqrt{K(\mathbf{f}^{(l)})}\sum_{i,j=0}^{D-1} f^{(l)}_i G^{\mathbf{f};\,-1}_{ik} f_k \tag{25}$$

This mapping is actually an original $\mathbf{x}\to\mathbf{f}$ mapping (17) with $\mathbf{x}$ and $\mathbf{f}$ linearly transformed and normalized. Contrary to the original mapping, (25) can be considered as a mapping of two Hilbert space states with the scalar product (18). It is convenient to introduce the moments of Christoffel function products:

$$\begin{aligned}&\left\langle x_k f_j\,\middle|\,K^{(\mathbf{x})}K^{(\mathbf{f})}\,\middle|\,x_{k'}f_{j'}\right\rangle = \\ &\sum_{l=0}^{M}\omega^{(l)}\frac{x^{(l)}_k x^{(l)}_{k'}}{\sum\limits_{q,q'=0}^{n-1} x^{(l)}_q G^{\mathbf{x};\,-1}_{qq'} x^{(l)}_{q'}}\cdot\frac{f^{(l)}_j f^{(l)}_{j'}}{\sum\limits_{s,s'=0}^{D-1} f^{(l)}_s G^{\mathbf{f};\,-1}_{ss'} f^{(l)}_{s'}}\end{aligned} \tag{26}$$

Assuming the $\mathbf{x}$ and $\mathbf{f}$ bases are orthogonalized (7), the tensor (10) is then

$$S_{jk;j'k'} = \left\langle x_k f_j\,\middle|\,K^{(\mathbf{x})}K^{(\mathbf{f})}\,\middle|\,x_{k'}f_{j'}\right\rangle \tag{27}$$

We have now obtained, for classical data, an optimization problem (9) subject to the (8) partial unitarity constraint. The result is the operator $\mathcal{U}$ maximizing $\mathcal{F}$. This problem has the same mathematical form as the quantum problem of Section II A above. But there is one

important difference: where the Hilbert space inner product comes from. In the quantum problem, it is an original property of Hilbert spaces used to formulate the problem, but the functional $\mathcal{F}$ is obtained from measurement data. In the classical problem, both the scalar product in the Hilbert space and the functional $\mathcal{F}$ are obtained from measurement data.

To evaluate the result at some given point $\mathbf{y}$, construct the localized density (23) and transform it to $\mathbf{f}$-space with (4). By projecting it to a $\mathbf{g}$-localized state in $\mathbf{f}$-space, obtain the probability of a given outcome $\mathbf{g}$ (here $\psi_{\mathbf{g}}(\mathbf{f})$ is a $\mathbf{f}$-space state localized at $\mathbf{f}=\mathbf{g}$, similarly to an $\mathbf{x}$-state (23), i.e. in $g_i G^{\mathbf{f};-1}_{ij} f_j$ put f_j from (4) after that obtained expression is a function on x_k that can be coupled with (23); for orthogonal bases (7) obtain (27))

$$\langle \psi_{\mathbf{g}} \,|\, \mathcal{U} \,|\, \psi_{\mathbf{y}} \rangle^2 = \frac{\left| \sum\limits_{k=0}^{n-1} \sum\limits_{j,s=0}^{D-1} g_j G^{\mathbf{f};-1}_{js} u_{sk} y_k \right|^2}{\sum\limits_{j,j'=0}^{D-1} g_j G^{\mathbf{f};-1}_{jj'} g_{j'} \sum\limits_{k,k'=0}^{n-1} y_k G^{\mathbf{x};-1}_{kk'} y_{k'}} \tag{28}$$

The optimization result is the u_{jk} matrix, $j = 0 \ldots D-1; k = 0 \ldots n-1$. This operator, given some input state (such as a localized state $|\psi_{\mathbf{x}}\rangle$), uniquely (within a phase) finds the function in $\mathbf{f}$-space $|\mathcal{U}|\psi_{\mathbf{x}}\rangle$ (coefficients $a_j(\mathbf{x})$) that predicts the probability (28) of outcome $\mathbf{f}$:

$$P(\mathbf{f})\Big|_{\mathbf{x}} = \langle \psi_{\mathbf{f}} \,|\, \mathcal{U} \,|\, \psi_{\mathbf{x}} \rangle^2 = \frac{\left| \sum\limits_{j=0}^{D-1} a_j f_j \right|^2}{\sum\limits_{j,j'=0}^{D-1} f_j G^{\mathbf{f};-1}_{jj'} f_{j'}} \tag{29}$$

$$a_j = \sum_{s=0}^{D-1} \sum_{k=0}^{n-1} G^{\mathbf{f};-1}_{js} u_{sk} x_k \sqrt{K(\mathbf{x})} \tag{30}$$

the $\mathbf{f}$ is equal to the value of the outcome we are interested in determining the probability of. Given $\mathbf{x}$, the probability of an outcome $\mathbf{f}$ is a squared linear function on f_j multiplied by the Christoffel function $K(\mathbf{f})$. This form of probability, a squared linear function on f_j divided by a quadratic form on f_j, can be obtained from many different considerations; the difference between them lies in the coefficients a_j. The mapping (25) corresponds to pure states. For mixed states mapping, the probability will be a ratio of two general quadratic forms instead of a rank one matrix in the numerator (29).

For a simple demonstration of creating $\mathbf{x} \to \mathbf{f}$ classical mapping, refer to Section VII below.

C. Learning Unitary Dynamics

In nature most of the dynamic equations are equivalent to a sequence of infinitesimal unitary transformations: Newton, Maxwell, and Schrödinger equations. Consider a simple classical problem. Let there be an initial state vector $\mathbf{X}^{(0)}$ of unit L^2 norm and a unitary matrix $\mathcal{U}$. The operator $\mathcal{U}$ is applied to $\mathbf{X}^{(0)}$ M times:

$$\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)} \tag{31}$$

From this sequence, M observations $(\mathbf{x}, \mathbf{f})$ (17) are created by taking $(l, l+1)$ elements of the sequence and multiplying them by a random phase (or ± 1 for real space).

$$\mathbf{x}^{(l)} = \exp(i\xi_l)\mathbf{X}^{(l)} \tag{32a}$$

$$\mathbf{f}^{(l)} = \exp(i\zeta_l)\mathbf{X}^{(l+1)} \tag{32b}$$

The random phases make any $\mathbf{x} \leftrightarrow \mathbf{f}$ regression-type methods inapplicable.[3] From the unitary property of operator $\mathcal{U}$, we immediately obtain (6); it does not depend on random phases. Whereas in the quantum problem we have a Hilbert space with an inner product $\langle\cdot\rangle$, in this classical problem there is no built in inner product available. The only available average is the sum (18) over M observations; there is no any other average such as ensemble averages or quantum measurements. For unitary dynamic with $\omega^{(l)} = 1$, this sum represents the regular time-average. The problem is to determine the operator $\mathcal{U}$ from the sample (32), $l = 0 \ldots M-1$, that undergoes unitary time-evolution (31). The goal is to determine a matrix u_{jk} (with $D = n$) maximizing the quality criterion $\mathcal{F}$ (5).

The Gram matrices $G^{\mathbf{x}}_{kk'}$ (19) and $G^{\mathbf{f}}_{jj'}$ (20) are time-average. The functional $\mathcal{F}$ is also time-average of the (31) data

$$\mathcal{F} = \sum_{l=1}^{M} \omega^{(l)} \left| \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} X_j^{(l+1)} u_{jk} X_k^{(l)} \right|^2 \underset{u_{jk}}{\longrightarrow} \max \tag{33}$$

Using (32) obtain

$$S_{jk;j'k'} = \sum_{l=1}^{M} \omega^{(l)} f_j^{(l)} x_k^{(l)} f_{j'}^{(l)\,*} x_{k'}^{(l)\,*} \tag{34}$$

---

[3] Consider the data in (32). Then $\langle f_j \,|\, x_k \rangle = \sum_{l=1}^{M} \omega^{(l)} f_j^{(l)} x_k^{(l)\,*} \exp(i(\zeta_l - \xi_l))$ which depends on random phases ζ_l and ξ_l, and thus it cannot be used to reconstruct the system.

Random phases (± 1 for real space) in (32) do not affect $S_{jk;j'k'}$ and the Gram matrices $\langle x_k \,|\, x_{k'} \rangle$, $\langle f_j \,|\, f_{j'} \rangle$ as the phases cancel each other in probabilities. The Gram matrices are not necessary unit matrices. This can be changed by basis regularization. Introduce regularization transformations $R^{\mathbf{x}}$ and $R^{\mathbf{f}}$ such that

$$\mathfrak{x} = R^{\mathbf{x}} \mathbf{x} \tag{35a}$$

$$\mathfrak{f} = R^{\mathbf{f}} \mathbf{f} \tag{35b}$$

produce unit Gram matrices; this can be for example $R^{\mathbf{x}} = G^{\mathbf{x};-1/2}$, $R^{\mathbf{f}} = G^{\mathbf{f};-1/2}$ or any other method, such as Gram-Schmidt with pivoting or QR decomposition. Then transformation (4) can be written in the form

$$\mathfrak{f} = R^{\mathbf{f}} \mathcal{U} R^{\mathbf{x};-1} \mathfrak{x} \tag{36}$$

and we can consider an optimization problem for the operator $\widetilde{\mathcal{U}} = R^{\mathbf{f}} \mathcal{U} R^{\mathbf{x};-1}$ instead of for $\mathcal{U}$. The solution in the original basis is then $\mathcal{U} = R^{\mathbf{f};-1} \widetilde{\mathcal{U}} R^{\mathbf{x}}$, see `com/polytechnik/algorithms/DemoRecoverUnitaryMatrixFromSeq.java:getUFromConfigGramMatrixChannel` for an implementation.

A question can be asked: In the case where $D = n$ for the data (32), will an operator of system dynamics $\mathcal{U}$ having the matrix u_{jk} maximizing $\mathcal{F}$ subject to constraints (6) always be unitary? It depends on the data. For a dataset representing unitary dynamics (31) the Gram matrices $\langle x_k \,|\, x_{k'} \rangle$ and $\langle f_j \,|\, f_{j'} \rangle$ must be the same, since the unitary operator $\mathcal{U}$ preserves the scalar product. But if the data contain non-unitary contributions, the Gram matrices $\langle x_k \,|\, x_{k'} \rangle$, $\langle f_j \,|\, f_{j'} \rangle$ can be different, and this difference contributes to the non-unitarity of u_{jk}; the constraints (6) preserve the Gram matrix passing through the quantum channel. This is the meaning of the constraints: the Gram matrix must transform from the Hilbert space IN into the Hilbert space OUT like any other operator. The idea is to measure the Gram matrix from the sample in both Hilbert spaces, construct u_{jk} by solving the optimization problem, and then generalize the model, stating that any other operator converts in the same way (38). If operators other than the Gram matrix are available in both Hilbert spaces, they can be used to build a quantum channel in exactly the same manner. Consider the unit matrix transformation, which corresponds to traditional unitary learning.

D. Traditional Unitary Learning

Consider "traditional" unitary learning. For the data sample (32), it is postulated that the operator $\mathcal{U}$ (31) is unitary; hence the Gram matrix properties are irrelevant to the task. All observation vectors $\mathbf{x}$ and $\mathbf{f}$ are of unit L^2 norm and of the same dimension. The problem becomes the following: maximize $\mathcal{F}$ (9) subject to unitary $\mathcal{U}$, i.e. constraints (8) with $D = n$. This is exactly the problem considered in Section II C above, but with a quantum channel transforming (38) a unit matrix from Hilbert space IN into a unit matrix in Hilbert space OUT instead of a Gram matrix. Practically, this unit matrix quantum channel can be implemented exactly as considered above. The only difference is that no regularization (35) should be performed at all, since it is postulated that $\mathcal{U}$ must always be unitary. The $\mathbf{f}$ and $\mathbf{x}$ should be used directly as is (without regularization) when constructing $S_{jk;j'k'}$ (34), see `com/polytechnik/algorithms/DemoRecoverUnitaryMatrixFromSeq.java:getUFromConfigUnitMatrixChannel` for an implementation. Contrary to the Gram matrix quantum channel of the previous section, the $\mathcal{U}$ obtained with the unit matrix quantum channel is always unitary. For a simple demonstration of recovering u_{jk} from unitary dynamics data (32) refer to Section IV below.

This unitary learning considers $\mathbf{x}$ and $\mathbf{f}$ vectors to be of *the same* dimension. The approach can be directly generalized to partial unitarity. Assume a dataset is of $\mathbf{x} \to \mathbf{f}$ mapping, where all vectors $\mathbf{x}$ and $\mathbf{f}$ have unit L^2 norm, but the vectors $\mathbf{x}$ and $\mathbf{f}$ are of *different* dimensions: n and D respectively. We want to build a *partially unitary* operator $\mathcal{U}$ of dimension $D \times n$ that converts a vector from $\mathbf{x}$ to $\mathbf{f}$ preserving probability. The model makes direct assumption about $\mathcal{U}$ and the dataset, hence Gram matrix properties are irrelevant to the task. The problem becomes the following: maximize $\mathcal{F}$ (9) subject to partial unitary constraints (8), now with $D < n$. The calculations are identical to the problem where $D = n$ that we just considered. Calculate $S_{jk;j'k'}$ (34) using $\mathbf{x}$ and $\mathbf{f}$ directly, without regularization. Then optimize $\mathcal{F}$ subject to (8) with corresponding D and n. The obtained partially unitary operator $\mathcal{U}$ is a quantum channel transforming a unit matrix of dimension n in Hilbert space $\mathbf{x}$ to a unit matrix of dimension D in Hilbert space $\mathbf{f}$. For a simple demonstration, refer to Section VI below. This quantum channel maximizes the fidelity of mapping between Hilbert spaces of different dimensions. This can offer a completely new perspective on unitary machine learning models.

E. Variational quantum algorithms

Variational quantum algorithms[32], which use a classical optimizer to train a parametrized quantum circuit, are among the most promising applications of near-term quantum computers. They often have a cost function in the form[33]

$$C(\theta) = \mathrm{Tr}\, O\mathcal{U}(\theta)\rho_0\mathcal{U}^\dagger(\theta) \tag{37}$$

where O is a Hermitian operator. This expression is a quadratic function of the unitary operator $\mathcal{U}$. Expanding the trace, we obtain the cost function exactly in the form (40), subject to unitary constraints (41). The tensor $S_{jk;j'k'}$ is obtained directly from (37). Thus, the theory developed in this paper is directly applicable to variational quantum algorithms. The application of the developed algorithm is demonstrated for problems of dimension as large as 40; see Section IV below. The limitation now is not due to vanishing gradients or a too-flat cost function[33], but to computational complexity. Our optimization algorithm can be parallelized (see Appendix B below), and we see no significant difficulty preventing an increase in problem dimension.

F. Algebraic Structure of the Optimization Problem

The formulated optimization problem maximizes (9) subject to the partial unitarity constraint (8). This is a variant of the QCQP problem. We find an operator $\mathcal{U}$ that optimally transforms a state $|\psi(\mathbf{x})\rangle$ from Hilbert space IN (of dimension n) into a state $|\phi(\mathbf{f})\rangle$ from Hilbert space OUT (of dimension D). The operator is a rectangular matrix of dimension $D \times n$ that transforms an operator A from IN to OUT as

$$A^{OUT} = \mathcal{U}A^{IN}\mathcal{U}^\dagger \tag{38}$$

This transformation converts any operator between two Hilbert spaces, for example a pure state $A^{IN} = |\psi\rangle\langle\psi|$ into a pure state $A^{OUT} = |\mathcal{U}|\psi\rangle\langle\psi|\mathcal{U}^\dagger|$. For $D = n$ it is a trace preserving map and for $D < n$ it is a trace decreasing map quantum channel (see Fig. 1 for a demonstration); for a more general form, refer to Kraus operators (78) below. There should always be an operator known in both Hilbert spaces, which is used to create constraints on $\mathcal{U}$ when maximizing fidelity. These constraints (probability preservation) determine the specific form of partial unitarity. We consider two such operators: the Gram matrix (used throughout

most of this paper) and the unit matrix (traditional in unitary learning, as in Section II D, a unital map). The constrained optimization problem on $\mathcal{U}$ is reduced to a new algebraic problem (a variation of the Lagrangian $\mathcal{L}$ (49) is set to zero (65))

$$S\mathcal{U} = \lambda\mathcal{U} \tag{39}$$

which determines the operator $\mathcal{U}$. It is remotely similar to the stationary Schrödinger equation (eigenvalue problem), but instead of a Hamiltonian there is a superoperator S (represented by the tensor $S_{jk;j'k'}$). The "eigenvector" $\mathcal{U}$ is a partially unitary operator (represented by a $D \times n$ matrix u_{jk}), and the "eigenvalue" λ is a Hermitian matrix (represented by a $D \times D$ matrix λ_{ij}). The extremal functional value $\mathcal{F}$ is equal to the trace (the sum of diagonal elements) of λ. The algebraic structure of this eigenvalue–like problem, let us call it an "**eigenoperator**" problem, requires a separate study. Currently, we only have a numeric solution algorithm.

III. NUMERICAL SOLUTION

The problem is to optimize (40) subject to constraints (41)

$$\mathcal{F} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u^*_{j'k'} \underset{u}{\longrightarrow} \max \tag{40}$$

$$\delta_{jj'} = \sum_{k=0}^{n-1} u_{jk} u^*_{j'k} \qquad\qquad j, j' = 0 \ldots D-1 \tag{41}$$

The tensor $S_{jk;j'k'} = S^*_{j'k';jk}$ is Hermitian. For simplicity, we consider $S_{jk;j'k'}$ and u_{jk} to be real, and do not explicitly denote complex conjugation (*) below, generalization to complex values is straightforward. The matrix u_{jk} is an isometry, which means it has orthonormal rows but may not be a square matrix. If we consider a subset of constraints (41), the optimization problem can be readily solved. Consider the squared Frobenius norm of matrix u_{jk} to be a "simplified constraint":

$$\sum_{j=0}^{D-1} \sum_{k=0}^{n-1} u^2_{jk} = D \tag{42}$$

This is a partial constraint (it is the sum of all diagonal elements in (41)). For this partial constraint, the optimization problem (40) is equivalent to an eigenvalue problem — it can be directly solved by considering a vector of Dn dimension obtained from the u_{jk}, saving all its components into a single vector, row by row.

The main idea of [29] was to modify the partial constraint solution to satisfy the full set of constraints (41). There are several options to adjust the solution from a partial to the full set of constraints. The one producing the minimal change to the solution is the application of the Gram matrix

$$G^u_{jj'} = \sum_{k=0}^{n-1} u_{jk} u_{j'k} \tag{43}$$

inverse square root $G^{u;-1/2} = 1/\sqrt{G^u}$ to u_{jk}.[4] There are 2^D square roots of a positively definite symmetric matrix of dimension D, differing only in the $\pm$ signs. The simplest method to calculate it is to convert $G^u_{jj'}$ to diagonal form in the basis of its eigenvectors

$$\left|G^u\right|g^{[i]}\rangle = \lambda^{[i]}_G \left|g^{[i]}\right\rangle \tag{44}$$

then change the eigenvalues to $\pm 1 \Big/ \sqrt{\lambda^{[i]}_G}$ and convert the matrix back to the initial basis

$$\left\|G^{u;-1/2}\right\| = \sum_{i=0}^{D-1} \frac{\pm 1}{\sqrt{\lambda^{[i]}_G}} \left|g^{[i]}\right\rangle \left\langle g^{[i]}\right| \tag{45}$$

By checking the result, one can verify that for any u_{jk} producing a non-degenerated Gram matrix (43), the vector

$$\widetilde{u}_{jk} = \sum_{i=0}^{D-1} G^{u;-1/2}_{ji} u_{ik} \tag{46}$$

satisfies all the constraints (41) (the most general form of adjustment is an application of $UG^{u;-1/2}$ to u_{ik}, where U is an arbitrary unitary operator; the $\pm$ square root signs can be included in U; below we consider all signs in (45) to be "+"), see `com/polytechnik/kgo/AdjustedStateToUnitaryWithEV.java` for an implementation. Thus, the multiple constraints optimization problem (40) can be reduced to an unconstrained optimization problem (we use the identity $D = \sum_{j=0}^{D-1} \sum_{k=0}^{n-1} u^2_{jk} = \sum_{j,j'=0}^{D-1} \sum_{k=0}^{n-1} u_{jk} G^{u;-1}_{jj'} u_{j'k}$):

$$\mathcal{F} = \frac{\sum\limits_{j,j',i,i'=0}^{D-1} \sum\limits_{k,k'=0}^{n-1} u_{jk} G^{u;-1/2}_{ji} S_{ik;i'k'} G^{u;-1/2}_{j'i'} u_{j'k'}}{\frac{1}{D} \sum\limits_{j=0}^{D-1} \sum\limits_{k=0}^{n-1} u^2_{jk}} \underset{u}{\longrightarrow} \max \tag{47}$$

[4] See Appendix A of [29] for adjustments in different bases and for an approach that uses SVD instead of the eigenproblem (44). Also note that both SVD-based and eigenproblem-based transformations convert a single u_{jk} state satisfying the partial constraint (42) to a state satisfying the full set of constraints (41). The problem of converting *several* $u^{[s]}_{jk}$ states satisfying a partial constraint to a single state satisfying the full set of constraints can greatly improve the initial convergence of the algorithm. This is a subject of future research.

However, this unconstrained problem[5]

- cannot be reduced to an eigenvalue problem since $G^{u;-1/2}_{ji}$ itself depends on u_{jk}.
- is degenerate: there are multiple u_{jk} producing the same $\mathcal{F}$. Convert a solution u_{jk} to the Gram matrix basis (44), change the eigenvalues, then convert it back to the initial basis, this is similar to the transformation in (45). Hence the Hessian matrix is degenerated, which prevents us from directly applying Newton and quasi-Newton optimization methods. One can possibly use a penalty function like $\sum 1/\lambda_G$

$$\mathrm{Tr}G^{u;-1} = \sum_{i=0}^{D-1} \frac{1}{\lambda_G^{[i]}} \tag{48}$$

  which has the minimum value D when all the constraints (41) are satisfied.

Alternatively, an iterative approach with Lagrange multipliers can be used. Iterations are required since we cannot simultaneously solve the equation for u_{jk} and $\lambda_{jj'}$. A single iteration consists in solving an eigenvalue problem, adjusting the obtained solution to satisfy the full set of constraints, and calculating the new values of Lagrange multipliers. There are three main elements of the algorithm:

- The eigenproblem solution (57) is used to solve the partially constrained optimization problem. An important feature is that any additional linear constraints on u_{jk} of the form (58) can be easily incorporated — obtain the eigenproblem (64).
- The solution adjustment operation (46) converts a solution satisfying the partial constraint (42) into one satisfying the full set (41).
- The linear system solution (53) obtains the new values of Lagrange multipliers.

In its naïve form (without additional linear constraints) a convergence of the iterative algorithm turned out to be poor. The major new result of the current paper is this iterative algorithm with *good convergence*. Good convergence was achieved by considering additional

[5] There are alternative ways to formulate an unconstrained optimization problem. In the unitary case where $D = n$, one can use $D(D+1)/2$ parameters to parametrize a Hermitian matrix. A unitary matrix is then obtained through matrix exponentiation (11); functional optimization, however, requires the derivatives of the matrix exponent, which complicates the problem[34, 35]. There is an option to parametrize a unitary matrix using the Cayley transform $U = (I - A)(I + A)^{-1}$ for a skew-Hermitian matrix $A^\dagger = -A$, see for example [36, 37]. Both methods are problematic to use, especially in the case of rectangular u_{jk} with $D < n$. See also [28, 38] for other methods of unitary parametrization.

linear constraints in the eigenproblem at each iteration step. Consider Lagrange multipliers $\lambda_{jj'}$, a matrix of dimension $D \times D$, to approach optimization problem (40) with constraints (41)

$$\mathcal{L} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} S_{jk;j'k'} u_{j'k'} + \sum_{j,j'=0}^{D-1} \lambda_{jj'} \left[\delta_{jj'} - \sum_{k'=0}^{n-1} u_{jk'} u_{j'k'} \right] \underset{u}{\longrightarrow} \max \tag{49}$$

Variating $\mathcal{L}$ over u_{sq} obtain (51)

$$b_{sq} = \sum_{j'=0}^{D-1} \sum_{k'=0}^{n-1} S_{sq;j'k'} u_{j'k'} \tag{50}$$

$$0 = b_{sq} - \sum_{j'=0}^{D-1} \lambda_{sj'} u_{j'q} \tag{51}$$

Here and below we consider the Lagrange multipliers matrix to be Hermitian $\lambda_{jj'} = \lambda^*_{j'j}$. This condition ensures the existence of an extremal solution[29]. The variation (51) contains Dn equations. The Hermitian Lagrange multipliers matrix $\lambda_{jj'}$ has D^2 real parameters ($D(D+1)/2$ independent ones) for real space and $2D^2$ real parameters (D^2 independent ones) for complex space. Thus, for a general u_{jk}, the variation (51) cannot be fully satisfied. The most straightforward way to obtain Lagrange multipliers for a given u_{jk} is to take the L^2 norm of the variation (51) and obtain the λ_{ij} that minimizes the sum of squares.

$$\sum_{i=0}^{D-1} \sum_{q=0}^{n-1} \left| b_{iq} - \sum_{j=0}^{D-1} \frac{\lambda_{ij} + \lambda_{ji}}{2} u_{jq} \right|^2 \underset{\lambda_{ij}}{\longrightarrow} \min \tag{52}$$

$$\lambda_{ij} = \mathrm{Herm} \sum_{k=0}^{n-1} u_{ik} b_{jk} = \mathrm{Herm} \sum_{j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{ik} S_{jk;j'k'} u_{j'k'} \tag{53}$$

The minimization yields a Hermitian matrix λ_{ij}, which is obtained as a solution to a linear system, see `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:calculateRegularLambda` for an implementation. A more general form of λ_{ij} is presented in Appendix A below. The problem can now be considered as maximizing a quadratic form with the matrix $\mathcal{S}_{jk;j'k'}$

$$\mathcal{S}_{jk;j'k'} = S_{jk;j'k'} - \lambda_{jj'} \delta_{kk'} \tag{54}$$

$$\sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} u_{jk} \mathcal{S}_{jk;j'k'} u_{j'k'} \underset{u}{\longrightarrow} \max \tag{55}$$

subject to constraints (41). Consider the eigenproblem[6]

$$\frac{\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k,k'=0}^{n-1} u_{jk}\mathcal{S}_{jk;j'k'}u_{j'k'}}{\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k=0}^{n-1} u_{jk}Q_{jj'}u_{j'k}} \underset{u}{\longrightarrow} \max \tag{56}$$

$$\sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1} S_{sq;j'k'}u^{[s]}_{j'k'} - \sum_{j'=0}^{D-1}\lambda_{sj'}u^{[s]}_{j'q} = \mu^{[s]}\sum_{j'=0}^{D-1} Q_{sj'}u^{[s]}_{j'q} \tag{57}$$

with an additional N_d linear constraints added (their specific form, which provides good convergence of the algorithm, is discussed below in Appendix A 1)

$$0 = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1} C_{d;jk}u_{jk} \qquad\qquad d = 0\ldots N_d - 1 \tag{58}$$

A common method of solving eigenproblem (56) with homogeneous linear constraints (58) is the Lagrange multipliers method[39]. This approach, however, creates difficulties when both linear and quadratic constraints are present, especially when the number of constraints is large. A better approach to dealing with linear constraints is to convert (58) to a form that expresses $\mathrm{rank}(C_{d;jk})$ components of u_{jk} in terms of its other components, with the coefficients given by the selected components $\widetilde{C}_{d;jk}$ being zero. We simply moved some of the terms in (58) from the right-hand side to the left-hand side.

$$u_{j^{[d]}k^{[d]}} = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1} \widetilde{C}_{d;jk}u_{jk} \qquad\qquad d = 0\ldots\mathrm{rank}(C_{d;jk}) - 1 \tag{59}$$

Then any u_{jk} satisfying the linear constraints (58) can be expressed as a linear combination of selected components. Let us denote these components as some general variables V_p, $p = 0\ldots N_V - 1$

$$N_V = Dn - \mathrm{rank}(C_{d;jk}) \tag{60}$$

$$u_{jk} = \sum_{p=0}^{N_V-1} M_{jk;p}V_p \tag{61}$$

---

[6] From the variation (51), it follows that the most interesting $u^{[s]}_{jk}$ states have a value of the functional (56) close to zero. Therefore, the matrix $Q_{jj'}$ in the denominator can be chosen as any positively definite Hermitian matrix to improve convergence. Choosing $Q_{jj'} = \delta_{jj'}$ gives familiar results; another choice can be $Q_{jj'} = \lambda_{jj'}$, i.e., having the Lagrange multipliers in the denominator instead of adding them to (54): consider a variation of $\sum\limits_{j,j'=0}^{D-1}\sum\limits_{k,k'=0}^{n-1} u_{jk}S_{jk;j'k'}u_{j'k'} \Big/ \sum\limits_{j,j'=0}^{D-1}\sum\limits_{k=0}^{n-1} u_{jk}\lambda_{jj'}u_{j'k}$ over u_{jk} to obtain the same result as (51); see `com/polytechnik/kgo/KGOIterationalLagrangeMultipliersInDenominatorU.java`.

Substitute (61) back to (56) to obtain an unconstrained generalized eigenproblem (63) with respect to N_V generalized variables V_p. The linear constraints (58) are incorporated into new variables V_p, resulting in a reduction of the total number of variables by the rank$(C_{d;jk})$.[7] The transformation in (61) is actually a regular Gaussian elimination, which is a special form of LU decomposition. A simple implementation with row and column pivoting is used in `com/polytechnik/utils/EliminateLinearConstraints_HomegrownLUFactorization.java`. For a very large number of linear constraints, a transformation such as RRQR factorization is probably more numerically stable. The new eigenproblem involves the matrices $\mathcal{S}_{p;p'}$ in the numerator and $Q_{p;p'}$ in the denominator.

$$\mathcal{S}_{p;p'} = \sum_{j,j'=0}^{D-1} \sum_{k,k'=0}^{n-1} M_{jk;p} \mathcal{S}_{jk;j'k'} M_{j'k';p'} \tag{62a}$$

$$Q_{p;p'} = \sum_{j,j'=0}^{D-1} \sum_{k=0}^{n-1} M_{jk;p} Q_{jj'} M_{j'k;p'} \tag{62b}$$

The $\lambda_{p;p'}$ converts in the same way. We can express eigenproblem (56) in the following form:

$$\frac{\sum\limits_{p,p'=0}^{N_V-1} V_p \mathcal{S}_{p;p'} V_{p'}}{\sum\limits_{p,p'=0}^{N_V-1} V_p Q_{p;p'} V_{p'}} \underset{V}{\longrightarrow} \max \tag{63}$$

$$\sum_{p'=0}^{N_V-1} S_{p;p'} V_{p'}^{[s]} - \sum_{p'=0}^{N_V-1} \lambda_{p;p'} V_{p'}^{[s]} = \mu^{[s]} \sum_{p'=0}^{N_V-1} Q_{p;p'} V_{p'}^{[s]} \tag{64}$$

The cost of this conversion is that if we originally set $Q_{jj'} = \delta_{jj'}$, then in the V_p basis, the denominator in (63) is no longer a unit matrix. This is not an issue, as any modern linear algebra package internally converts a generalized eigenproblem to a regular one. See, for example, DSYGST, DSPGST, DPBSTF, and similar subroutines.

Let the original eigenproblem (57) be already solved, and extremal $u_{jk}^{[s]}$ satisfying the partial quadratic constraint (42) are obtained. Consider the Lagrangian variation $\delta\mathcal{L}/\delta u_{sq}$. If

[7] Note that at this stage, some of the vectors that were removed by the constraints can possibly be reintroduced into the V_p basis. They can be directly added as additional column(s) to $M_{jk;p}$. This changes only the size of the V_p basis to $N_V = Dn - \text{rank}(C_{d;jk}) + N_{inj}$. A better option, however, is to modify $C_{d;jk}$ initially and avoid injecting vectors into the basis.

the state u_{jk} is extremal in (56), then the variation (51) is zero

$$0 = \sum_{j'=0}^{D-1} \sum_{k'=0}^{n-1} S_{sq;j'k'} u_{j'k'} - \sum_{j'=0}^{D-1} \breve{\lambda}_{sj'} u_{j'q} \tag{65}$$

$$\breve{\lambda}_{jj'} = \lambda_{jj'} + \mu^{[s]} Q_{jj'} \tag{66}$$

It is *exactly zero* (for all Dn equations) if u_{jk} is one of the $u^{[s]}_{jk}$ and the Lagrange multipliers are as in (66). However, this u_{jk} may not satisfy the full set of constraints (41). It is adjusted with (46) to satisfy all the constraints, and a new $\lambda_{jj'}$ is calculated to use in the problem (57). A difficulty we encountered in [29, 30] is that this iterative algorithm, when $\lambda_{jj'}$ (53) is used, does not converge to a solution. In this paper, this difficulty is overcome by solving the eigenproblem (57) with additional linear constraints (58). The improved iterative algorithm becomes the following:

1. Take initial λ_{ij} and linear constraints $C_{d;jk}$ to solve the optimization problem (63) with respect to V_p. The solution method involves solving an eigenvalue problem of dimension N_V, which corresponds to the number of columns in the $M_{jk;p}$ matrix. A new u_{jk} is obtained from V_p using (61). The result: $s = 0 \ldots N_V - 1$ eigenvalues $\mu^{[s]}$ and corresponding matrices $u^{[s]}_{jk}$ reconstructed from $V^{[s]}_p$. The value of N_V is typically $Dn - (D-1)(D+2)/2$.

2. A heuristic is required to select the u_{jk} among all N_V eigenstates. Trying a number of them and selecting the maximum (i.e. from all $s = 0 \ldots N_V - 1$ select the best state among the top eigenvalues $\mu^{[s]}$: try all positive and 10 highest negative ones) providing a large value of the original functional (47) typically results in only a local maximum. Numerical experiments show that the index of the eigenstate is a good invariant: *always* selecting the state $V^{[s]}_p$ with the largest $\mu^{[s]}$, second largest $\mu^{[s]}$, third largest $\mu^{[s]}$, etc., converges to a different solution of the original problem. Starting with about the fifth largest eigenvalue, convergence may not always be observed. The global maximum of $\mathcal{F}$ typically corresponds to selecting the largest $\mu^{[s]}$. A good heuristic is to run the algorithm $\mathfrak{n}$ times, always selecting the state of the $\mathfrak{n}$-th largest $\mu^{[s]}$. Then select the global maximum among these $\mathfrak{n}$ runs; the remaining $\mathfrak{n} - 1$ solutions are also good — this way we managed to obtain up to a dozen different solutions. At worst — a cycle without convergence (usually with a period of 2 iterations) is observed; this was an

issue in [29]. With the linear constraints technique of Appendix A 1 this difficulty is overcome.

3. The obtained u_{jk} is not partially unitary since constraint (42) is a subset of the full constraints (41). Apply the adjustment (46) and calculate the λ_{ij} (53) in the adjusted state; these are the Lagrange multipliers for the next iteration.

4. For a good convergence, in addition to λ_{ij}, we need to select a subspace for the next iteration's variation of u_{jk}. Using the full size Dn basis leads to poor convergence [29]. There are two feasible options to improve it: either use an advanced method for calculating λ_{ij}, as detailed in Appendix A below, or constrain the subspace of u_{jk} variation, as discussed in Appendix A 1 below. The latter technique of additional linear constraints $C_{d;jk}$ (58) provides superior results.

5. Insert this new λ_{ij} into (54) and, using the basis V_p obtained (61) from $C_{d;jk}$, calculate the matrices for the numerator and denominator of the generalized eigenproblem (62) to be used in the next iteration. Repeat the iteration process until converging to a maximum (presumably global) of $\mathcal{F}$ with u_{jk} satisfying the constraints (41). If convergence is achieved, the λ_{ij} stops changing from iteration to iteration, and the $\mu^{[s]}$ of the selected state in step 2 above is close to zero. On the first iteration, take initial values of Lagrange multipliers $\lambda_{ij} = 0$ and have no linear constraints.

If, in the eigenvalue selection step 2, the state of the maximal eigenvalue is unconditionally selected, the algorithm becomes linear: it contains no conditional expressions, no selectively executed instructions, no "branching". In this case, it possesses a very simple flat logic that converges to the global maximum of the objective function. This represents a repeated process of: Finding the state corresponding to the maximal eigenvalue of an eigenproblem. Adjusting the found state to satisfy the full set of constraints. Finding from it the Lagrange multipliers λ_{ij} and the constraints $C_{d;jk}$ (a subspace for the next iteration's variation). Such a flat structure greatly simplifies the algorithm's complexity analysis, computer implementation, and offers great potential for parallelization.

Based on a number of numerical experiments, we can conclude that this iterative algorithm almost always converges. Determining the exact convergence domain is a subject of future research. A distinctive feature of this algorithm is that instead of the usual iteration internal

state in the form of a pair (approximation, Lagrange multipliers) (u_{jk}, λ_{ij}), it uses an iteration internal state in the form of a triple (approximation, Lagrange multipliers, homogeneous linear constraints) $(u_{jk}, \lambda_{ij}, C_{d;jk})$. Whereas most optimization algorithms use linear system solutions (Newtonian iteration) as a building block, the algorithm in question employs eigenproblem solutions as the building block. This allows us to develop a much more fine-grained solution selection in step 2 above, which makes the algorithm less sensitive to degeneracy and more likely to converge to the global maximum. The drawback of using an eigenproblem solution as a building block is that it is more computationally costly than a linear system solution or first-order gradient methods. However, the main goal of the paper is to present a working proof-of-concept algorithm capable of solving a new algebraic "eigenoperator" problem (39); the computational complexity of optimization is a separate concern. An apparent optimization would be to replace a general-purpose eigenproblem solver with one that finds only the largest eigenvalue in step 2, which is typically adequate and expected to significantly enhance algorithm performance, see Appendix B for a preliminary analysis.

A reference implementation of this algorithm is available at `com/polytechnik/kgo/KGOIterationalSubspaceLinearConstraints.java`. There are dozens of other algorithms implemented, but only this one provides iterative convergence.[8] There is a driver in the `com/polytechnik/kgo/KGOSolutionVectorXVectorF.java:main` class for a quick test of the algorithm. The driver generates a deterministic random sample from which the tensor $S_{jk;j'k'}$ is calculated, and then the specified algorithm (taken from the command argument) is applied to find the optimal partially unitary operator. Let us compare the convergence of this paper's iterative algorithm with the results from [29] (that uses vanilla Lagrange multipliers) on the driver's deterministically randomly generated data with default settings: $D = 4$ and $n = 19$ of $M = 13540$ observations.

```
java com/polytechnik/kgo/KGOSolutionVectorXVectorF ITERATIONS_KGOIterationa
lSubspaceLinearConstraints 2>&1 | grep Selected
```

and

```
java com/polytechnik/kgo/KGOSolutionVectorXVectorF ITERATIONS_KGOIterationa
```

[8] An equivalent implementation with additional features[40], `com/polytechnik/kgo/KGOIterationalSubspaceLinearConstraintsB.java`, follows the same logic, but all calculations are performed in the original basis without using $v_{jk}^{[s]}$. The result is identical to `KGOIterationalSubspaceLinearConstraints.java`, which once again demonstrates the basis-invariance of our theory.

TABLE I. Comparison of convergence between the algorithm presented in this paper and the one from our previous work [29], conducted on a data sample with $D = 4$, $n = 19$, $M = 13540$ for the first $i = 0 \dots 17$ iterations. We present: μ – the eigenvalue of the selected state, $\mathcal{F}$ – the value of the functional (47) normalized to the number of observations, and an indicator of unitarity $\sum 1/\lambda_G$.

	KGOIterationalSubspaceLinearConstraints			KGOIterationalSimpleOptimizationU		
i	μ	$\mathcal{F}$	$\sum 1/\lambda_G$	μ	$\mathcal{F}$	$\sum 1/\lambda_G$
0	2296.97	7864.08	38.86	2296.97	7864.08	38.86
1	201.08	7936.32	20.02	239.84	8020.58	34.94
2	113.98	8010.69	27.04	110.62	8032.89	17.70
3	100.82	8178.92	7.07	53.94	8014.30	14.31
4	64.91	7890.37	26.06	150.60	8138.49	15.82
5	204.71	8078.93	32.47	57.33	8113.74	11.39
6	134.24	7873.87	34.30	109.05	8071.24	17.90
7	201.39	8112.94	24.76	38.24	8005.98	11.93
8	90.78	7869.54	34.38	92.91	8059.80	18.31
9	159.77	8057.35	12.58	49.79	8101.24	6.51
10	83.15	8212.67	8.06	148.46	8094.14	29.02
11	28.15	8194.82	6.22	86.11	8035.41	16.24
12	25.06	8292.05	4.03	156.60	8060.77	26.22
13	3.08	8302.52	4.02	95.37	8112.07	17.72
14	0.23	8303.46	4.00	73.73	8016.44	23.42
15	$5.63 \cdot 10^{-4}$	8303.46	4.00	202.11	8040.93	32.05
16	$1.05 \cdot 10^{-8}$	8303.46	4.00	93.91	7963.84	21.20
17	$2.39 \cdot 10^{-13}$	8303.46	4.00	192.02	8030.50	24.14

`lSimpleOptimizationU 2>&1 | grep Selected`

The output is `grep`-filtered to show only the iteration status. The results are presented in Table I. From the table, we see that the algorithm cycles through a number of initial iterations without convergence. However, once it enters a contraction mapping area, convergence becomes very fast (faster than any geometric progression). The plain vanilla Lagrange multipliers

method does not converge at all: optimizing (56) over the full Dn space severely violates constraints (41), and problem degeneracy impedes convergence. This can be demonstrated by running `KGOIterationalSubspaceLinearConstraints` in the standard manner for 20 iterations to obtain a perfect solution. Then, starting from iteration 21, turn off the linear constraints (A8), effectively switching to the vanilla Lagrange multipliers method internally. The result is that, for the next 4 to 7 iterations, the solution remains almost the same. However, after 7-10 iterations, it starts to diverge due to accumulated floating-point errors, leading to irregular cycling without convergence. This demonstrates the critical importance of linear constraints (A8) for the convergence of iterative algorithms and the intrinsic instability of vanilla methods.

There are several QCQP software packages available, both commercial and open-source, such as [41], python qcqp, NAG QCQP among many others. A question arises: How well can existing software handle a QCQP optimization problem like that in this paper? We did not extensively test third-party software (although we plan to do so in the future), but generally, the results were unsatisfactory, especially in terms of finding the global maximum. The reason is that existing QCQP software packages were designed as general-purpose solvers intended to solve any QCQP problem. Many of them are heuristic Newton-style solvers with Lagrange multipliers often combined with some form of linear programming and convex optimization. The optimization problem considered in this paper is special: it has only quadratic equality constraints, is non-convex, and exhibits multiple solutions, local maxima, and multiple saddle points. For such a problem, solvers based on single solution iterations (e.g., Newtonian type (second order), gradient type (first order), etc.) are unlikely to identify the global maximum. One may try Genetic programming solvers, but they lack knowledge about the underlying algebraic structure of the problem. The problem of finding an operator that optimally converts an operator from one Hilbert space to another requires a specialized solver. Our use of the generalized eigenvalue problem as the algorithm's building block has the advantage of obtaining many solution candidates (eigenvectors) at once. With proper selection, we can greatly increase the chances of finding the global maximum. Compared to [29], the quality of our optimization algorithm has been greatly improved, and we intend to use it as a "black box" for applying it to several problems for demonstration purposes. Let us demonstrate the value of finding a partially unitary operator that optimally transforms a vector from Hilbert space IN to Hilbert space OUT.

IV. A DEMONSTRATION OF RECOVERING UNITARY DYNAMICS FROM PHASE-STRIPPED DATA

Consider an inverse problem. Let a dynamic system evolve with the equation $\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)}$ (31). The problem is to recover the orthogonal operator $\mathcal{U}$ from an observed sample $\mathbf{X}^{(l)}$. The problem would be trivial if $\mathbf{X}^{(l)}$ were directly observed — a regression analysis could reveal $\mathcal{U}$, see e.g. the section "An application of LRR representation solution to dynamic system identification problem" of [30]. What greatly complicates matters is that we consider observations of a quantum channel type; hence, $\mathbf{X}^{(l)}$ is observable only up to a phase. This means that any wavefunction mapping (2) can only be observed with an unknown phase. This can be modeled by multiplying the actually measured classical system values by random phase factors. The problem is to determine the operator $\mathcal{U}$ from the sample given in (32).[9]

In this section a few simple examples are presented. All calculations are performed in real space, using a ± 1 factor instead of a complex phase factor. A timeserie is initially generated with equation (31). The obtained vectors $\mathbf{X}^{(l)}$ are then multiplied by random ± 1 factors. These new vectors (32) are now considered as observables. The problem is to recover $\mathcal{U}$ from observable data. We present a recovery with two quantum channels: transforming the Gram matrix of Section II C and transforming the unit matrix of Section II D. Since the unitarity of the test data is exact, both methods recover the underlying operator $\mathcal{U}$ exactly. Therefore, only the Gram matrix quantum channel is presented below.

Let us start with a simple SO(3) rotation group. Rotation matrices can be represented using Euler angles (φ, θ, ψ) (in a space of dimension d, there are a total of $d(d-1)/2$ angles).

$$g(\varphi,\theta,\psi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \times \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{67}$$

[9] The operator itself is defined within a phase. Further degeneracy may arise from the data; for instance, if some components X_j and X_{j+1} consistently remain the same in the sample, then we cannot distinguish between a unit matrix and a permutation matrix. The input data sample should be information-complete (IC) [22].

Take $\mathcal{U} = g(\varphi = 0.1, \theta = 0.4, \psi = 0.7)$

$$\mathcal{U} = \begin{pmatrix} 0.7018 & -0.7113 & 0.0389 \\ 0.6668 & 0.6366 & -0.3875 \\ 0.2509 & 0.2978 & 0.9211 \end{pmatrix} \tag{68}$$

and apply it to transform the initial vector $(0.0921, 0.5523, 0.8285)$ through 1000 transformations. The datafile `SO3.csv` is generated using

```
java com/polytechnik/algorithms/PrintOrthogonalSeq
Writing 1000 points to /tmp/SO3.csv .
```

There is a simple demo program that recovers operator $\mathcal{U}$ from sampled data (32) with random phases ± 1 possibly introduced into observations. This program maximizes (33) with a crude regularization (36) (numerical stability depends on regularization, but the optimization result does not [30]) and does not address possible data degeneracy. However, its simplicity makes its internals clear.

```
java com/polytechnik/algorithms/DemoRecoverUnitaryMatrixFromSeq --data_file
_to_build_unitarymodel_from=/tmp/SO3.csv --data_cols=6:1,3:0
```

The program recovers the $\mathcal{U}$ identically

$$\mathcal{U} = \begin{pmatrix} -0.7018 & 0.7113 & -0.0389 \\ -0.6668 & -0.6366 & 0.3875 \\ -0.2509 & -0.2978 & -0.9211 \end{pmatrix} \tag{69}$$

As discussed earlier, the operator $\mathcal{U}$ is defined within a ± 1 factor in real space. The same test was run on randomly generated orthogonal matrices of dimensions $3, 5, 7, 17, 40$. All tests can be run automatically using

```
java com/polytechnik/algorithms/PrintOrthogonalSeq\$TestAuto
```

The maximal difference in matrix elements (for all dimensions tried) is less that 10^{-13}, which is about floating point errors. This leads us to conclude that the developed numerical algorithm in Section III can recover system dynamics from wavefunction measurements (without a phase) for dynamic systems of high dimension. This algorithm is a powerful method for solving the inverse problem in quantum mechanics.

V. A DEMONSTRATION OF POLYNOMIAL MAPPING RECOVERY

Consider polynomials in the $[-1:1]$ interval. Let there be $l = 1 \dots M$ points $y^{(l)}$ split equidistantly in the interval. Define the data

$$x_k^{(l)} = \xi^{(l)} T_k(y^{(l)}) \qquad k = 0 \dots n-1 \tag{70a}$$

$$f_j^{(l)} = \zeta^{(l)} P_j(y^{(l)}) \qquad j = 0 \dots D-1 \tag{70b}$$

Here, T_k is a Chebyshev polynomial and P_j is a Legendre polynomial. The $\xi^{(l)}$ and $\zeta^{(l)}$ are deterministic random functions of l that take the value ± 1. The problem is to build the u_{jk} matrix that maximizes (5) subject to the constraints in (6). The mapping (4) for the (70) data has the meaning of finding the coefficients that expand D Legendre polynomials in n Chebyshev polynomials. The problem would be trivial (reduced to solving a linear system) were it not for the random ± 1 factors $\xi^{(l)}$ and $\zeta^{(l)}$ in (70). With the presence of random phases, any direct projection of one basis onto another becomes unavailable. The only feasible way is to consider the quantum channel mapping (4). The solution is similar to that of the previous section. Calculate the Gram matrices (19), (20) and the $S_{jk;j'k'}$ tensor (34). After any regularization of the input data, for example (36), apply the algorithm from Section III. Specifically, generate a data file named `ChebyshevLegendre.csv` by running

```
java com/polytechnik/algorithms/PrintChebyshevToLegendreMapping
Writing 500 points to /tmp/ChebyshevLegendre.csv
```

This file contains 500 points in the interval $[-1:1]$ of (70) data, with $T_0 \dots T_{10}$ and $P_0 \dots P_5$ randomly multiplied by ± 1 factors. Consider the simple task of converting $T_0 \dots T_4$ to $P_0 \dots P_4$. Use a simple demonstration program that recovers the mapping (4) from sampled data (70), with random phases ± 1 possibly introduced into observations (the approach is phase-agnostic). This program maximizes $\mathcal{F}$, and its simplicity makes its internals clear.

```
java com/polytechnik/algorithms/DemoRecoverMapping --data_file_to_build_mod
el_from=/tmp/ChebyshevLegendre.csv --data_cols=21:2,6:14,18:-1:0
```

The program recovers the mapping identically (values below 10^{-15} are replaced by zero).

$$\mathcal{U} = \begin{pmatrix} -1.0 & 0 & 0 & 0 & 0 \\ 0 & -1.0 & 0 & 0 & 0 \\ -0.25 & 0 & -0.75 & 0 & 0 \\ 0 & -0.375 & 0 & -0.625 & 0 \\ -0.140625 & 0 & -0.3125 & 0 & -0.546875 \end{pmatrix} \tag{71}$$

Exact values can be obtained using a polynomial library from [42], which is included with the software for this paper[43]. Run in jshell `new Legendre().convertBasisToPBASIS(5,new Chebyshev())` to obtain P_j over T_k expansion

$$\begin{pmatrix} 1.0 & & & & \\ 0 & 1.0 & & & \\ 0.25 & 0 & 0.75 & & \\ 0 & 0.375 & 0 & 0.625 & \\ 0.140625 & 0 & 0.3125 & 0 & 0.546875 \end{pmatrix} \tag{72}$$

i.e. $P_3 = 0.375T_1 + 0.625T_3$. This matches (71) within a $\pm$ sign. In this $n = D$ case we have a perfect recovery. Now consider the same problem of polynomial mapping for $D < n$, let us run it with $D = 4$ and $n = 5$.

```
java com/polytechnik/algorithms/DemoRecoverMapping --data_file_to_build_mod
el_from=/tmp/ChebyshevLegendre.csv --data_cols=21:2,6:14,17:-1:0
```

The obtained mapping

$$\mathcal{U} = \begin{pmatrix} 0 & -1.29434 & 0 & 0.56023 & 0 \\ -0.63950 & 0 & -0.44948 & 0 & 0.07596 \\ 0 & -0.67678 & 0 & -0.69274 & 0 \\ -0.14967 & 0 & -0.62083 & 0 & -0.50233 \end{pmatrix} \tag{73}$$

is not a subset of the $D = 5$, $n = 5$ mapping (71), a subset can be obtained by running with $D = 4$, $n = 4$. We previously discussed [29] the difficulties of the $D < n$ case. A mapping between two Hilbert spaces of different dimensions sometimes leads to unusual behavior, as

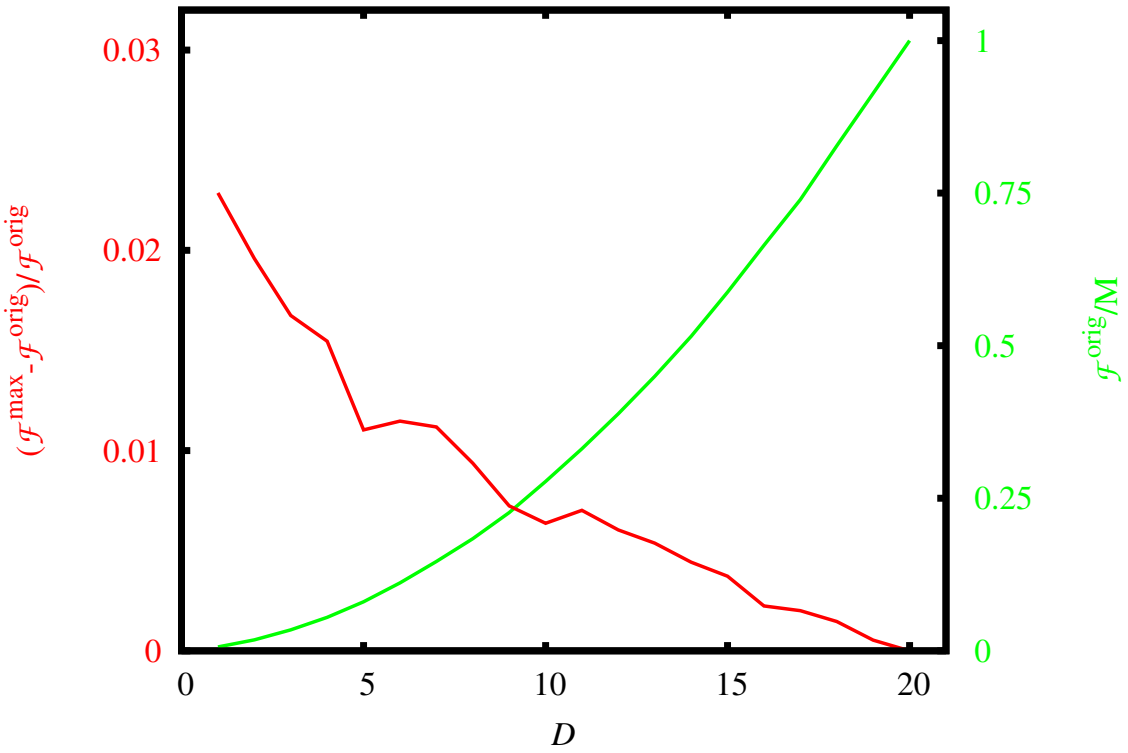

FIG. 1. For $D \leq n = 20$, a sample $\psi_l \to \phi_l$ is constructed with a known u_{jk}, the "original" one. The result is compared with the fidelity-maximizing solution of optimization problem (5). For $D < n$ (partial unitarity), the fidelity of the optimization problem solution is always greater than the original fidelity. For $D = n$ (unitarity), the original and optimization problem solutions match exactly. A deviation of $\mathcal{F}^{orig}/M$ from 1 indicates that the transform does not preserve trace — it is a trace preserving map only for $D = n$.

the mapping is no longer trace-preserving. The value of $\mathcal{F}$ increases when going from $D = 4$, $n = 4$ to $D = 4$, $n = 5$ but the mapping is no longer a subset. Consider a simple numerical experiment demonstrating the behavior of trace-decreasing maps.

VI. A DEMONSTRATION OF PARTIALLY UNITARY BEHAVIOR $D < n$

Consider a numerical experiment with a partially unitary mapping where $D < n$. For demonstration we select a unitary matrix with matrix elements U_{jk} $j, k = 0 \ldots n-1$ and generate a sample of random ψ_l (each of dimension n) with $M = 1000$ observations. Then, for a given D, we select the first D rows of U_{jk} and use them as the partially unitary operator

u_{jk}, $j = 0 \dots D-1$, $k = 0 \dots n-1$. Applying it to every ψ_l, a corresponding ϕ_l (of dimension D) is created, and the mapping (2) is obtained for all sample elements. Then $S_{jk;j'k'}$ is created from the sample and the numerical algorithm of Section III is applied. For $D = n$, this is essentially the same problem considered in Section IV above. Now we run the problem for all $D = 1 \dots n$. For every D, we calculate the original u_{jk} fidelity and the fidelity of the obtained u_{jk} as the solution to the optimization problem, run `java com/polytechnik/algorithms/DemoDMUnitaryMappingTest 2>&1 | grep Map:`. In the $D = n$ case both fidelities are identical as well as the operators u_{jk}. For $D < n$, however, the fidelity $\mathcal{F}^{\max}$ calculated with the fidelity-maximizing operator is greater than the fidelity $\mathcal{F}^{orig}$ of the original operator used to construct the sample (2). In Fig. 1 we present the result for $n = 20$ — the original operator fidelity (normalized to sample size $M = 1000$) and its relative difference $(\mathcal{F}^{\max} - \mathcal{F}^{orig})/\mathcal{F}^{orig}$. We determined that for $D < n$, $\mathcal{F}^{\max}$ is always greater than $\mathcal{F}^{orig}$. The behavior is determined by the form of fidelity. For fidelity definition (5), the optimization problem solution in the case of partial unitarity $D < n$ produces greater fidelity than the fidelity of the mapping used to construct the sample. Consider the case where $D = 1$. From the constraints (8) it follows that $\sum_{k=0}^{n-1} u_{0k}^2 = 1$ and the wavefunction transform is $f_0 = \sum_{k=0}^{n-1} u_{0k} x_k$. The vector $\mathbf{f}$ (now a vector of a single element) has a unit norm only if $x_k = u_{0k}$ (for $D = 1$, this is actually a projection). For $D < n$, the quantum channel (38) is a trace decreasing map, and the algorithm finds a mapping with higher fidelity than the original mapping used to construct the quantum channel in the first place.

VII. A DEMONSTRATION OF FUNCTION INTERPOLATION

In the two previous sections, we considered examples of constructing u_{jk} when the data of $\mathbf{x} \to \mathbf{f}$ form has vectors $\mathbf{x}$ and $\mathbf{f}$ already belonging to corresponding Hilbert spaces. For classical measurements, this is often not the case; a transformation $\mathbf{x} \to \psi$, $\mathbf{f} \to \phi$ is required to build Hilbert space states ψ and ϕ from the original observations $\mathbf{x}$ and $\mathbf{f}$. The most straightforward method to construct such states is to consider states localized at given $\mathbf{x}$ (or $\mathbf{f}$) as wavefunctions (23) and use them for the mapping (25). As considered in section II B above, localized states $\psi_{\mathbf{y}}(\mathbf{x})$ use the same Gram matrix (19) for the scalar product, which is obtained from $l = 1 \dots M$ sampled data. Effectively this is sample average (18) with weight multiplied by a localized at $\mathbf{y}$ non-negative function $\psi_{\mathbf{y}}^2(\mathbf{x})$ (it is normalized as $1 = \langle \psi_{\mathbf{y}}^2(\mathbf{x}) \rangle$).

It provides a Radon-Nikodym approximation of some characteristic g

$$g_{RN}(\mathbf{y}) = \left\langle g\psi_{\mathbf{y}}^2 \right\rangle \tag{74}$$

The approximated value here is a superposition of the observed g with the positive density $\psi_{\mathbf{y}}^2(\mathbf{x})$. The familiar least squares interpolation (22), which expands the value rather than representing a probability, has a similar form.

$$g_{LS}(\mathbf{y}) = \psi_{\mathbf{y}}(\mathbf{y}) \left\langle g\psi_{\mathbf{y}} \right\rangle \tag{75}$$

Here, the average is taken with the density $\psi_{\mathbf{y}}(\mathbf{x})$ — it is not always positive as $\psi_{\mathbf{y}}^2(\mathbf{x})$ in the Radon-Nikodym (74) expression; the $\psi_{\mathbf{y}}(\mathbf{y})$ is a normalizing factor of $1/\sqrt{K(\mathbf{y})}$. The expressions 74) and (75) differ in how they represent the delta function $\delta(\mathbf{x}-\mathbf{y})$: as $\psi_{\mathbf{y}}^2(\mathbf{x})$ or as $\psi_{\mathbf{y}}(\mathbf{y})\psi_{\mathbf{y}}(\mathbf{x})$.

Consider the scalar function interpolation problem in the form of mapping between two Hilbert spaces with u_{jk}. Let there be a scalar $f(x)$ and M observation points $f^{(l)} = f(x^{(l)})$, $l = 1 \dots M$. Convert this scalar mapping to a vector one (the $\xi^{(l)}$ and $\zeta^{(l)}$ are deterministic random functions on l that take the value ± 1)

$$x_k^{(l)} = \xi^{(l)} \left(x^{(l)}\right)^k \qquad k = 0 \dots n-1 \tag{76a}$$

$$f_j^{(l)} = \zeta^{(l)} \left(f^{(l)}\right)^j \qquad j = 0 \dots D-1 \tag{76b}$$

For numerical stability it is better, instead of monomials (powers of the argument), to use Chebyshev or Legendre polynomials as $x_k^{(l)} = \xi^{(l)} T_k(ax^{(l)} + b)$ and $f_j^{(l)} = \zeta^{(l)} T_j(cf^{(l)} + d)$ with a, b, c, d chosen to bring the argument into $[-1 : 1]$ interval. This greatly increases the numerical stability of calculations. However, the result itself is invariant with respect to the polynomial basis choice — the result will be the same with any polynomial basis. From the obtained vector to vector mapping construct $\mathbf{x}^{(l)}$ and $\mathbf{f}^{(l)}$ localized states $\psi_{\mathbf{x}^{(l)}}(\mathbf{x}) \to \psi_{\mathbf{f}^{(l)}}(\mathbf{f})$ mapping (2) with the former one defined in (23) and the latter one obtained from it with argument and index replacement (25). Put them into (28) and obtain (27). After solving the problem for u_{jk} an evaluation of interpolated $f(x)$ can be performed as follows: From a given x construct the vector $\mathbf{x}$ (76a). Substitute it to (29) to obtain the probability of a given vector of outcome $\mathbf{f}$. In the scalar case the outcome value can be evaluated, for example, by constructing (76b) vector $\mathbf{f}$ from f and then finding the value of f that provides the maximal value of probability (29). For scalar f this problem can be reduced to finding the roots of a single variable polynomial.

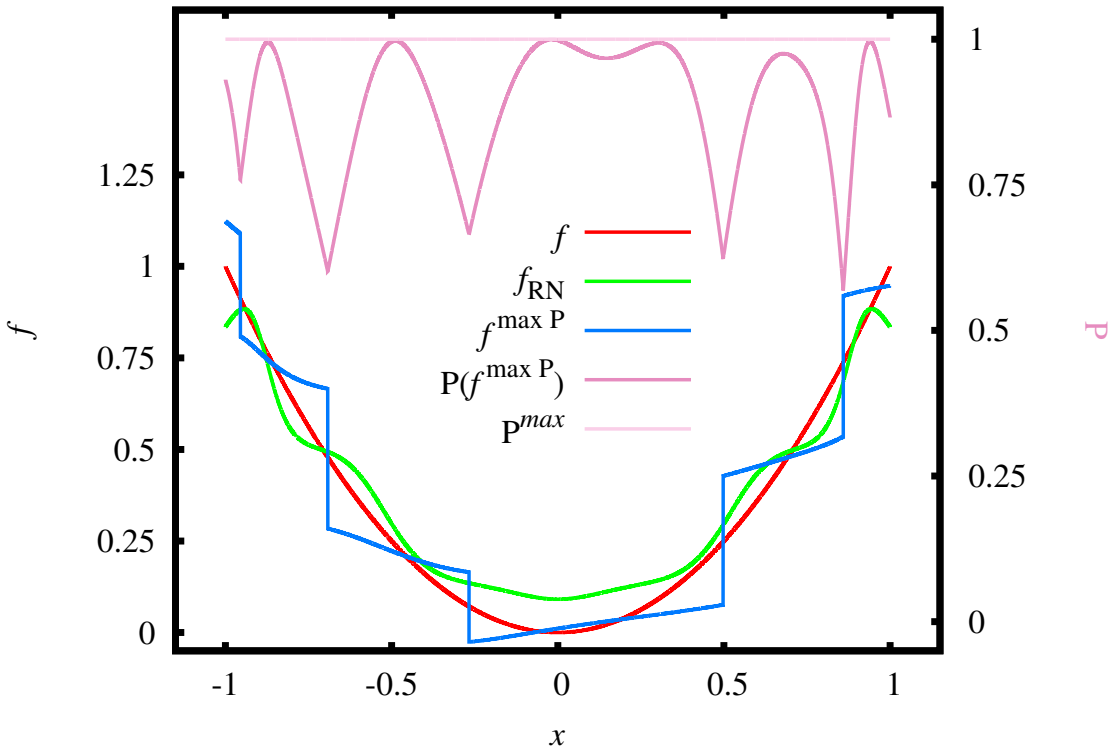

FIG. 2. Scalar function $f = x^2$ (red) interpolation with: green: Radon-Nikodym (74), blue: f corresponding to the maximal probability (29), the dependence has discontinuities; some numerical instability presents even in this $D = n = 6$ case. Pink: probabilities corresponding to (29) and (77a).

This approach creates a number of issues. An attempt to fit a scalar function $f(x)$ to Hilbert spaces mapping using moments like $\left\langle f^j x^k \,\middle|\, K(\mathbf{x})K(\mathbf{f}) \,\middle|\, f^{j'} x^{k'} \right\rangle$ (26) create difficulties both in computations and in mapping back from Hilbert space to function value. As a simple demonstration, let $f(x) = x^p$ with $p = 2$. This requires calculating x-moments of maximal degree $2(n-1) + 2p(D-1)$, which for $n = D = 6$ gives a maximal polynomial degree of 30, creating numerical instability difficulties. There is an additional problem of converting back from the Hilbert space to the value of f by checking all extrema of (29) to find the f that provides the maximal probability is a "switching" function that creates a non-continuous solution when polynomial roots change. An example is presented in Fig. 2. For $f = x^2$ (red line), we calculated the Radon-Nikodym approximation (74) (least squares is identical to the original f since x^2 is in the basis, thus it is not presented in the plot), as well as f corresponding to the maximum probability (blue line) and its corresponding probability (pink). In the plot, one can see starting numerical instability (asymmetry in the plot) and

discontinuity in f corresponding to the change of the selected root. As expected for the x where $P(f^{\max P}(x)) = 1$ the value of $f^{\max P}$ is equal to the exact $f(x)$.

Alternatively[29], we can consider (29) without the requirement (76b) that all components of $\mathbf{f}$ are obtained from a single scalar f.

$$f_j^{\max P} = \sum_{j'=0}^{D-1} G^{\mathbf{f}}_{jj'} a_{j'} \tag{77a}$$

$$P(\mathbf{f}^{\max P})\Big|_{\mathbf{x}} = \sum_{j,j'=0}^{D-1} a_j G^{\mathbf{f}}_{jj'} a_{j'} \tag{77b}$$

The maximal value of the probability (29) at (77b) is an important characteristic of the obtained solution. If the problem dimensions are balanced – the value is equal to 1 for all x (pink line P^{max} in Fig. 2), it can be lower in the case where $D < n$, which requires separate consideration. In [29] we evaluated $f(x)$ from the first two elements of the vector $\mathbf{f}$: for a polynomial basis $Q_j(f)$ (in (76b) $Q_j(f) = f^j$) we can obtain the value of f from the ratio $Q_0 = f_0^{\max P}$ and $Q_1(f) = f_1^{\max P}$. Back in [29], we believed that observed singularities were caused by a poor solution to the optimization problem. While the algorithm developed in this paper provides a very good solution to the optimization problem of Hilbert spaces mapping, the method of obtaining the value of the scalar f from the ratio of the first two basis functions does not work well.

Currently we do not have a good solution to the problem of converting two Hilbert spaces mapping u_{jk} into a scalar function $f(x)$. This is a subject of future research.

VIII. CONCLUSION

The problem of mapping between Hilbert spaces, from IN of $|\psi\rangle$ to OUT of $|\phi\rangle$, based on a set of wavefunction measurement (within a phase) observations $\psi_l \to \phi_l$, $l = 1 \dots M$, is formulated as an optimization problem maximizing the total fidelity $\sum_{l=1}^{M} \omega^{(l)} \left|\langle \phi_l \left| \mathcal{U} \right| \psi_l \rangle\right|^2$ subject to probability preservation constraints on $\mathcal{U}$. This optimization problem is reduced to a novel QCQP problem of maximizing a quadratic form $\langle \mathcal{U} \left| S \right| \mathcal{U} \rangle \to \max$ subject to partial unitarity constraints on $\mathcal{U}$.

The operator $\mathcal{U}$, represented by a rectangular matrix u_{jk} of $\dim(OUT) \times \dim(IN)$ dimensions with $D = \dim(OUT) \le n = \dim(IN)$, can be viewed as a quantum channel (38). Time evolution represents a special case of this problem. The optimization problem involving

a quadratic function on $\mathcal{U}$ with quadratic constraints is solved using a numerical method outlined in Section III. The method is an iterative approach with a generalized eigenproblem as its building block. This differs from commonly used optimization methods that employ Newtonian iteration or gradient iteration as their building blocks. This approach allows us to find the global maximum for almost any input. In addition, instead of the usual iteration internal state in the form of a pair (approximation, Lagrange multipliers) (u_{jk}, λ_{ij}), the algorithm uses an iteration internal state in the form of a triple: approximation, Lagrange multipliers, and homogeneous linear constraints $(u_{jk}, \lambda_{ij}, C_{d;jk})$. It is these linear constraints that contribute to the convergence of the algorithm.

An important feature of the algorithm is that, on each iteration, there is no single solution candidate. While Newtonian or gradient-based algorithms have a single iterative candidate, our eigenproblem-based method provides multiple solutions simultaneously (eigenvectors), the λ_{ij} and constraints $C_{d;jk}$ are then calculated from the selected eigenvector. Therefore, instead of a single solution, a group of solution candidates (eigenvectors) is obtained at every iteration. This is similar to genetic programming optimization, where multiple solutions "flow around" local maxima and saddle points. It is the multiple solutions that allow us to identify the global maximum.

This numerical method can be applied to various classical and quantum problems, such as variational quantum algorithms and quantum mechanical inverse problems, like recovering a Hamiltonian from a sequence of wavefunction observations. An operator $\mathcal{U}$ is obtained from these observations, after which equation (16) can be applied to derive the Hamiltonian.

A demonstration of unitary dynamics $\mathbf{X}^{(l+1)} = \mathcal{U}\mathbf{X}^{(l)}$ for system identification involves determining $\mathcal{U}$ from measured $\widetilde{\mathbf{X}}^{(l)}$, which are actual $\mathbf{X}^{(l)}$ multiplied by random phase factors. This was presented for a number of orthogonal operators $\mathcal{U}$ with dimensions $n = D = \{3, 5, 7, 17, 40\}$. An exact recovery of $\mathcal{U}$ was observed. The technique was also applied to problems involving polynomial bases mapping and scalar function interpolation. An exact solution was obtained for polynomial bases mapping. However, solving the problem of scalar function interpolation with the developed technique is challenging due to the lack of good equivalence between the problem of mapping between Hilbert spaces and scalar functions.

The problem of optimal mapping between two Hilbert spaces is reduced to a new algebraic "eigenoperator" problem (39). Currently, we only have a numerical solution for it. An important generalization of Unitary Learning was made in this paper by considering Hilbert spaces of

different dimensions $D \leq n$. A question arises about further generalization. An important topic for future research could be Kraus' theorem, which determines the most general form of mapping between Hilbert spaces[44]:

$$A^{OUT} = \sum_s B_s A^{IN} B_s^\dagger \tag{78}$$

with Kraus operators B_s satisfying the constraints that the unit A^{IN} is converted to the unit A^{OUT}

$$\sum_s B_s B_s^\dagger = \mathbb{1} \tag{79}$$

This is a generalization of the transform (38); it is applicable to systems with quantum decoherence. In Appendix I of [30], the corresponding optimization problem is formulated, but its numerical solution is the subject of future research[40].

ACKNOWLEDGMENTS

This research was supported by Autretech Group, www.атретек.рф, a resident company of the Skolkovo Technopark. We thank our colleagues from the Autretech R&D department who provided insight and expertise that greatly assisted the research. Our grateful thanks are also extended to Mr. Gennady Belov for his methodological support in doing the data analysis.

V.M. is grateful to Professor Arthur McGurn. Working with him at WMU was an important step in my education, where I learned approaches that combine quantum mechanics and numerical computations. This paper is dedicated to Professor Arthur McGurn on the occasion of his 75th birthday.

Appendix A: On Lagrange Multipliers Calculation With Selected States Variation

In this work, we do not iterate for Lagrange multipliers. Instead, at each iteration for the current solution approximation u_{jk}, we calculate Lagrange multipliers corresponding to an extremum in this state. This enhances algorithm stability and allows for directly applying a solution adjustment procedure (46) from the partial constraints (42) to the full set (41).

The variation of the Lagrangian $\mathcal{L}$ (49) must be zero for the state u_{jk}

$$0 = \frac{1}{2}\frac{\delta\mathcal{L}}{\delta u_{iq}} = \sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1} S_{iq;j'k'}u_{j'k'} - \sum_{j'=0}^{D-1}\lambda_{ij'}u_{j'q} \tag{A1}$$

These are Dn equations. The Lagrange multipliers λ_{ij} are a Hermitian $D \times D$ matrix with $D(D+1)/2$ independent elements. Thus, for a general u_{jk}, all Dn equations (A1) cannot be simultaneously satisfied. They are satisfied in (65) for the eigenstate $u^{[s]}_{jk}$ of (57), but after the adjustment (46), this is no longer the case. A trivial solution is to take the L^2 norm of the variation vector (A1) and minimize the sum of squares (52) to obtain (53) as the solution to the linear system. A direct implementation, however, has poor convergence [29]. The idea is to calculate the sum of squares only in specific states. Consider the problem

$$\sum_{s=0}^{N_v-1}\left|\sum_{i=0}^{D-1}\sum_{q=0}^{n-1} v^{[s]}_{iq}\left[\sum_{j'=0}^{D-1}\sum_{k'=0}^{n-1} S_{iq;j'k'}u_{j'k'} - \sum_{j=0}^{D-1}\frac{\lambda_{ij}+\lambda_{ji}}{2}u_{jq}\right]\right|^2 \underset{\lambda_{ij}}{\longrightarrow} \min \tag{A2}$$

The variation (A1) is projected onto $v^{[s]}_{iq}$, $s = 0 \ldots N_v - 1$ states, and the sum of projection squares is taken. If $v^{[s]}_{iq}$ form a full basis, e.g., all the $s = 0 \ldots Dn-1$ eigenvectors $u^{[s]}_{jk}$ of (57), then (A2) is exactly (52). One may think about (A1) as the variation of the Lagrangian (49), $\delta\mathcal{L}/\delta u_{iq}$. The sum of squares (52) is $\left\langle \frac{\delta\mathcal{L}}{\delta u} \middle| \frac{\delta\mathcal{L}}{\delta u} \right\rangle$, the sum of squares in $v^{[s]}_{iq}$-projected states (A2) is $\sum_{s=0}^{N_v-1}\left\langle \frac{\delta\mathcal{L}}{\delta u} \middle| v^{[s]} \right\rangle \left\langle v^{[s]} \middle| \frac{\delta\mathcal{L}}{\delta u} \right\rangle$, for any full basis $v^{[s]}_{iq}$ they are the same. In practice, the $v^{[s]}_{iq}$ may not necessarily be full or orthogonal. If they are a subset of the original eigenstates $u^{[s]}_{jk}$ (57) from the iterative algorithm, they are orthogonal. If the (46) adjustment is applied to them, they are not. One may consider cross states with a $Dn \times Dn$ Gram matrix, similar to (23), as $\sum_{j,j'=0}^{D-1}\sum_{k,k'=0}^{n-1} u_{jk}G^{-1}_{jk;j'k'}v_{j'k'}$. However, this is usually not necessary since the selection of vectors $v^{[s]}_{iq}$ is performed solely to improve the algorithm's convergence.

Once the vectors $v^{[s]}_{iq}$ are selected – in the problem (A2) the $D \times D$ Hermitian matrix λ_{ij} should be expressed via the vector λ_r of dimension $D(D+1)/2$. A variation over λ_r gives a linear system of dimension $D(D+1)/2$ that can be readily solved. The number of $v^{[s]}_{iq}$ vectors should be at least $D(D+1)/2$, otherwise the linear system will be degenerated. For matrix to vector conversion and linear system solution see `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:getLambdaForSubspace`, which implements this functionality to solve the minimization problem (A2) and obtain λ_{ij} from a subspace chosen

as the states of high eigenvalues of problem (57). The success of this approach is moderate: The number of top-$\mu^{[s]}$ states to select is not precisely clear, the linear system may become degenerate, etc. One can check these attempts at `com/polytechnik/kgo/LagrangeMultipliersPartialSubspace.java:getLambdaForSubspace` and their usage in `com/polytechnik/kgo/KGOIterationalLagrangeMultipliersPartialSubspace.java`. This leads us to conclude that instead of considering a subspace for constructing λ_{ij} we should consider a subspace for variation of u_{jk}.

1. Linear Constraints On Variation

Degeneracy of the problem and quadratic constraints require not only a good approximation for Lagrange multipliers, but also a restricted subspace for variation of u_{jk} in the (56) problem. The difficulty arises from partial unitarity constraints (41). The optimization (56) preserves only the partial constraint (42). Consider a full orthogonal basis $v^{[s]}_{jk}$, where $\delta_{ss'} = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1} v^{[s]}_{jk} v^{[s']}_{jk}$, $s = 0 \dots Dn-1$, and a variation vector δu_{jk} expanded over this basis

$$\delta u_{jk} = \sum_{s=0}^{Dn-1} a_s v^{[s]}_{jk} \tag{A3}$$

$$a_s = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1} v^{[s]}_{jk} \delta u_{jk} \tag{A4}$$

If we were working in a regular vector space, the only available operation would be the scalar product

$$\langle v \,|\, u \rangle = \sum_{j=0}^{D-1}\sum_{k=0}^{n-1} v_{jk} u_{jk} \tag{A5}$$

Now, when we study partially unitary operators (38), we involve a tensor

$$G^{v|u}_{ij} = \sum_{k=0}^{n-1} v_{ik} u_{jk} \tag{A6}$$

The scalar product corresponds to $\langle v \,|\, u \rangle = \mathrm{Tr} G^{v|u}$, the Gram matrix (43) is $G^u_{ij} = G^{u|u}_{ij}$. Consider a variation $G^{u+\delta u|u+\delta u}_{ij}$. To preserve the partial unitarity constraints (41) on u_{jk} within linear terms (on δu) we require all off-diagonal elements of $\mathrm{Herm} G^{u|\delta u}_{ij}$ to be zero $0 = G^{u|\delta u}_{ij} + G^{u|\delta u}_{ji}$, $i \neq j$ (homogeneous) and the diagonal elements to be one (inhomogeneous).

The partial constraint (42) preserves the matrix trace; thus, it suffices to have the diagonal elements equal, which forms a homogeneous constraint $0 = G^{u|\delta u}_{ii} - G^{u|\delta u}_{i-1\,i-1}$, $i = 1 \dots D-1$. Expanding δu in the basis (A3), we obtain the constraints

$$0 = \sum_{s=0}^{Dn-1} a_s \left(G^{u|v^{[s]}}_{ij} + G^{u|v^{[s]}}_{ji} \right) \qquad j < i,\ i = 0 \dots D-1 \tag{A7a}$$

$$0 = \sum_{s=0}^{Dn-1} a_s \left(G^{u|v^{[s]}}_{ii} - G^{u|v^{[s]}}_{i-1\,i-1} \right) \qquad i = 1 \dots D-1 \tag{A7b}$$

There are $(D-1)(D+2)/2$ total linear homogeneous constraints on the expansion coefficients a_s. Now, not all Dn coefficients a_s are independent; there are only $Dn - (D-1)(D+2)/2$ independent ones. The constraints (A7) are homogeneous

$$N_d = (D-1)(D+2)/2 \tag{A8a}$$

$$d_{offd} : 0 \dots D(D-1)/2 - 1 \tag{A8b}$$

$$d_{diag} : D(D-1)/2 \dots (D-1)(D+2)/2 - 1 \tag{A8c}$$

$$s : 0 \dots Dn - 1 \tag{A8d}$$

$$C_{d;s} = \begin{cases} G^{u|v^{[s]}}_{ij} + G^{u|v^{[s]}}_{ji} & \text{if } d \in d_{offd} \\ G^{u|v^{[s]}}_{ii} - G^{u|v^{[s]}}_{i-1\,i-1} & \text{if } d \in d_{diag} \end{cases} \tag{A8e}$$

Here $d \in d_{offd}$ corresponds to $j < i, i = 0 \dots D-1$ and $d \in d_{diag}$ corresponds to $i = 1 \dots D-1$. Equations (A8e) can be directly applied to (58) after the basis transformation.

$$C_{d;jk} = \sum_{s=0}^{Dn-1} C_{d;s} v^{[s]}_{jk} \tag{A9}$$

There are $D(D-1)/2$ constraints for zero off-diagonal elements (A8b) and $D-1$ constraints (one less the dimension) of diagonal elements equal to each other (A8c). In total there are $(D-1)(D+2)/2$ homogeneous constraints. The constraints $C_{d;jk}$, like the Lagrange multipliers λ_{ij}, are calculated solely from the current iteration u_{jk}; see `com/polytechnik/kgo/LinearConstraints.java:getOrthogonalOffdiag0DiagEq` for an implementation.

The result of the consideration above is: if, instead of a full basis $v^{[s]}_{jk}$ of dimension Dn, we take the basis V_p (61) that has $\operatorname{rank}(C_{d;jk}) = (D-1)(D+2)/2$ fewer elements — the variation of (63) will preserve partial unitarity of u_{jk} within the first order. This drastically changes the algorithm convergence. It begins to converge perfectly to a true solution only if both sets of

constraint (A7a) and (A7b) are satisfied; a single set alone does not ensure convergence. An iterative algorithm that finds a partially unitary operator optimally converting an operator from the *IN* Hilbert space to the *OUT* Hilbert space (38) is the main result of this paper. The result was achieved by considering, on each iteration, not just a pair of approximation and Lagrange multipliers (u_{jk}, λ_{ij}), but a triple: approximation, Lagrange multipliers, and homogeneous linear constraints: $(u_{jk}, \lambda_{ij}, C_{d;jk})$. This approach addresses the challenges posed by a quadratically constrained degenerate problem that exhibits local extrema and multiple saddle points. A similar situation can be observed in dynamic systems with a singular Lagrangian [45], where the solution can be obtained by considering a "constrained Hamiltonian system" in which the evolution is constrained to a subspace of the phase space. In the current work, the constrained subspace (determined by the coefficients $C_{d;jk}$ (A9)) itself depends on the current iteration u_{jk}.

A problem of optimizing the quadratic form $\sum_{i,j=0}^{D-1} u_i M_{ij} u_j \underset{u}{\longrightarrow} \max$ subject to a single quadratic constraint $1 = \sum_{i,j=0}^{D-1} u_i Q_{ij} u_j$ where Q_{ij} is a positively definite matrix can be reduced to an eigenvalue problem. In Appendix F of [46] and later in [47] a quadratic form optimization problem with *two* quadratic constraints was considered. An additional constraint was in the form $0 = \sum_{i,j=0}^{D-1} u_i C_{ij} u_j$. This problem is much simpler compared to the one addressed in the current paper. For this problem, an algorithm using vanilla Lagrange multipliers iterations converges without requiring additional linear constraints to be added, see `com/polytechnik/utils/IstatesConditionalV2.java` for an implementation. However, even in this simple case, it is necessary to try several starting values for the iterations to find the global maximum. Numerical experiments have shown that adding a single linear constraint on u_j in each iteration greatly increases the likelihood of finding the global maximum. This constraint takes the form $0 = \sum_{i,j=0}^{D-1} u_i C_{ij} u_j^{(cur)}$, where $u_j^{(cur)}$ represents the value of u_j to compute the Lagrange multipliers for the current iteration. A reference implementation `com/polytechnik/utils/IstatesConditionalSubspaceLinearConstraints.java` on each iteration solves an eigenproblem of dimension $D-1$ (since a single constraint reduces the dimension by 1), always selecting the maximal eigenvalue. This approach is akin to the heuristic used in the algorithm described above. The result is almost always better than that of `IstatesConditionalV2.java` which solves an eigenproblem of dimension D and attempts to select the next iteration from a large number of vectors. This demonstrates the advantage of considering the iteration state as a triple (solution, Lagrange multipliers, linear

constraints), even in the simple case of a single additional quadratic constraint.

Appendix B: A Preliminary Analysis of Computational Complexity

Let us estimate the computational complexity of the algorithm. From a sample of M observations (2), the tensor $S_{jk;j'k'}$ is obtained and used to solve the algebraic problem (39). The calculation of $S_{jk;j'k'}$, for not very large samples, contributes little to the overall complexity; that is, the complexity does not significantly depend on the number of observations in the sample. The only requirement for the input sample is that it must be information-complete [22] in order to recover the $\mathcal{U}$.

The tensor has dimensions $Dn \times Dn$. In each iteration, we solve an eigenvalue problem (64). If it were not for the constraints in (58), the dimension of the eigenproblem would be Dn. However, the convergence helper constraints from Appendix A 1 reduce the dimension to $Dn - (D-1)(D+2)/2$. This is the problem that requires the most computations. The full list of problems of substantial computational difficulty is as follows:

- From M observations (2) of vectors with dimensions n and D, construct the tensor $S_{jk;j'k'}$. The calculations are similar to those used, for example, in covariance matrix calculation. Each component of the tensor is a sum over all M observations; this task can be trivially parallelized.

- LU decomposition (61) of the matrix $C_{d;jk}$ (A9) with dimensions $(D-1)(D+2)/2 \times Dn$.

- Taking the square root to obtain the matrix $G^{u;-1/2}$ (45) requires solving an eigenproblem of dimension D. There exist methods to calculate the square root of a positively definite Hermitian matrix without solving an eigenproblem; see for example [48, 49] and the textbook [50]. Since the dimension of this problem, D, is small compared to the dimension of the eigenproblem (64) that we consider next, we conclude that optimizing the square root calculation is not worth the effort.

- Solving the main eigenproblem (64), which has a dimension of $Dn - (D-1)(D+2)/2$. This problem is the most computationally intensive one. The problem can be parallelized [51–53], which can potentially greatly increase the algorithm's performance.

- Calculating new values of Lagrange multipliers: for the problem addressed in this paper, an analytic solution (53) is available. In the general case [40], a linear system with a dimension equal to the number of independent components in the Lagrange multipliers must be solved. For (53), this dimension is $D(D+1)/2$.

The problem (64) is the most computationally difficult. For $D = n$ (unitary learning), the dimension of this eigenproblem is $N = 1 + n(n-1)/2$. In each iteration, we need to solve an eigenproblem of dimension N. Finding all eigenvectors has the same complexity as matrix multiplication and is $O(N^3)$ in practice. However, it can be reduced to $O(N^w)$ for some $2 < w < 3$ [54]. The state selection step, however, typically requires only a single eigenvector corresponding to the maximum eigenvalue. This problem has lower computational complexity, which can be estimated as $O(N^2)$. Thus, the algorithm's complexity can be optimistically estimated at $O(n^4)$ in the unitary learning case.

Appendix C: Software description

- Install java 22 or later.
- Download the latest version of the source code `code_polynomials_quadratures.zip` from [43] or from an alternative location.
- Decompress and recompile the program. Run a simple test to recover orthogonal matrices of dimensions $3, 5, 7, 17, 40$.

  ```
  unzip code_polynomials_quadratures.zip
  javac -g com/polytechnik/*/*java
  java com/polytechnik/algorithms/PrintOrthogonalSeq\$TestAuto >/tmp/diag 2>&1
  ```

  The diagnostics is saved to the file `/tmp/diag`
- Check the maximal absolute difference between the elements of the original and recovered orthogonal matrices, do `grep DIFF /tmp/diag`

```
GRAM DIFF for dim=3 is 2.3314683517128287E-15
UNIT DIFF for dim=3 is 4.440892098500626E-16
GRAM DIFF for dim=5 is 6.439293542825908E-15
```

```
UNIT DIFF for dim=5 is 7.105427357601002E-15
GRAM DIFF for dim=7 is 5.064698660461886E-14
UNIT DIFF for dim=7 is 1.6431300764452317E-14
GRAM DIFF for dim=17 is 4.6851411639181606E-14
UNIT DIFF for dim=17 is 3.2807090377673376E-14
GRAM DIFF for dim=40 is 4.5630166312093934E-14
UNIT DIFF for dim=40 is 3.907985046680551E-14
```

Since the unitarity of the test data is exact — both quantum channels: the invariant Gram matrix of Section II C and the invariant unit matrix of Section II D recover the operator u_{jk} exactly.

---

[1] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychological review **65**, 386 (1958).

[2] V. Vapnik and A. Y. Chervonenkis, The method of ordered risk minimization, I, Avtomatika i Telemekhanika **8**, 21 (1974).

[3] P. Hájek and T. Havránek, On generation of inductive hypotheses, International Journal of Man-Machine Studies **9**, 415 (1977).

[4] V. Vapnik, *The nature of statistical learning theory* (Springer science & business media, 2013).

[5] I. H. Witten and E. Frank, Data mining: practical machine learning tools and techniques with Java implementations, Acm Sigmod Record **31**, 76 (2002).

[6] L. A. Zadeh, Fuzzy sets, Information and control **8**, 338 (1965).

[7] P. Hájek, Fuzzy logic and arithmetical hierarchy, Fuzzy sets and Systems **73**, 359 (1995).

[8] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence **35**, 1798 (2013).

[9] A. Bisio, G. Chiribella, G. M. D'Ariano, S. Facchini, and P. Perinotti, Optimal quantum learning of a unitary transformation, Physical Review A **81**, 032324 (2010).

[10] M. Arjovsky, A. Shah, and Y. Bengio, Unitary evolution recurrent neural networks, in *International conference on machine learning, NY, USA, 2016* (Proceedings of Machine Learning Research (proceedings.mlr.press), 2016) pp. 1120–1128.

[11] S. Hyland and G. Rätsch, Learning unitary operators with help from u(n), in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (Association for the Advancement of Artificial Intelligence (aaai.org), 2017).

[12] M. Razavy, *An introduction to inverse problems in physics* (World Scientific, 2020).

[13] J. R. Johansson, P. D. Nation, and F. Nori, QuTiP: An open-source Python framework for the dynamics of open quantum systems, Computer Physics Communications **183**, 1760 (2012).

[14] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[15] J.-y. Choi, S. Hild, J. Zeiher, P. Schauß, A. Rubio-Abadal, T. Yefsah, V. Khemani, D. A. Huse, I. Bloch, and C. Gross, Exploring the many-body localization transition in two dimensions, Science **352**, 1547 (2016).

[16] M. Yan, H.-Y. Hui, M. Rigol, and V. W. Scarola, Equilibration dynamics of strongly interacting bosons in 2D lattices with disorder, Physical review letters **119**, 073002 (2017).

[17] P. Bordia, H. Lüschen, S. Scherg, S. Gopalakrishnan, M. Knap, U. Schneider, and I. Bloch, Probing slow relaxation and many-body localization in two-dimensional quasiperiodic systems, Physical Review X **7**, 041047 (2017).

[18] M. Yan, H.-Y. Hui, and V. W. Scarola, Dynamics of disordered states in the Bose-Hubbard model with confinement, Physical Review A **95**, 053624 (2017).

[19] S. B. Ramezani, A. Sommers, H. K. Manchukonda, S. Rahimi, and A. Amirlatifi, Machine learning algorithms in quantum computing: A survey, in *2020 International joint conference on neural networks (IJCNN)* (IEEE, 2020) pp. 1–8.

[20] B. T. Kiani, *Quantum artificial intelligence: learning unitary transformations*, Ph.D. thesis, Massachusetts Institute of Technology (2020).

[21] Y.-M. Huang, X.-Y. Li, Y.-X. Zhu, H. Lei, Q.-S. Zhu, and S. Yang, Learning Unitary Transformation by Quantum Machine Learning Model., Computers, Materials & Continua **68**, 10.32604/cmc.2021.016663 (2021).

[22] G. Torlai, C. J. Wood, A. Acharya, G. Carleo, J. Carrasquilla, and L. Aolita, Quantum process tomography with unsupervised learning and tensor networks, Nature Communications **14**, 2858 (2023).

[23] S. Pai, B. Bartlett, O. Solgaard, and D. A. B. Miller, Matrix optimization on universal unitary photonic devices, Physical review applied **11**, 064044 (2019).

[24] V. G. Malyshkin and M. G. Belov, Market Directional Information Derived From (Time, Execution Price, Shares Traded) Sequence of Transactions. On The Impact From The Future, arXiv preprint arXiv:2210.04223 10.48550/arXiv.2210.04223 (2022).

[25] B. T. Kiani, S. Lloyd, and R. Maity, Learning unitaries by gradient descent, arXiv preprint arXiv:2001.11897 10.48550/arXiv.2001.11897 (2020).

[26] B. Kiani, R. Balestriero, Y. LeCun, and S. Lloyd, projUNN: efficient method for training deep networks with unitary matrices, Advances in Neural Information Processing Systems **35**, 14448 (2022).

[27] S. Lloyd and R. Maity, Efficient implementation of unitary transformations, arXiv preprint arXiv:1901.03431 10.48550/arXiv.1901.03431 (2019).

[28] X. Wang, C. Yang, and M. Gu, Variational Quantum Circuit Decoupling, arXiv preprint arXiv:2406.05619 10.48550/arXiv.2406.05619 (2024).

[29] V. G. Malyshkin, On Machine Learning Knowledge Representation In The Form Of Partially Unitary Operator. Knowledge Generalizing Operator, arXiv preprint arXiv:2212.14810 10.48550/arXiv.2212.14810 (2022).

[30] V. G. Malyshkin, On The Radon-Nikodym Spectral Approach With Optimal Clustering, arXiv preprint arXiv:1906.00460 10.48550/arXiv.1906.00460 (2019).

[31] T. A. Loring, Computing a logarithm of a unitary matrix with general spectrum, Numerical Linear Algebra with Applications **21**, 744 (2014).

[32] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, Nature Reviews Physics **3**, 625 (2021).

[33] C.-Y. Park and N. Killoran, Hamiltonian variational ansatz without barren plateaus, Quantum **8**, 1239 (2024).

[34] I. Najfeld and T. F. Havel, Derivatives of the matrix exponential and their computation, Advances in applied mathematics **16**, 321 (1995).

[35] A. Raza, Differentiating exponentials of Hamiltonians (2020), https://araza6.github.io/posts/hamiltonian-differentiation/.

[36] K. Lu and X. Guo, Efficient training of unitary optical neural networks, Optics Express **31**, 39616 (2023).

[37] K. Schäfers, M. Peardon, and M. Günther, A modified Cayley transform for SU(3), arXiv

preprint arXiv:2406.11337 10.48550/arXiv.2406.11337 (2024).

[38] A. Raza, Learning unitary matrices (2020), https://araza6.github.io/posts/unitary-learning/.

[39] G. H. Golub, Some modified matrix eigenvalue problems, Siam Review **15**, 318 (1973).

[40] M. G. Belov, V. V. Dubov, A. V. Filimonov, and V. G. Malyshkin, Quantum Channel Learning, arXiv preprint arXiv:2407.04406 10.48550/arXiv.2407.04406 (2024).

[41] G. Frison, J. Frey, F. Messerer, A. Zanelli, and M. Diehl, Introducing the quadratically-constrained quadratic programming framework in HPIPM, in *2022 European Control Conference (ECC)* (IEEE, 2022) pp. 447–453.

[42] V. G. Malyshkin and R. Bakhramov, Mathematical Foundations of Realtime Equity Trading. Liquidity Deficit and Market Dynamics. Automated Trading Machines, arXiv preprint arXiv:1510.05510 10.48550/arXiv.1510.05510 (2015).

[43] V. G. Malyshkin, The code for polynomials calculation (2014), http://www.ioffe.ru/LNEPS/malyshkin/code.html and an alternative location.

[44] K. Kraus, *States, Effects, and Operations: Fundamental Notions of Quantum Theory*, Lecture Notes in Physics, Vol. 190 (Springer-Verlag, 1983) Lectures in Mathematical Physics at the University of Texas at Austin.

[45] J. D. Brown, Singular Lagrangians and the Dirac–Bergmann algorithm in classical mechanics, American Journal of Physics **91**, 214 (2023).

[46] V. G. Malyshkin, Market Dynamics: On Directional Information Derived From (Time, Execution Price, Shares Traded) Transaction Sequences, arXiv preprint arXiv:1903.11530 10.48550/arXiv.1903.11530 (2019).

[47] L. Boudjemila, V. V. Davydov, and V. G. Malyshkin, On Quadratic Form Optimization Problem With Multiple Constraints of the Quadratic Form Type, in *The 5th International Conference on Future Networks & Distributed Systems, 2021, Dubai, United Arab Emirates* (Association for Computing Machinery acm.org, 2021) pp. 568–571.

[48] Å. Björck and S. Hammarling, A Schur method for the square root of a matrix, Linear algebra and its applications **52**, 127 (1983).

[49] N. J. Higham, Newton's method for the matrix square root, Mathematics of computation **46**, 537 (1986).

[50] Å. Björck, *Numerical methods for least squares problems* (SIAM, 2024).

[51] J. J. Dongarra and D. C. Sorensen, A fully parallel algorithm for the symmetric eigenvalue problem, SIAM Journal on Scientific and Statistical Computing **8**, s139 (1987).

[52] K. J. Maschhoff and D. C. Sorensen, P_ARPACK: An efficient portable large scale eigenvalue package for distributed memory parallel architectures, in *International workshop on applied parallel computing* (Springer, 1996) pp. 478–486.

[53] T. Katagiri and Y. Kanada, An efficient implementation of parallel eigenvalue computation for massively parallel processing, Parallel Computing **27**, 1831 (2001).

[54] J. Demmel, I. Dumitriu, and O. Holtz, Fast linear algebra is stable, Numerische Mathematik **108**, 59 (2007).